The Hong Kong Polytechnic University

Department of Computing

High Performance Publish/Subscribe Protocols for Wireless Ad Hoc Networks Using Geographic Information

by

YUAN ZHENG

A thesis submitted in partial fulfillment of the requirements for the Degree of Doctor of Philosophy

October, 2008

Certificate of Originality

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

_____ (Signature)

YUAN ZHENG (Name of Student)

Abstract

Wireless Ad hoc networks are emerging network paradigm which can be self-organized without any aids from the pre-established infrastructures. The networks have been widely used to support distributed, wireless, and mobile computing and communication applications. One of the major tasks of wireless ad hoc networks is to provide event service to the applications over the networks. Using event services, the applications can focus on producing and consuming events other than the transmitting and receiving events. Publish/Subscribe (pub/sub) is one of the most widely used middleware paradigm for event service in traditional distributed systems. However, wireless ad hoc networks have many unique characteristics different from traditional wired networks in terms of communication, mobility and resource constraints. These characteristics make the design of the pub/sub to be much more challenging.

In this thesis, we investigate how to design efficient publish/subscribe (pub/sub) protocols for wireless ad hoc networks. We review the existing pub/sub protocols designed for wireless ad hoc networks. Based on the review of previous studies, we proposed a 3-D framework for protocol design in wireless ad hoc networks. We describe three novel pub/sub protocols designed, respectively, for wireless sensor networks (WSNs), wireless mesh networks (WMNs), delay-tolerant networks (DTNs). The proposed protocols take the advantage of the geographical information of the network nodes to achieve high performance in event subscription and delivery.

First, we describe *HESPER*, a Highly Efficient and Scalable Publish/subscribe protocol designed for wireless sEnsoR networks. HESPER uses the geographical information of the interesting events to efficiently disseminate subscriptions without flooding and establish the event delivery paths shared by subscribers without the coordination among the subscribers. Thus, HESPER can reduce the amount of traffic

for event delivery and, thus significantly save energy of WSNs. To avoid rapid exhaustions of the nodes on the delivery paths, HESPER uses the rotation of intermediate nodes to balance the working loads for the nodes on the delivery paths. In addition, HESPER offers two different strategies to balance the tradeoff between the subscription and publication costs to satisfy the different requirements of the practical applications.

Then, we propose a novel PUb/sub protocol for wireless Mesh networks based on locAtion prediction, namely PUMA, which can reliably and efficiently deliver events to mobile clients in a continuous manner. Using the location prediction for a mobile client, PUMA can deliver the interesting events to the location where a mobile client may stay. We show that the problem of delivery events to these locations with a minimum message cost, namely all-location event delivery problem, is a NP-complete problem. To further reduce the message cost, PUMA first disseminate the events to the locations, namely hot locations, where a mobile client has a higher probability to stay. We show that the problem of delivering an event to hot locations with a minimum message cost, namely hot-location event delivery problem, is also a NP-complete problem. We propose two algorithms for the both of two NP-complete problems. Thus, PUMA can guarantee the reliability of event delivery with low cost.

Next, we propose a publish/subscribe (pub/sub) protocol, called GIANTs, that uses geographical information to improve the reliability and efficiency of event service for DTNs. Comparing with existing solutions, GIANT provides greater flexibility for applications in the sense that it allows recruiting randomly moving ferry nodes rather than only the ferries moving along specific directions. To guarantee the specified event delivery ratio, multiple ferries are needed to deliver the events, but this can lead to delivery redundancy. GIANT reduces the redundancy by first constructing an event delivery tree with optimal number of hops on the paths from the publishers to the subscribers, and then, using the location information of forwarding nodes to minimize the number of ferries to be recruited on each hop. Thus, in addition to flexibility and

reliability, GIANT also achieves high efficiency.

Keywords: wireless ad hoc networks; event service; publish/subscribe; geography information; wireless sensor networks; wireless mesh networks; delay-tolerant networks.

Publications

Journal Papers

1. **Yuan Zheng**, Jiannong Cao, *An Efficient Publish/Subscribe Protocol for Event Delivery in Delay-Tolerant Networks*, Submitted to IEEE transaction on Vehicular Technology.

2. Chao Song, Jiannong Cao, Ming Liu, **Yuan Zheng**, Haigang Gong, Guihai Chen, *Maximizing Network Lifetime Based on Transmission Range Adjustment in Wireless Sensor Networks*, Accepted by Computer Communications, Elsevier.

3. Ming Liu, Jiannong Cao, **Yuan Zheng**, Li Xie. *An Energy Efficient Protocol for Data Gathering and Aggregation in Wireless Sensor Networks*, Journal of Supercomputing, (2008) 43:pp107-125, Springer.

4. **Yuan Zheng**, Jiannong Cao, Alvin T.S. Chan, Keith K.C. Chan. *Sensors and Wireless Sensor Networks for Pervasive Computing Applications*, Invited paper by *Journal of Ubiquitous Computing and Intelligence*. Vol. 1, No. 1, April 2007. pp.17-34, American Scientific Publishers.

5. Deying li, Jiannong Cao, Ming Liu, **Yuan Zheng**, *Construction of Optimal Data Aggregation Trees for Wireless Sensor Networks*, Submitted to Computer Communications, Elsevier.

Conference Papers

6. Steven Yi Lai, Jiannong Cao, **Yuan Zheng**, *PSWare: a Publish / Subscribe Middleware Supporting Composite Event in Wireless Sensor Networks*, 5th IEEE PerCom International Workshop on Sensor Networks and Systems for Pervasive Computing (PerSeNS'09), Mar. 9-13, 2009, Galveston, Texas, USA.

7. **Yuan Zheng,** Jiannong Cao, Ming Liu. *RSQS: Resource-saving Multi-Query Scheduling*. Proc. IEEE Asia-Pacific Services Computing Conference (IEEE APSCC 2008), Dec. 9-12, 2008, Taiwan.

8. Jinqi Zhu, Jiannong Cao, Ming Liu, **Yuan Zheng,** Haigang Gong, and Guihai Chen. *A Mobility Prediction-based Adaptive Data Gathering Protocol for Delay Tolerant Mobile Sensor network.* Proc. IEEE Communication Conference (GlobeCom 2008), Nov.30-Dec.4, New Orleans, USA.

9. Chao Song, Jiannong Cao, Ming Liu, **Yuan Zheng,** Haigang Gong and Guihai Chen, *Mitigating Energy Holes Based on Transmission Range Adjustment in Wireless Sensor Networks*, Proc. 5th International Conference on Quality of Service in Heterogeneous Wired/Wireless Networks (QShine 2008), Jul. 28 – 31, 2008, Hong Kong.

10. Wanhong Wu, Jiannong Cao, **Yuan Zheng**, and Yong-Ping Zheng, *WAITER: A Wearable Personal Healthcare and Emergency Aid System*, Proc. First International Workshop on Pervasive Digital Healthcare (in conjunction with the Sixth Annual IEEE International Conference on Pervasive Computing and Communications - Percom-08), Mar. 17-21, 2008, Hong Kong.

11. **Yuan Zheng** and Jiannong Cao, *Highly Scalable and Efficient Publish/Subscribe Protocols Using Geographic Information for Wireless Sensor Networks*, Proc. Symposium on Middleware for Sensor Systems (MiSS'07), 3rd International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP'07) , Dec. 3 – 6 2007. Melbourne, Australia.

12. Deying Li, Jiannong Cao, Ming Liu and **Yuan Zheng**. *K-connected Target Coverage Problem in Wireless Sensor Networks.* Proc. of The 1st International Conference on Combinatorial Optimization and Application(COCOA'07), August 12-15, 2007, Xi'an, Shanxi, China**.**

13. Ming Liu, **Yuan Zheng**, Jiannong Cao, Wei Lou, Guihai Chen, and Haigang Gong. *A Lightweight Scheme for Node Scheduling in Wireless Sensor Networks*. Proc of the 4th International Conference on Ubiquitous Intelligence and Computing (UIC2007), July 11-13, 2007, Hong Kong, China.

14. Ming Liu, Jiannong Cao, **Yuan Zheng**, *An Energy-Aware Protocol for Data Gathering Applications in Wireless Sensor Networks*, Proc. 42nd Annual IEEE International Conference on Communications (ICC'07). June 24 – 28, 2007, Glasgow, UK.

15. Yi Lai, **Yuan Zheng**, Jiannong Cao. *Protocols for Traffic Safety Using Wireless Sensor Network*. Proc. of 7th International Conference on Algorithms and Architectures for Parallel Processing. June 11-14, 2007, Hangzhou China.

16. **Yuan Zheng**, Jiannong Cao, Ming Liu, Jinling Wang, *Efficient Event Delivery in Publish/Subscribe Systems for Wireless Mesh Networks*, 2007 IEEE Wireless Communications and Networking Conference (WCNC'07), March 11-15, 2007. Hong Kong, China.

17. Deying Li, Jiannong Cao, Ming Liu, **Yuan Zheng**, *Construction of Optimal Data Aggregation Trees for Wireless Sensor Networks*, the 15th IEEE International Conference on Computer Communications and Networks (ICCCN2006), Oct. 9 – 11, 2006 , Arlington, Virginia USA.

18. Vanessa W.S. Tang, **Yuan Zheng**, Jiannong Cao, *An Intelligent Car Park Management System based on Wireless Sensor Networks*, the 1st International Symposium on Pervasive Computing and Applications (SPCA2006), Aug. 3 – 5, 2006. Urumchi, Xinjiang, China.

19. Kirk H.M. Wong, **Yuan Zheng**, Jiannong Cao , Shengwei Wang, *A Dynamic User Authentication Scheme for Wireless Sensor Networks*, the 1st IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC2006), June 5-7, 2006, Taichung, Taiwan.

Acknowledgements

I am deeply grateful to Professor Jiannong Cao for his rigorous supervision of my research. I thank him for his guidance, patience and encouragement during my Ph.D. study. It is the unremitting training from him that makes me to be qualified PH.D and independent research. He teaches me so many things, such as hunting research issues and topics, establishing research framework, finding interesting and important problems, solving thorny problems. So many times, he patiently tells me how to clearly express my ideas and write high-quality academic papers. His knowledge, vision, passion, and professional spirit towards the research have a significant impact to me. What I have obtained and experienced during my PH.D. study will always help and encourage me in my life.

I thank Ming Liu, Yang Liu, and Yi Lai for our collaborations on the research and projects. We explored ideas and wrote papers together. I also greatly thank Weigang Wu, Xiaopeng Fan, Miaomiao Wang, Kirk Wong, Gang Yao, and all other members of Prof. Cao's research group that I cannot enumerate here. I thank them for interesting discussions and suggestions on my work.

Also, I wish to acknowledge my appreciation to Yi Xie, Meng Wang, Lei Ye, Xiapu Luo, and Xie Jin, who shared with me the pain and pleasure of the PH.D. study at the Hong Kong Polytechnic University.

Last, but not least, I would like to thank my family. Their love, understanding, support and encouragement were the most important ones that help me finish the thesis.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AD	advertisement-driven
ALD	all-location event delivery
AN	all nodes
BS	broker selection
CBR	content-based routing
DD	directed diffusion
DF	diffusion filter
DTN	delay-tolerant network
FL	flooding
FT	fault-tolerant
GE	geography-aided
HLD	hot-location event delivery
LA	location-aware
LDO	lowest delivery overhead
LF	limited flooding
MANET	mobile ad hoc network
MD	message dissemination
ON	one node
PE-DD	power-efficient data dissemination
PD	publish/subscribe driven
PHB	publisher hosting broker
Pub/Sub	Publish/Subscribe
RAFT	reliable and fault-tolerant
SD	subscription-driven
SDCT	steiner data cache tree

SDP	shortest delivery path
SDP_LOD	shortest delivery path with lower delivery overhead
SHB	subscriber hosting broker
SL	structure-less
SN	selective node
TR	transmission range
VANET	vehicle ad hoc network
WMN	wireless mesh network
WSN	wireless sensor network

Chapter 1

Introduction

The main objective of this research is to investigate high performance publish/subscribe (Pub/Sub) protocols for wireless ad hoc networks. In this thesis, we describe three novel pub/sub protocols designed, respectively, for three different kinds of wireless ad hoc networks, including wireless sensor networks (WSNs), wireless mesh networks (WMNs), delay-tolerant networks (DTNs). The proposed protocols take the advantage of the geographical information of the network nodes to achieve high performance in event subscription and delivery.

In this chapter, we first give an introduction to wireless ad hoc networks, including their definitions, specific properties, and classifications. Then, we discuss the basic concepts of publish/subscribe middleware that will be used throughout the rest of the thesis. We also describe the most challenging issues faced in the design of pub/sub protocols for wireless ad hoc networks. At the end of the chapter, we summarize the contributions of this research and outline the organization of the thesis.

1.1 Wireless Ad hoc Networks

Wireless Ad hoc networks are emerging network paradigm which can be self-organized by themselves [CHA01] [ROY99] [AKY05] [AKY02] [AKK05]

without any aids from the pre-established infrastructures. A wireless ad hoc network usually is composed of a large number of nodes equipped with computing, storage, and wireless communication devices. These nodes can be easily and rapidly deployed into various environments, such as indoor spaces, metropolitan areas, high ways, battle fields, wild forests and desserts, and cooperatively perform the computation, storage, and wireless communication tasks to support wireless, distributed, and mobile computing and communication applications.

1.1.1 Properties of Wireless Ad hoc Networks

Compared with traditional wired networks and wireless networks, a wireless ad hoc network has the following specific properties:

- *Mobility*. The nodes in MANETs and Mobile WSNs and the clients or some base stations in WMNs can be deliberately and elaborately moved in the deployment region of the networks.
- *Low capabilities*. The nodes in wireless ad hoc networks are usually hand-held or pocket-loaded devices which can only equipped with less powerful computation, storage, wireless communication units than desktop or laptop computers.
- *Limited energy*. Due to tiny-size and portability of the wireless ad hoc nodes, these nodes can be powered by battery and are easily or low-costly be recharging while moving.

- *Short transmission range*. The limited power of the wireless ad hoc nodes is strongly restrict the wireless transmission range.
- *Dynamic topologies*. The topological variation of an wireless ad hoc network usually occurs, because of the mobility of networking nodes, the short transmission range, the surrounding interferences, the energy exhaustion of nodes, or the broking caused by unexpected ambient forces.
- *Position awareness*. The nodes in wireless an ad hoc network are usually equipped with a positioning device since the applications increasingly tends to location awareness.
- *Heterogeneous hardware*. Because the large number of nodes in a wireless ad hoc network, the nodes in wireless an ad hoc network usually are the products from different manufacturers and even follow different industrial standards.

1.1.2 Classification of Wireless Ad hoc Networks

As shown in Fig. 1.1, we can classify wireless ad hoc networks according to three orthogonal dimensions, namely functionalities, mobility of nodes, and connectivity of networks. According to the functionalities of the networks, we can categorize wireless ad hoc network into three major different types, including mobile ad hoc networks (MANETs) [ROY99], wireless sensor networks (WSNs) [AKY02], and wireless mesh networks (WMNs) [AKY05].

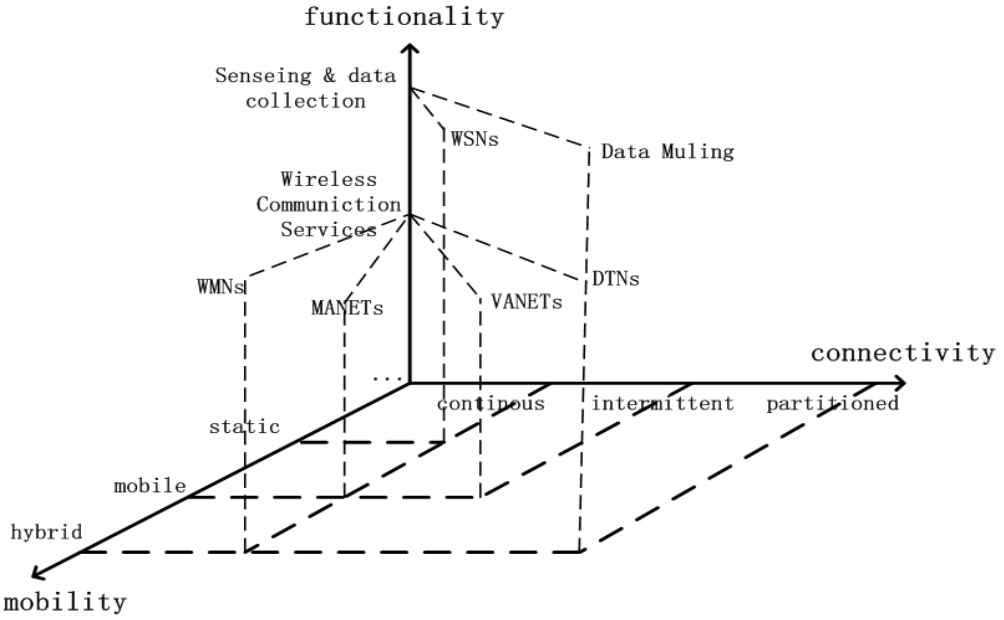


Fig. 1.1 the categorization of wireless ad hoc networks

A MANET is a kind of wireless ad hoc networks which are composed of mobile nodes. Using the attached computation, storage, and wireless communication devices, these nodes can cooperatively finish the data transmission, computation, and data storage tasks during movement and then, provide the wireless communication services. One of the special MANETs is sparse MANETs, also called a Delay-Tolerant Networks (DTNs), where a limited number of nodes are deployed in a large area. The communication path between a pair of nodes is usually disconnected or the wireless links on such a path is never connected contemporarily. MANETs have many application examples, such as vehicular ad hoc networks, battle-field communication networks, ship fleet networks, satellite networks, etc.

A WSN is composed of a large number of tiny-size sensor nodes, which are equipped with sensors, CPUs, memories, and wireless transceivers. These nodes can be deployed in various environments to cooperatively sense, collect, process, and transmit data. After the operations of the data, the sensor nodes will transmit to a sink node, which is a gateway to connect the WSN to the applications deployed on

the Internet. One of the special kind of WSNs are mobile-sink WSNs, where the sink nodes can proactively move in the WSNs to collect the data from the static sensor nodes. Mobile sensor networks is a special kind of WSNs, where the sensor nodes can move around, float in water or air, or attached on mobile objects to collect more information or provide the surveillance in a large area or space. A wireless sensor network has many applications, such as, wild environment monitoring systems, intelligent building monitoring systems, battle-field surveillance systems, intelligent elderly and personal healthcare systems, and structure health monitoring systems.

A WMN is composed of a collection of base stations, which are equipped with high performance computation and storage devices as well as high power wireless transceivers. The base stations are inter-connected by wireless links and cooperatively transmit data for users. This emerging networking paradigm is regarded as a promising solution to provide wireless Intranet and Internet services to users in a large area, such as high ways, large-scale communities, large-size squares, high buildings, and metropolitans.

According to the mobility of nodes, wireless ad hoc networks can be classified into static, mobile, and hybrid networks. Most of WSNs are static wireless ad hoc networks. Usually, sensor nodes reside in the deploying locations and keep monitoring the physical phenomenon or objects and collect interesting data. Another kind of static ad hoc networks usually provides wireless communication services. For example, in a conference room, the laptops of conference participants can be temporarily organized into a wireless ad hoc network to share the Internet access in the room. Because the participants do not move during a conference, such a wireless ad hoc network is a static one.

Usually, a MANET can be regarded as the mobile one, where the nodes can move freely in the deploying region of the networks. In some applications, WSNs can also be a kind of mobile wireless ad hoc networks. Sometimes, people need to deploy the sensor nodes to collect the physical properties of some mobile objects, such as the quality and temperature of air, the pollution of rivers, the speeds and directions of see currents [ZHE07]. In such applications, the sensor nodes need to move with the monitored objects and then, compose mobile WSNs.

WMNs are usually composed of static base stations and mobile clients and thus, are regards as hybrid wireless ad hoc networks. Usually, a WMN's backbone composed of base stations is stable, but the connections between a base station and a client are usually changed frequently. Mobile-sink WSNs is also a kind of hybrid wireless ad hoc networks. In some special applications of WSNs, the sink nodes do not reactively keep static to receive the data transmitted by sensor nodes but proactively move in the network to visit all sensor nodes for data. The examples of mobile-sink WSNs include WSNs with data-muling [JEA05], ferry-based WSNs [FAR06], etc. Another example of hybrid wireless ad hoc networks is vehicle ad hoc networks (VANETs) [WU04] [ZHA06-2]. A vehicle ad hoc network (VANET) is composed of the computation and wireless communication nodes attached on mobile vehicles and the static computation and wireless communication unit resided at road side (RSU). The data can be transmitted in a VANET by vehicle-to-vehicle and vehicle-to-RSU manners, respectively.

According to the connectivity, wireless ad hoc networks can be categorized into always connected, intermittently connected, and disconnected ones, where the intermittently connected and disconnected ones are also called Delay-Tolerant Networks (DTNs). Most of the wireless ad hoc networks are always connected. In

such a network, a node can establish a data transmission path to another node whenever this node needs to communicate with another node. However, sometimes, the network will be intermittently connected due to the mobility of nodes, the energy exhaustion, or some other ambient forces. In such networks, a mobile node can only transmit data in a carry-forward manner, where a mobile node will temporarily carry a data, carry the data, continuously move, and finally, forward the data to a suitable node met by this node during movement. One example is sparse MANETs. A MANETs with low node density may be not always connected and then, a node in the network may not find a path to another node for data transmission. Thus, the two nodes can perform wireless communication only in an opportunistic manner. Similar to sparse MANETs, VANETs is one example of intermittently connected wireless ad hoc networks. The vehicles moving on roads may have a very distance between each other and may not keep the wireless links with other vehicles.

In disconnected wireless ad hoc networks, the static nodes are disconnected with each other and mobile nodes move among these static nodes to receive the data, carry the data and move to another static node, and then, forward the data to the target static nodes. A well-known example is inter-village networks [SET06]. In some remote villages, there is no infrastructure to provide Internet access service, but there are some vehicles, such as buses, cars, or motors, shuttling between the village and the metropolitans with Internet access points. Then, these villages can only use the vehicles to carry the data, which need to be forwarded, to the metropolitan and then, send them out via the Internet access points there. Meanwhile, these vehicles will obtain the data target at these villages from the access point and then, carry them back to the village, when the vehicles return to the villages. Another well-known example is data ferry for WSNs, where the sensor nodes are deployed in

the river and are disconnected. A ferry equipped with wireless communication devices moves along the river. When the ferry bypasses the sensor nodes deployed in the river, it will collect the sensory data cached in the sensor nodes.

1.2 Publish/Subscribe Middleware

Middleware is highly recognized by its ability of improving the feasibility and availability of complicated large-scale systems, because middleware can effectually hide the implementation details irrelevant to applications and effectively abstract the programming interfaces to facilitate the development and deployment of the applications. Middleware can be even more important to the mobile computing applications over wireless ad hoc networks, since it can effectively hide the details of the underlying wireless ad hoc networks and significantly alleviate the challenging problems caused by the networks for the developers of the upper layer applications.

In wireless ad hoc networks, one of the major tasks of middleware is to provide event service for the upper layer mobile computing and wireless communication applications. Using event services, the applications can focus on producing and consuming events other than the transmitting and receiving events. Publish/subscribe (pub/sub) is one kind of the most widely used middleware for event service in traditional distributed systems [CAR00][CAR01][EUG03], as shown in Fig. 1.2. A pub/sub system may contain many entities that can be classified into two categories, including clients and brokers. A client can be a publisher, a subscriber, or both of them in a pub/sub system, which generates and consumes events, respectively. The

brokers, also called servers, are responsible for disseminating the events into distributed systems and notifying the client to receive the interesting events. A client usually registers itself on a broker at first and then it can publish its events, subscribes some events and receives the interesting events via the broker. The Pub/Sub middleware can effectively realize the spacing, timing and flowing decoupling in the event services. In other words, this paradigm does not require that the publishers and subscribers necessarily know each other, they actively participate in the interaction at the same time, and they synchronously transmit or receive events. Thus, this middleware paradigm can be a suitable solution to satisfy the requirements of event service in wireless ad hoc networks.

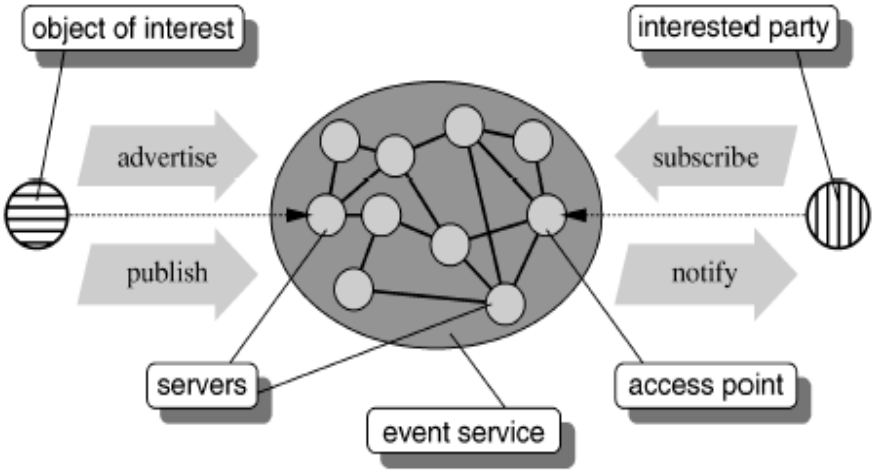


Fig. 1.2 pub/sub middleware[CAR01]

A pub/sub system for a wireless ad hoc networks usually is deployed between the application layer and the network layer, as shown in Fig. 1.3. The application in the upper layer usually use four operators, including pub(), sub(), adv(), and notify() to communicate with the pub/sub systems. The specific functions of the four operators are defined as follows, respectively.

- Adv() is used by the applications to send the pub/sub system the description of the events, which will be generated by the applications.
- Pub() is used by the applications to send the pub/sub system the generated events .
- Sub() is used by the applications to send the pub/sub system the description of the events, which the applications are interest in.
- Notify() is used by the pub/sub system to send the applications the events which the applications are interested in.

Using the four operators, the applications can easily and simply publish and subscribe the events via a pub/sub system. In addition, a pub/sub system needs to communicate with the underlying wireless ad hoc network. Usually, the pub/sub system will use the network drivers to communicate with the networks and the networks can help the pub/sub system to collect the data in the events and transmit the events.

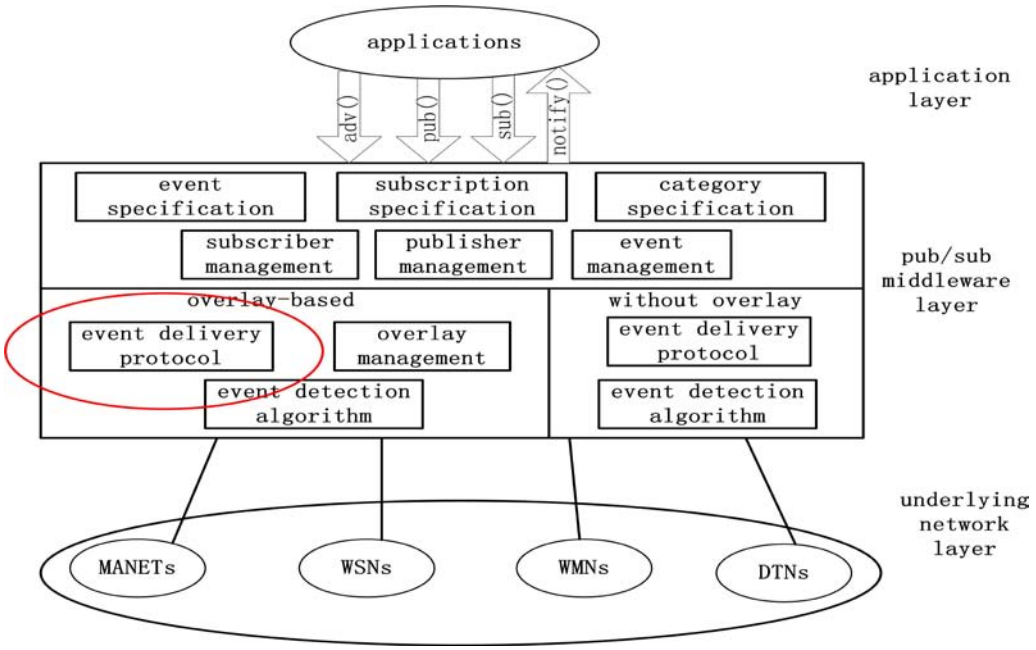


Fig. 1.3 pub/sub system architecture for wireless ad hoc networks

A pub/sub system for a wireless ad hoc networks is usually divided into two sub-layers (see Fig. 1.3). The upper sub-layer, namely specification and management layer, mainly has 6 components, including event specification, subscription specification, category specification, subscriber management, publisher management, and event management. The specific functions of the components are as follows.

- *Event specification* specifies the formal description of the events used in the pub/sub system. The event specification usually defines the fields in an event and the specific function of the corresponding fields.

- *Subscription specification* is used to specify the formal description of the subscription used in the pub/sub system. The subscription specification usually defines the fields in a subscription and the specific functions of the corresponding fields.

- *Category specification* defines the specific category of a pub/sub system, which usually can be a topic-based, content-based, or type-based system

[EUG03]. In a pub/sub system, different category will use different matching rules. According to the matching rules, the pub/sub system can determine whether an event can satisfy a specific subscription.

- *Subscriber management* is used to maintain the relationship between the subscribers and the subscriptions submitted by the subscribers. Using subscriber management, a pub/sub system will know how to send the events to the corresponding subscribers.

- *Publisher management* is used to maintain the relationship between the publishers and the events generated by the publishers. Using publisher management, a pub/sub will know what events can be generated by a specific publisher.

- *Event management* is used to event storage and deletion. Sometimes, the events cannot be directly sent to the subscribers due to the disconnection of the subscribers. The events should be temporarily cached in the system. In addition, an event can be expired in a pub/sub system and the pub/sub need to remove the obsolete events.

The underlying sub-layer in a pub/sub system, namely event delivery layer, can be categorized into overlay-based and non-overlay-based types, respectively. Since the most important functions of a pub/sub system include how to disseminate the advertisement to the subscribers, how to disseminate the subscriptions to the publishers, how to determine the occurrence of an event, and how to deliver a detected event to the interested subscribers. Both of the two types of the sub-layer should have advertisement dissemination, subscription dissemination, event detection, and event delivery components, respectively. The specific functions of the

four components are as follows, respectively.

- *Advertisement dissemination* component is used to distribute the event advertisements generated by the publishers in the system and guarantee the receipt of the subscribers.
- *Subscription dissemination component* is used to distribute the subscriptions generated by the subscribers in the system and thus, the system will know the interests of the subscribers.
- *Event detection component* usually is used to determine the occurrence of the composite events, which are composed of a number of simple events published by the applications with a spatial and temporal relationship. Since the applications in a distributed system will publish the events in different time and at different locations. Thus, the pub/sub system needs to use the specific protocols and algorithms to collect the distributed occurrences of the simple events and then, determine the occurrence of the composite events.
- *Event delivery component* is used to establish and maintain the event paths for the system to deliver the events generated by the publishers to the subscribers.

In an overlay-based pub/sub system, some nodes, namely broker, can be employed specifically to be responsible for the event detection and delivery. Thus, in a broker-based pub/sub system, there is a component for the broker management. This component is responsible for the broker selection, organization, and maintenance.

In this thesis, we focus on the event delivery component, as shown in Fig. 1.3. Since the event delivery is the most important task in a pub/sub system, the

performance of the event delivery component have significant impact on the performance of the whole pub/sub system. However, previous studies [BAC00][BAN99][CAR01][PIE02] on pub/sub protocols for the traditional distributed systems have not taken full account of the specific properties of wireless ad hoc networks. The specific properties have posted new challenges to the design and implementation of pub/sub protocols for wireless ad hoc networks.

1.3 Challenging issues in the Design of Pub/Sub Protocols for Wireless Ad hoc Networks

There are many previous studies on the design and implementation of pub/sub protocols for traditional networks. However, because wireless ad hoc networks have several unique properties, the existing pub/sub protocols for the traditional networks are not suitable to wireless ad hoc networks. More specifically, the design of pub/sub protocols needs to face four major challenges, including reliability, high energy efficiency, scalability, and low latency. We discuss these challenges in details respectively in this section.

- *Reliability.* The applications using event services requires that the all of interesting events need to be delivered to the applications without loss. Usually, the communication among the nodes in wireless ad hoc networks cannot be reliable as that in wired networks. The unreliability of the wireless communication can considerably affect the effectiveness and efficiency of the pub/sub protocols. In addition, the design of pub/sub protocols for different

wireless ad hoc networks need to faces different challenges on reliability, which are caused by specific properties of different wireless ad hoc networks. In a MANET, all nodes keep moving and the distance between a pair of nodes, which is performing a wireless communication, is frequently beyond the range of wireless transmission. Similarly, the communication reliability between a client node and base station in WMN also suffers from the mobility. Another example is that, in DTNs, the mobile nodes have not connected communication paths among them and can only conduct opportunistic communication. Therefore, the wireless links among the nodes are highly frequently broken. As for a wireless mesh network, the number of wireless channels used by a WMN is very limited and the communication range of a mesh node is very large. Thus, there will be high interferences among these channels, when the different nodes use the same channels for wireless transmission simultaneously. Another concern in WMNs is the mobility of clients. The clients move in the network and switches among different mesh nodes. Thus, during the switches, the mobile clients can be disconnected from the WMNs temporarily. In a WSN, the wireless transceivers of sensor nodes are low-power devices and thus, the wireless signal power are also very low. The wireless transmission can be greatly interfered by the environmental factors, such as obstacles, raining, or strong wind. In addition, the wireless sensor nodes can be easily invalid with the reasons of the energy exhaustion and the damage of the nodes. Both of the reasons can disable the wireless communication.

- *Energy Efficiency.* A pub/sub designed for a wireless ad hoc network should consume the nodes' energy as little as possible. This is because the nodes in MANETs and client nodes in WMNs are usually portal devices and the nodes in

WSNs are wireless devices scattered in different environments. These nodes are usually powered by the batteries attached on these nodes. Because these nodes usually have a strictly limitation on their sizes, the energy supply of the attached batteries is very limited. Further, the previous studies show that the wireless transceivers on a node consume a large portion of the node's energy. Thus, the design of a pub/sub for wireless ad hoc networks needs to reduce the amount of wireless transmission of control messages for the protocols as much as possible to save the energy. In addition, the pub/sub protocols should reduce the on-duty time on the nodes as much as possible to avoid the energy consumption of wireless transceivers on the idle status.

- *Scalability.* The number of nodes in a wireless ad hoc network or the size of the deployment region of the network should not affect the performance of a pub/sub protocol. One example is that the sensor nodes in a WSN may be deployed in very large geographical areas and the number of the nodes in a network usually is over a hundred or a thousand. These post challenges on the design of pub/sub protocols in terms of scalability. In wired distributed systems, the performance of a pub/sub protocol can be guaranteed or optimized based on the knowledge of the global information of the systems. However, wireless ad hoc networks, especially WSNs, usually need to pay a very high cost to perform the global information collection, or sometimes, such a global information collection is impossible. To guarantee the high performance of a wireless ad hoc network, a pub/sub protocol for these network usually are based only on the local information or a partial of networks' information.

- *Low Latency.* The applications usually require that an interesting event can be delivered to them in a short time. One example is that the intelligent traffic

system over WSNs needs to send the real-time traffic events to different vehicles. Such kinds of real-time events also occur in WMNs and should be delivered with low latency. Another one is that, in DTNs, the opportunistic communication used by the network may cause very large delay in event delivery and, finally, cause the events to be obsolete. However, due to the low communication reliability in a wireless ad hoc network, an event needs to be re-transmitted several times from a publisher to a subscriber. This will cause a long time to finish the event delivery. In addition, the scale of a wireless ad hoc network is often very large in terms of network size and deploying region. An event delivery path usually is composed of the large number of hops and causes a high delivery delay on the path. Another factor cause high latency is energy efficiency. Aiming at saving energy, many nodes will not continuously work in a wireless ad hoc network and keep sleeping in most of the time. Thus, an interesting event can only be delivered when the intermediate nodes or the receivers is in on-duty status; otherwise, the event sender can only locally cache the event and wait until the next-hop node is waken up. Such a waiting time can also cause the delivery delay higher.

1.4 Contributions of The Thesis

We aim to design novel pub/sub protocols and algorithms to satisfy the specific requirements of the wireless ad hoc networks, solve the problems resulted from the challenging issues in wireless ad hoc networks, and meanwhile, achieve high performance in terms of reliability, energy-efficiency, scalability, or low latency. In

this section, we first briefly discuss the limitations of previous works on pub/sub protocols for different wireless ad hoc networks. Then, we describe three different pub/sub protocols designed for WSNs, WMNs, and DTNs, respectively.

Although there is a large number of pub/sub protocols designed for traditional wired networks, these protocols are not suitable for WSNs due to the low reliability caused by wireless communication used in WSNs. In addition, most of the existing protocols used in traditional wireless networks and MANETs used the flooding to disseminate the subscriptions and events. The flooding will cause very high energy consumption in a WSN, which causes rapid exhaustion of a network. In WSNs, most previous Pub/Sub protocols employ subscription flooding in the whole network[HEI01][INT00] [MAR02][COS05-2][SOU05] or in the existing pub/sub trees[PRA05], which suffers from poor scalability and incurs high energy costs. To avoid subscription flooding, D. Estrin, *at el*, proposed Diffusion Filters (DF) [HEI02], which employs geographic energy-aware routing (GEAR) [YU01] to disseminate event subscriptions. However, DF establishes independent event delivery paths for different subscribers, where the events are delivered replicated and consequently, cause high energy consumption. Given a publisher and multiple subscribers in a WSN, we aim at minimize the amount of wireless transmission for events. Such a problem has been adequately addressed as a Stenier Minimal Tree problem, where the event delivery paths from a publisher to all subscribers are organized into a tree and the event delivery in this tree needs minimum cost. However, in a pub/sub tree, the messages transmitted include not only the events, but also pub/sub control messages, and the messages for tree establishment. We want to minimize the total amount of the wireless transmission for such messages to save energy. In addition, we also want to use only localized topology information in the

protocols and algorithms, which can effectively help pub/sub protocols to achieve high scalability.

The existing pub/sub protocols designed for traditional wired networks are yet not suitable to WMNs, because WMNs are quite different from the traditional distributed systems due to the existence of a large number of mobile clients and multi-hop wireless communications between network nodes. Recently, some solutions have been proposed for designing Pub/Sub systems in the cellular networks [PIT97] to support mobile clients [BUR04][FIE03][ION04][POD04] [SUT01][TAR03][WAN05]. However, all these works address only the problem of the clients' reconnection, where the clients were disconnected from the base station and re-connect to another base station in a new place after movement. Different from cellular networks, WMNs need to handle the problem of how to provide continuous event service to the mobile clients, which is much more challenging due to the reliability, system message cost, and energy efficiency of the mobile clients in event delivery. In addition, most of existing pub/sub protocols for MANETs employs the flooding to leverage the highly dynamic topology generated by mobile nodes. However, the base stations of WMNs usually do not move and then, the flooding used by MANETs should be avoid for reducing the message cost and network resource consumption.

The existing pub/sub protocols designed for traditional wired networks and traditional wireless networks are developed based on the connected underlying networks, which is quite different from the intermittent connectivity of DTNs. All of these protocols have not provided the mechanism to handle the frequent partitions of networks and cannot guarantee the system performance when the networks are suffering from the frequent disruptions. Thus, these previous works cannot be

simply used in DTNs. Although some previous works in MANETs used the mechanism based on flooding to handle the intermittent connectivity, the mechanism is not suitable to DTNs. This is because a DTN usually has very limited network resources, for example, the mobile nodes for message sending, which cannot support the flooding in the whole network. SAFIRE system and socio-aware pub/sub system are claimed to design specifically for DTNs, However, both of the works are established based on overlay networks, which are built on the underlying physical DTNs. The establishment and maintenance of an overlay network is highly cost, especially over a large-scale delay-tolerant network.

Taking the advantages of the position awareness of the nodes, the proposed pub/sub protocols can achieve more attractive performance in terms of efficiency, scalability, and reliability, in comparison with the previous works. This thesis has the following specific contributions.

1) We propose *HESPER*, a Highly Efficient and Scalable Pub/sub protocol in wireless sEnsoR networks, which HESPER outperforms the previous protocols in terms of scalability and energy-efficiency by taking the following advantages. First, in HESPER, using the geographical information of the interesting events, the subscribers can efficiently subscribe the events without flooding. Second, HESPER can support event pub/sub for multiple subscribers and publishers in a distributed manner. According to the locations of subscribers and publishers, HESPER can establish the event delivery paths, which greatly overlap with each other and are shared by the subscribers in event delivery. Since HESPER does not need the coordination among the subscribers to establish the delivery paths, it can significantly reduce the amount of event delivery and thus, save energy. Third, HESPER uses the

intermediate nodes rotation to balance the working loads for the intermediate nodes on the delivery paths. This can avoid the rapid exhaustions of the nodes on the delivery paths and the consequent delivery disruptions. Fourth, HESPER offers two different strategies to balance the tradeoff between the subscription and publication costs to satisfy the different requirements of the practical applications.

We have conducted the theoretical analysis and the extensive experimental evaluation to show the performance advantages of HESPER in compared with the previous solutions. Our extensive simulation results show that, in comparison with the previous work of DF and SDCT, HESPER can save event delivery energy up to 50% and 80% respectively, and meanwhile, save the event subscription energy up to 50% and 90%, respectively. In addition, our theoretical analysis shows that, measured by hop count, the length of delivery path established by HESPER between a publisher node and a subscriber node is at most 0.8 time larger than the theoretical shortest path. The experimental results show the average length of the delivery path in HESPER is less than l.5 times of the delivery path established by DF. Considering the significant energy saving in HESPER, such a little bit larger delivery latency of HESPER is still quite reasonable and attractive.

2) We address the problem of how to support continuous event delivery for mobile clients in a pub/sub system for WMNs. Aiming at solve the problem, we propose a reliable and efficient PUb/sub protocol for wireless Mesh networks based on location prediction, namely PUMA. Using the location prediction, PUMA can estimate all of the candidate locations where the interested client possibly stays and then, deliver the events to all of such candidate locations. We

show that such an event delivery with minimum delivery cost is a Minimum Steiner Tree (MST) problem, which is NP-hard. However, according to the calculation of the possibility of a client staying in a grid, we can find that a client has a low possibility to stay in partial of the candidate locations, namely cold locations. Delivery events to the cold locations may cause high unnecessary message cost due to the large number of the cold locations. To further reduce the message cost caused by cold locations, PUMA defines a probability threshold and at first, delivers the events to the locations, namely hot locations, where the sum of the possibility of a client staying in is larger than the probability threshold. Once the client is not hot locations, PUMA will then deliver the events to the cold locations. Therefore, PUMA can guarantee the reliability of the event delivery and meanwhile, achieve high efficiency. We can show that, given a probability threshold, delivering an event to satisfy the threshold is still a NP-hard problem. In addition, we proposed an algorithm for the problem with polynomial time complexity. We conduct extensive simulations to evaluate the performance of our protocol and algorithm. The simulation results show that, in comparison with the paging algorithms for clients' mobility management in cellular networks, PUMA can save up to over 90% system message cost and over 60% client message cost, respectively.

3) We propose GIANT, a Geography-aIded pub/sub protocol in delAy-tolerant NeTworks, to address the above three issues. First, unlike the previous solutions employing only specific ferries moving in a controlled manner, GIANT can recruit randomly moving ferries to conduct reliable, efficient, and timely event delivery without disruption of ferry mobility. This makes GIANT more feasible and flexible to support applications, because it allows recruiting

randomly moving ferries, which can be ordinary vehicles and people attaching the portable wireless communication and computation devices and without specific mobility pattern. The second feature of GIANT is that it can guarantee user specified event delivery ratio by recruiting multiple randomly moving ferries. However, recruiting multiple ferries can cause redundant event delivery and therefore, incur high resource consumption. GIANT uses the geographical information of subscribers and publishers to minimize the number of such ferries and reduce resource consumption. The third feature is that GIANT can reduce the event delivery delay by establishing the delivery paths with only 2 or 3 hops. It is a widely accepted observation in the traditional wireless networks that the data transmission with fewer hops can achieve higher performance. However, upon the simulations, we found that the event delivery with 2 or 3 hops usually outperforms that with single hop or other numbers of hops, especially in terms of delivery delay. We have conducted simulations to evaluate the performance of GIANT in comparison with the centralized optimal minimum spanning tree (MST) algorithm, which is well-known for establishing the optimal event delivery tree in networks. The results show that, in terms of the important metrics of resource saving and delivery latency, GIANT can recruit 30% less number of ferries and reduce delivery latency up to 50%, respectively.

1.5 Outline of the thesis

The remaining chapters of this thesis are organized as follows. Chapter 2 briefly

discusses the previous works on the relevant topics. Chapter 3 describes the architecture of a pub/sub system and discusses the geographic information used in the design of pub/sub protocols. Chapter 4 presents a Highly Scalable and Energy-efficient Pub/Sub protocol for Wireless Sensor Networks, namely HESPER. Chapter 5 presents an Efficient Pub/Sub protocol for Wireless Mesh Networks based on location prediction, namely PUMA. Chapter 6 presents a Geography-aided Pub/Sub Protocol for Delay-tolerant Networks, namely GIANT. Chapter 7 gives the conclusions and a discussion of future works.

Chapter 2

Literature Review

In this chapter, we review the existing pub/sub protocols designed for wireless networks, including traditional wireless networks and wireless ad hoc networks. The organization of this chapter is as follows. Firstly, Section 2.1 reviews the existing pub/sub protocols for traditional wireless networks. Section 2.2 depicts the previous pub/sub protocols designed for wireless ad hoc networks, where the previous studies are discussed for MANETs, WSNs, WMNs, and DTNs, respectively. Section 2.3 discusses the 3-D framework for designing a pub/sub protocol for wireless ad hoc networks based on the three orthogonal strategies, including message dissemination, broker selection, and pub/sub driven strategies.

2.1 Pub/Sub Protocols in Traditional Wireless Networks

Traditional wireless networks, also called cellular networks [PIT97], are composed of wired-connected base stations and wireless client devices, where the data transmission between base stations is through wired links and the transmission between a base station and a client is through wireless links. In traditional wireless networks, the wireless client devices are usually hand-held devices of users and moves around with the users. Thus, one of the challenging issues on the design of pub/sub protocol in such a network is user mobility. Although a large number of works have been conducted on Pub/Sub middleware for the traditional distributed

computing systems, such as SIENA [CAR01], CEA [BAC00], and Gryphon [BAN99], these works have not considered the problem on how to support mobile clients. In recently, there are some novel protocols proposed to support mobile clients.

In traditional wireless networks, the events usually cannot be delivered to the mobile user immediately during the movement of the users. Then, the system should use an effective mechanism to temporarily cache the event and delivery the event to the mobile subscribers by using a handover protocol after the mobile users re-register to the pub/sub system. Such a mechanism can be classified into centralized and distributed ones, respectively.

One example of the centralized mechanism is the extension version Elvin[SEG00] for supporting mobile clients, which is proposed by P. Sutton et al. [SUT01]. The main idea is to employ a central proxy that tackles subscriptions for disconnected clients. After reconnecting to the system, a mobile client will first connect to the central proxy to obtain the events published during their leaving. The central proxy, however, tends to become a performance bottleneck and the system does not have good scalability.

However, many existing works use a distributed mechanism to handle the mobility of subscribers to alleviate the performance bottleneck and poor scalability caused by the centralized mechanism. According to where the events are cached, the distributed mechanism can also be classified into two categories, including publisher-caching and broker-caching ones, respectively. In publisher-caching mechanisms, a publisher will cache all the events published by it. After re-connection, the mobile clients will obtain the events published during their

mobility from the publishers. In broker-caching mechanism, if a broker has a subscriber moving away, the broker will cache all the received events subscribed by the subscriber. Then, the subscriber can re-connect the system and take all the events from the broker.

Broker-caching mechanism can also be classified into two different kinds, including previous-broker-caching and all-broker-caching. Using the previous-broker-caching, a broker will cache the events subscribed by a mobile client that once served by the broker but now, moves away. When the client reconnects to the system, it will obtain the cached events from the previous-broker. However, using all-broker-caching, all brokers in the system will cache all of the received events. Once a mobile client connects to a broker, the broker will first check its local memory and forward the corresponding events to the client according to the interests of the client. Then, the broker will seek for more interesting events in the network.

A persistent notification protocol [POD04] has used the publisher-caching mechanism to support mobile clients in Pub/Sub systems. In this approach, every broker keeps a list for the IDs of the events published by the broker and buffers the published events. When a mobile client connects a new broker, the client will submit the IDs of latest events received by it to the new broker. The new broker will search the new events for the client in the whole system.

Using previous-broker-caching mechanism, L. Fiege et al. [FIE03] have adapted the Rebeca [MUH02] system to support mobile clients by employing a handover protocol. Using the protocol, a broker buffers the events for a client that has already moved away. Once the mobile client re-connects to the system via another broker,

the system send it the events buffered in the original broker, which can be found according to the subscriptions and event sequence number re-submitted by the client. [TAR03] and [ION04] have used the all-broker-caching mechanism in their pub/sub protocol to support the mobility of subscribers. All of the brokers in the system maintain a log of the event recently received by them. When a mobile client reconnects to the system, the broker connected by the client will scans its log for the event of interest to the mobile client. The cached events that match the interests of the clients need not to be obtained from the previous broker of the client.

To further reduce the message cost and delivery latency, the mobility prediction has been used by Burcea, I., H.-A. Jac, et al, to handle the user mobility [BUR04]. Using the mobile prediction, the brokers can effectively and accurately predict the place where is the destination of a mobile client and then, the broker which is nearest to the destination can pre-fetch the events for the mobile client. Once the mobile client connects to the broker and the broker can directly deliver the pre-fetched events to the mobile client immediately.

After re-connecting to the system, a mobile client wants to obtain all the published events during its movement. Thus, the client will ask the connected broker to get all these events. The broker needs to connect either the event publishers or the previous broker that served the clients to obtain the cached events. Thus, the pub/sub protocol need to a handover operation to deliver the events from the publishers or previous brokers to the broker currently serving the clients. A 2-phase handover protocol has been proposed in [WAN05]. The operations supporting a mobile client include handover request phase and handover response phase. The authors also proposed the approaches to handle concurrent handover and deadlock in handover caused by the mobility of multiply clients.

2.2 Pub/Sub Protocols in Wireless Ad hoc Networks

Recall that we have mentioned the challenging issues in the design of pub/sub protocols for wireless ad hoc networks. However, the priorities of the challenging issues are quite different in the protocol design for different wireless ad hoc networks. The reliability and scalability are the most important issues in the protocol design for MANETs. Similar to MANETs, the pub/sub protocols designed for WMNs need also achieve high reliability and scalability due to the mobility of clients. However, the energy efficiency and scalability is the major concern in designing the pub/sub protocols for WSNs. In DTNs, the reliability and delay are the most important issues, which need to be considered in the design of pub/sub protocol. In this section, we discussed the existing pub/sub protocols designed for MANETs, WSNs, WMNs, and DTNs, respectively.

2.2.1 Pub/Sub Protocols for MANETs

In a MANETs, due to the highly dynamic topology, the events are very easily to be lost in the delivery. Thus, the reliability has the paramount importance in the design of the pub/sub protocols for MANETs. One of the intuitive methods to reduce the event loss is to use redundant event delivery, for example the message flooding.

As the first protocol family to provide pub/sub service to MANETs, FT-CBR and RAFT-CBR [PET05] have used the limited flooding for event delivery in the context of MANETs. FT-CBR allows the publishers and intermediate brokers to broadcast the interesting events to the subscribers' neighbors within a limited number of hops.

Then, the breakage of the paths between clients and brokers or publishers will not lead to the failure of event receiving of the client, since the client can still obtain the interesting event from its neighbors. The RAFT-CBR can provide reliability and fault-tolerance of event delivery in MANETs. The publishers in RAFT-CBR need periodically flood their topic advertisements in the whole network and guarantee that all subscribers can receive the advertisements received by all subscribers. Thus, if there are a limited number of subscribers, the knowledge of all subscribers can be learnt by a publisher via the subscriptions sent by subscribers. With a high connectivity of the network, a publisher can send the events to all the subscribers by using a simple-yet-costly multicast routing protocol designed for MANETs.

The structure-less content-based routing protocol (SL-CBR) [BAL05] also uses the limited flooding to improve the reliability of event delivery. The protocol allows a publisher to forward the events for a subscriber to a group of intermediate brokers but not only one. In addition, the selected brokers are closer to the subscriber than the publisher and thus, they have a higher possibility to successfully deliver the events to the subscriber. Similar to the publisher, the intermediate brokers can autonomously select a group of intermediate broker and forward the received events to them until a subscriber has received the events.

A semi-probabilistic (semi-PROB) method [COS05-1] is to use a probabilistic flooding to improve the reliability of the event delivery in a MANETs. The determination of forwarding an event is depended on the results of matching the event with the subscription. If the matching is successful, the event will be forwarded along the matching link. Otherwise, it will be probabilistically forwarded along the connected links. This method can effectively provide high event delivery and low overhead in highly dynamic scenarios without sacrificing scalability.

As a variant of limited flooding, epidemic algorithms [COS04] were investigated to alleviate the problem of event loss caused by the mobility of nodes. One is subscriber-based pull, which will transmit the gossip messages, some cached events, towards the events subscribers. The other is publisher-based pull, which will forward the gossip messages and the identifying information of the lost events to the events publishers and then the publishers will publish the required events, again.

The approach of redundant event delivery used by the above works can improve the reliability of the event delivery. However, this approach also incurs a high message cost, which may cause the network congestion, the node's battery exhaustion, and thus, the drop of network performance. To avoid the redundancy of event delivery and guarantee the reliability of event delivery, the location-awareness routing has been used to improve the performance of pub/sub protocols. In the location-awareness routing, the events are delivered to the subscribers along the direction of the subscribers' locations. LPS [EUG05] is a location-based Pub/Sub middleware, which allows application entities over a MANET to communication with each other, dependently on their locations. The publish/subscribe operations in LPS are based on not only the matching topics and contents of event against the interests filters, but also the geographical context of the event. The publisher can publish the events to all the subscribers within a publishing range. Also, a subscription is a request to receive events published by producers located within a determined geographical range around the subscriber – the subscription space.

G. Cugola, *et al* [CUG05] also have proposed location-awareness (LA) in pub/sub middleware. Their location-aware pub/sub middleware allows the publishers to publish messages only towards specific areas or the subscribers to subscribe the messages originating in specific areas in a MANET. The subscription and

publication routing employs the location-based routing based on the local location tables maintained by a broker. The entries in a local table are used to record the locations of neighbors and clients.

2.2.2 Pub/Sub Protocols for WSNs

Different from MANETs, WSN is uniquely characterized by the limited energy of the sensor nodes. Thus, the energy-efficiency should be considered with the highest priority in the design of the pub/sub protocols and thus, the pub/sub protocols should reduce the energy consumed by wireless communication as much as possible. However, in a pub/sub system, the publishers need to learn about the interests of the subscribers or the subscribers need to learn about the interesting topics of the events, which will be produced by the publishers. Thus, the pub/sub protocols need to use subscriptions or advertisements flooding in the whole network.

Low-level naming [HEI01] is the first pub/sub middleware based on the subscription flooding. In this architecture, each sensor node is named by the attributes relevant to the applications. When combined with the deployment of sensor nodes, this kind of named data enable in-network processing for data aggregation, collaborative signal processing, and data query. In this architecture, almost all of the data are named by sensory type, geographical location, and sensory value. Similar to the traditional Pub/Sub systems, Low-level naming architecture also adopted filters as the criteria to choose data and provide the filter matching to route the data. The subscription and publication in this architecture are called interest and gradients respectively. The routing protocol for publishing and subscribing is

directed diffusion (DD) [INT00]. At the subscription stage, directed diffusion disseminates the interests from sink node to all the sensor nodes via flooding. As a result, a sensor node perhaps receives one interest several times from different neighbor nodes. When a sensor node receives the interests, it matches the interests with the gradients generated and received. If the matching is successful, the sensor node replies the interest with the gradients reversely along all the links that the interest comes from. In order to decrease the duplicate reply, the sink node and intermediate nodes observes the arrival times of all the replies in a period. After the observation, they choose the paths with most rapid reply as the publishing path. Then, they send out commands to reinforce the path.

Another example to use the subscription flooding is CodeBlue. CodeBlue [MAL04] proposed a very comprehensive and integrated architecture for event service in wireless sensor networks, although it targeted at the wireless sensor networks in emergency medical cares. CodeBlue has been designed to obtain five modules, naming discovery, authentication and encryption, event delivery, filtering and aggregation, and handoff. In addition, CodeBlue is designed over a wide range of network densities, ranging from sparse clinic and hospital deployments to very dense, ad hoc deployments at a mass casualty site. This architecture satisfied almost all the primary functions of event services in wireless sensor networks. In CodeBlue, the subscribers use a flooding to discover the publisher devices in the system and then, use the queries to subscribe the events. The system uses an adaptive demand-drive multicast protocol to deliver the events to multiple subscribers.

[MAR02] also uses the flooding. First, the interval routing protocol (interval-R) [KRA96], where routing table has a low space complexity, is used to establish broker-based pub/sub architecture in WSNs. Every node in this architecture can

individually determine itself to become a broker. After that, every broker will establish a spanning tree rooted at itself by broadcasting its own information in network. The subscribers can choose the most suitable tree to disseminate its subscription. Although the reliability of event transmission is improved, the duplicated establishments of these spanning trees will considerably increase the control overhead.

Mires [SOU05] is designed for an indoor environment-monitoring application over WSNs based on the advertisement flooding. It has been supported by the network-level protocols, including routing and topology control protocols, and provides a high-level APIs that facilitates the development of applications. The provided APIs include Publish/Advertise, Notifier, PublishState, and Send/Receive/Intercept. In Mires, the communication between nodes consists of three steps. First, the nodes collect the types of their on-board sensors and encapsulate the type information into the advertisement with different topics. Next, the advertised topics are transmitted to the sink node by using an ad-hoc routing. Third, the users can subscribe the interesting topics from those collected by sink node reversely along the paths for advertisement dissemination.

Although the subscription flooding or the advertisement flooding allows all nodes to learn about the information of subscribers or publishers, respectively, The flooding is composed of a large amount of wireless communication and is a highly energy-cost operation. Aiming at reducing the amount of wireless communication, some pub/sub protocols use limited flooding to reduce the energy consumption in the flooding.

Semi-probabilistic (semi-PROB) approach [COS05-2] is a kind of limited

flooding, where the number of nodes participating in the flooding has been effectively limited and meanwhile, improve the reliability of event delivery. Usually, on the event delivery path, an intermediate node can keep the wireless link between this node and the next hop one. However, due to the highly dynamics of a WSN, an intermediate node cannot keep such a wireless connection and thus, the intermediate node do not know which node is forwarded the received events to. The semi-probabilistic approach allows a node to forward events to randomly selected neighbor nodes when the node cannot determine the next hop node. This approach allows that a WSN can transmit several copies of an event and then, improves the event delivery ratio.

Different from the flooding in the random selected neighbor nodes used by semi-probabilistic approach, Stenier Data Cache Tree performs the flooding just in the established pub/sub structures. Stenier Data Cache Tree [PRA05] (SDCT) is proposed to use data cache nodes to improve the performance of pub/sub protocols over WSNs. SDCT establishes pub/sub tree in a WSN where the publisher node is the root and subscribing nodes are leaf nodes. The internal-tree nodes are selected from the network to locally cache the data and satisfy the data requirements of the subscribing nodes. When a new subscriber joins the tree, the subscribers will first submit the joining request to the root of the pub/sub. Then, the root will flood the request in the existing pub/sub tree. According to the location of the subscriber, the pub/sub protocol can find the best point where the subscriber can join. Such an incremental extension of the SDCT allows new SDCT to be a near-optimal binary one.

To reduce the energy consumed in pub/sub protocols, the pub/sub protocols aided by geographical information have been proposed. Using the location information of

subscribers, publishers, or events, the flooding can be avoided and thus, the energy can be saved. Diffusion Filters [HEI02] can be viewed as the extension of low-level naming architecture. The extended version provides more expressiveness to gradients, where the gradients can include a list of key-value-operation attribute tuples. In addition, the filter matching and computation inherited the former version. In addition, they assumed that the events in a WSN are closely relevant to a specific geographical region, and thus, they changed the routing protocol of the event service to the geographic and energy-aware routing protocol (GEAR) [YU01]. This improvement helped the whole system to win much higher performance than the old one.

Although the geographical routing can effectively save energy, establishing routing path for every subscriber individually can still incur the high energy consumption. In a WSN, many subscribers may subscribe the same interesting event and these subscribers may be geographically very near to each other. Thus, the subscribers can share the delivery paths and the amount of the event delivery can be reduced. The shortest delivery path (SDP), the shortest delivery path with lower delivery overhead (SDP_LDO), and lowest delivery overhead (LDO) routing protocols [ZHE07-3] are proposed to support multiple subscribers and publishers. Using the geographical information, the three protocols allow the subscribers to share the event delivery paths and thus, reduce the amount of wireless transmission to save the energy of the networks. Further more, to avoid the control message transmission among subscribers on establishing delivery path, the three protocols need not any coordinate among the subscribers. Thus, a large amount of message transmission has been reduced and the energy has been saved. In addition, due to the different application requirements on event subscription and delivery, the three

protocols can be used to different application scenarios with different performances and protocol costs.

In the wireless communication in WSNs, the channel competition can consume a large amount of energy. An effective channel assignment mechanism can significantly reduce the energy consumption in WSNs. Thus, the event transmission scheduling in WSNs is also one of the important methods to save the energy, where the elaborate wireless transmission scheduling can effectively avoid the channel contentions among sensor nodes and allow the sensor node in the sleeping status to save energy.

U. Centitenmal, A. Flinder, and Y. Sun [CEN03] presented a power-efficient data dissemination (PE-DD) in WSNs based on transmission scheduling. The new event-based communication model just aimed to the energy-constrained devices in sensor networks. In this model, the network was arranged as an event dissemination tree, with nodes subscribing to the event types they were interested in. An event scheduler dynamically allocated and multiplexed upstream and downstream time slots for each event type. Power consumption among wireless nodes was reduced by allowing each node to power down its radio during the portions of the schedule that do not match its particular event subscription. The event dissemination schedule could be determined in both a centralized and distributed fashion, and was highly adaptive to the changing rates at which events were generated and distributed in the network.

2.2.3 Pub/Sub Protocols for WMNs

In a WMN, the clients can move around in the networks and the wireless connection between a base station and a client cannot exist in a long time. After a client moves out of the range of a base station, the base station cannot directly deliver the received events to the client. Thus, the reliability becomes the most important performance target in the design of the pub/sub protocol for WMNs. Mobility prediction is one of the method to improve the reliability with lower cost. Using this method, the clients proactively report their locations or mobility parameters when a certain time is expired or the mobility parameters of the clients are changed. Then, the base stations can estimate the location of clients and deliver the received event to the estimated locations of the clients.

There are few pub/sub protocols designed for WMNs. The first pub/sub protocol specifically designed for wireless mesh networks is proposed in [ZHE07-2]. In WMNs, a client can probably move out of the range of the original serving base stations and consequently, the subscribed events cannot quickly deliver to this client reliably. Aiming at reducing the number of re-transmission of the events and costs for the paging of the client, the work proposed to the mobility prediction to estimate the locations of the client and the events can be delivered to the client more reliably and quickly. In this work, the mobile client is assumed to move by following a random way-point model. The mobile client will send its new mobility parameters to the original serving base station every time when the mobile client needs to stop and change its mobility status. Then, the original base station can use the received parameters to compute the approximate current location of the client and accurately deliver the received event to the client.

Usually, the mobile users just use hand-held communication devices, such as mobile phone, notebook, etc, to communication with pub/sub systems. The devices

need continuously monitor the locations and mobility status of themselves. Once the locations or the mobility status has changed, the devices need to report the new information. Both the monitoring and the reporting may easily exhaust the mobile client and then, cause the failure of event receiving of mobile users.

2.2.4 Pub/Sub Protocols for DTNs

In DTNs, there are no continuously connected data delivery paths in DTNs. The data will be opportunistically transmitted by the mobile nodes in a carry-and-forward manner. Thus, the most important considerations on the design of pub/sub protocols for DTNs are reliability and delay. The partitioned network and the opportunistic forwarding may cause data cached in a node in a very long time until the data are expired and dropped. Thus, a large amount of data cannot be successfully transmitted to the destination nodes. To avoid the data delivery failure, one approach is to establish a stable and connected overlay on the physical underlying DTNs. Then, the pub/sub protocols can run on the overlay and then, the reliability and latency of event delivery can be improved. At present, there are two works on the design of pub/sub protocols for DTNs based on overlay networks.

One of the works, namely SAFIRE [AHM07], is a system designed for disaster salvage. In SAFIRE, the communication devices or terminals attached on the responders and deployed in the mobile hospitals and temporary command centers are organized into a DTN. Such a DTN can help different functional units to exchange information in the disaster salvage. To improve the reliability and delivery delay of the pub/sub system, SAFIRE establishes an overlay network using DHT

based on the underlying DTNs. The overlay network uses a link-state routing protocol to select the most reliable data delivery path. In addition, according to the requirement of applications, the overlay network assigns different priorities to different data packets and prefers delivering the packets with higher priority to the ones with lower priority. Thus, the reliability and delivery delay in the overlay network can be largely improved. Taking advantages of the overlay network, a pub/sub system built on it can deliver events reliably and efficiently.

Another work [YON07] studies information exchange in a social network, which is referred to as a sparse MANETs composed of the computation and wireless communication devices obtained by people. In this work, a social-aware (SA) community-based overlay is built for DTNs. The nodes in the same community tend to contact each other more often and for longer durations. Using this observation and taking advantages of reliable and low-latency communications within a community, the event delivery from publishers to subscribers takes a community-oriented approach and, thus, can be done in a more reliable, timely, and efficient manner. The establishment and maintenance of an overlay network incur highly cost, especially over a large-scale DTN. Our work reported in the thesis does not need to establish any overlay network and, meanwhile, can guarantee the delivery ratio specified by the applications.

The overlay networks with high performance can effectively guarantee the performance of event delivery based on them. However, the establishment and maintenance of the overlay networks in DTNs is not trivial. Due to the unreliable communication in DTNs, the transmission of the control messages for overlay networks may be highly cost and the performance of overlay networks can be affected by the unreliable message transmission. Consequently, the performance of

pub/sub protocol running on the overlay networks can be largely degraded.

2.3 Summary

In this chapter, we first review the existing pub/sub protocols designed for traditional wireless networks and show that the existing protocols is not suitable to the wireless ad hoc networks because the wireless ad hoc networks have not wired infrastructures used in traditional wireless networks. Then, we review the existing pub/sub protocols specifically designed for wireless ad hoc networks, including including MANETs, WSNs, WMNs, and DTNs, respectively. We also discuss the disadvantages of the existing works and show that the flooding-based message dissemination is most important factor to degrade the performance of the existing protocols. In next chapter, based on the literature review, we summarize a 3-D framework for designing pub/sub protocols in wireless ad hoc network and show the summary of existing works according to the proposed 3-D design framework.

Chapter 3

Design Framework of Pub/Sub Protocols for Wireless ad hoc Networks

In this chapter, we propose a framework for designing pub/sub protocols in wireless ad hoc networks. The framework addresses the design issues in different dimensions and provides the space for the design of pub/sub protocols in wireless ad hoc networks. It also lays the foundation of and motivates the research in the remaining part of the thesis.

This chapter is organized as follows. Section 3.1 describes the framework for pub/sub protocol design in wireless ad hoc networks. Based on the proposed design framework, section 3.2 discusses how geographic information can be used to design the pub/sub protocols and identifies the problems addressed by our work.

3.1 A Framework for Designing Pub/Sub Protocols in Wireless ad hoc Networks

Based on the observation in the design of the existing protocols discussed in the last chapter, we classify the issues of the design of pub/sub protocols into three dimensions, namely pub/sub initialization, message dissemination, and broker selection.

The strategies for pub/sub initialization allow the subscribers and publishers to

learn about the events and the interests from the publishers and the subscribers, respectively. In the existing works, the most important pub/sub initialization strategies include advertisement-driven (AD), subscription-driven (SD), and hybrid (HD) ones. Using the AD strategy, the publishers will disseminate the topics of the events to the networks and then the interested nodes can subscribe the corresponding events according to the advertisements. The SD strategy allows the nodes to disseminate the subscriptions to the network and then the publishers and brokers can generate and deliver the interesting events to the corresponding subscribers. Using the HD strategy, publishers and subscribers can send the advertisements and subscriptions, respectively. Then, the brokers in the networks can obtain all of the information about the publishers and subscribers.

The second dimension is the strategies for message dissemination. The strategies are used to send the advertisements, subscriptions, and events to the corresponding subscribers, publishers, and brokers. Because the subscribers and the publishers are decoupled from each other in pub/sub protocols, it is quite difficult to establish the message dissemination paths between the publishers and subscribers. Existing protocols usually use flooding (FL) and limited flooding (LF) to disseminate the messages in the network. The flooding strategy broadcasts the messages to all the nodes, while the limited flooding strategy broadcast the messages just to partial nodes in the networks. Although flooding or limited flooding can effectively improve the reliability of the pub/sub protocols, the resource consumption is very high. However, using geographic information, pub/sub protocols can significantly reduce the resource consumption and meanwhile, guarantee the reliability of the protocols. Thus, the geographical information has been used to aid the message dissemination. According to the physical locations of the target nodes or the events,

the pub/sub protocols can use geography-aid (GE) strategy to forward messages in a more efficient manner.

The third dimension is the strategies for broker selection, which are used to select the suitable nodes to serve as brokers. The strategies used in existing protocols include one node (ON), selective nodes (SN), and all nodes (AN), meaning that, the pub/sub protocols have one node, partial nodes, or all the nodes as brokers. As for ON, the gateway or the sink node serves as the only broker in the network, where all the publishers send the events and all the subscribers send the subscription to this node. Using the SN strategy, a pub/sub protocol can also elaborately select a part of nodes to be the brokers, such as the cluster heads in MANETs or WSNs, and the base stations in WMNs. Finally, using the AN strategy, pub/sub protocols can employ all the nodes to be brokers. Then, any node can handle the subscription dissemination and event delivery.

Based on the three dimensions of the major design issues, we propose a framework for designing the pub/sub protocol (see Fig. 3.1). Using the framework, a pub/sub protocol can be designed by combining the strategies selected from the three dimensions. In practice, the pub/sub protocols using the combinations of different strategies can be applied in different situations and/or achieve different levels of performance. Thus, according to the specific network types and performance requirements of the applications, we can select different strategies from the dimensions in the framework to design the corresponding pub/sub protocols. Next, we discuss the feasible and popular combinations of the strategies. Some of the combinations have been used in existing works. Some are identified as new design strategies and addressed by this thesis. Detailed discussion will be provided in Section 3.2.

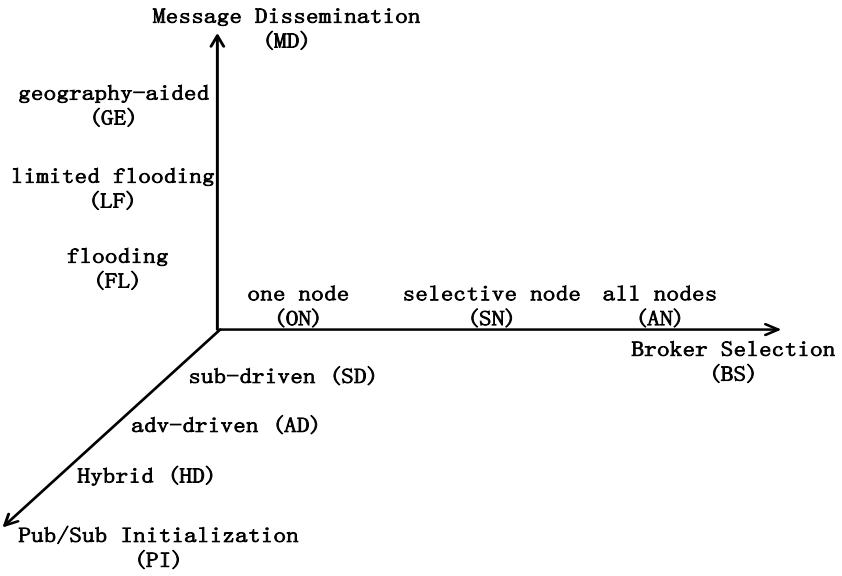


Fig. 3.1 The design framework of pub/sub protocols in wireless ad hoc networks

SD-ON-FL and AD-ON-FL are flooding-based pub/sub protocols with only one broker and using subscription-driven and advertisement-driven strategies, respectively. These protocols can be used in a small-scale network, where the nodes are just subscribers or publishers and the gateway or sink node of the network is the broker. They can significantly improve the reliability of event delivery and reduce the control overhead of pub/sub protocols. In WSNs, Low-level Naming [HEI01], Interval-R[MAR02], and PE-DD [CEN03] use SD-ON-FL, where all the sensor nodes are publishers and just send the events to the sink node. The applications based on the WSN just send their subscriptions to the sink node. The broker, i.e. the sink node, is responsible for filtering and disseminating the events from the sensor nodes.

AD-AN-FL and SD-AN-FL are flooding-based pub/sub protocols with all nodes as brokers and using advertisement-driven and subscription-driven strategies, respectively. AD-AN-FL can be used in small-scale wireless ad hoc networks, where the number of publishers is much less than the number of subscribers and all nodes are brokers. In such a network, the brokers flood the advertisements to all the nodes.

SD-AN-FL can be used in small-scale wireless ad hoc networks where the major part of the nodes is publishers and all nodes are brokers. The subscribers just flood the subscriptions in the whole network. Since all nodes are brokers, these brokers can learn about the subscriptions or advertisements and help deliver the interesting events to the corresponding subscribers. Thus, AD-AN-FL and SD-AN-FL can effectively enhance the reliability of the pub/sub protocols, especially in highly dynamic or frequently partitioned wireless ad hoc networks. The SL-CBR protocol proposed in [BAL05] uses HD-AN-FL, the hybrid of AD-AN-FL and SD-AN-FL, in the MANETs with small number of subscribers and publishers.

AD-AN-LF and SD-AN-LF are limited-flooding-based pub/sub protocols with all nodes as brokers and using advertisement-driven and subscription-driven strategies, respectively. The LF strategy can also guarantee the reliability of the event delivery and meanwhile, effectively reduce the message cost in comparison with the flooding strategy. Thus, AD-AN-LF and SD-AN-LF can be used in middle-scale wireless ad hoc networks, where the number of nodes is not too large. Semi-PROB proposed in [COS05-1] and [COS05-2] have used SD-AN-LF in middlew-scale MANETs and WSNs, respectively.

SD-SN-FL and AD-SN-FL are the flooding-based pub/sub protocols with selective nodes as brokers and using subscription-driven and advertisement-driven, respectively. The SN strategy can avoid all of the nodes involved in the subscription dissemination, event filtering, and event delivery. Thus, the protocols using SN have higher scalability and cause lower resource consumption. In addition, the selected nodes usually have high node degree, high stability, sufficient energy, or high performance in computing and wireless communication. Thus, these selected nodes can serve as brokers more effectively, efficiently and reliably. SD-SN-FL and

AD-SN-FL can be used in the small-scale wireless ad hoc networks where the number of subscribers and publishers is very small, respectively. HD-SN-FL, the hybrid of SD-SN-FL and AD-SN-FL, can be used in the networks where the numbers of both subscribers and publishers are small. Epidemic [COS04] and CodeBlue [MAL04] use the SD-SN-FL and AD-SN-FL in MANETs and WSNs, respectively. Mires [SOU05] use HD-SN-FL in WSNs.

SD-SN-LF and AD-SN-LF are the limited-flooding-based pub/sub protocols with selective nodes as brokers and using subscription-driven and advertisement-driven respectively. As we have mentioned, the limited-flooding (LF) can guarantee the reliability of pub/sub protocols and meanwhile, effectively reduce the resource consumption in comparison with flooding. Thus, SD-SN-LF and AD-SN-LF can be used in wireless ad hoc networks with larger scales. FT-CBR[PET05] and SDCT [PRA05] use SD-SN-LF in MANETs and WSNs, respectively. SA[YON07] and SAFIRE[AHM07] use SD-SN-LF in DTNs.

SD-SN-GE and AD-SN-GE are the geography-aided pub/sub protocols with selective nodes as brokers and using subscription-driven and advertisement-driven, respectively. Using the GE strategy in message dissemination, the pub/sub protocols can effectively reduce the resource consumption caused by flooding and limited-flooding. Thus, SD-SN-GE and AD-SN-GE can be used in large-scale wireless ad hoc networks, where the number of subscribers and publishers are small, respectively. LPS [EUG05] and LA [CUG05] use SD-SN-GE in large-scale MANETs. Another kind of geography-aided pub/sub protocols is SD-ON-GE, where all nodes are subscribers or publishers and the gateway node or the sink node is the broker. The nodes in wireless ad hoc networks just send their subscriptions or publications to the broker. DF [HEI02] uses SD-ON-GE in WSNs.

As we have mentioned previously, the SN strategy can also improve the scalability and efficiency of the protocols. Thus, we can use the combination of GE and SN strategies in large-scale WSNs, WMNs, and DTNs, for significantly reducing the network resource consumptions and improving the scalability and efficiency of the protocols. In practice, a pub/sub protocol can use the geographic information in different manners, which can be used to satisfy the different performance requirements of the applications.

Using the proposed framework of the pub/sub protocols, we summarize the strategies used in the existing protocols for wireless ad hoc networks in Table 3.1.

Table 3.1 The combinations of strategies used in the existing pub/sub protocols in Wireless ad hoc networks

Protocol	Network	PI			BS			MD		
		SD	AD	HD	ON	SN	AN	FL	LF	GE
FT-CBR	MANETs	√				√			√	
RAFT-CBR	MANETs	√				√			√	
SL-CBR	MANETs			√			√	√		
Semi-PROB	MANETs	√					√		√	
Epidemic	MANETs	√				√			√	
LPS	MANETs	√				√				√
LA	MANETs	√				√				√
Low-level Naming	WSNs	√			√			√		
CodeBlue	WSNs		√			√		√		
Interval-R	WSNs	√			√			√		
Mires	WSNs			√		√		√		
Semi-PROB	WSNs	√					√		√	
SDCT	WSNs	√				√			√	
DF	WSNs	√			√					√
SDP, SDP_LDO, LDO	WSNs	√				√				√
PE-DD	WSNs	√			√			√		
SA	DTNs	√				√			√	
SAFIRE	DTNs	√				√			√	

3.2 Geographic Information Used in Pub/Sub Protocol Design

In this thesis, we focus on how to use geographic information to address different performance challenges in WSNs, WMNs, and DTNs, as discussed in Chapter 1 and Chapter 2. Most existing pub/sub protocols designed for wireless ad hoc networks just use the flooding or limited-flooding strategies to disseminate messages, which can cause very high delivery cost and therefore, lead to high resource consumption and poor scalability. Geographic information can be used to aid the message dissemination, avoid flooding, reduce resource consumption, and improve scalability accordingly. With the development of the location technologies, a node in a wireless ad hoc network can easily obtain its location. In the outdoor environments, many location systems have been developed. For example, a node moving in a city can use the global positioning system (GPS) [JON95] to obtain its location. Another example is the positioning system based on 3G communication system. In addition, many protocols [NIC04][HU04][HE03] have proposed to solve the locating problem in traditional wireless networks and wireless ad hoc networks. In indoor environments, systems [HIG01] also have been developed for node positioning. For example, Cricket system [PRI00] developed by MIT can help nodes learn about their indoor positions. In practice, the cost of such a device or a system for the indoor or outdoor positioning become lower and lower.

Geographic information in a pub/sub system usually includes the location information of publishers, subscribers, brokers, and events. In addition, in DTNs, there is a very special kind of nodes, namely ferries, which also has the geographic

information for their locations. In addition, the geographic information can also be categorized into the mobility and static geographic information. In fact, the mobility information can be derived from the location information changing along with the time.

In wireless ad hoc networks, pub/sub protocols can use geographic information in four different ways (see Fig. 3.2), including geographic forwarding (GE/GF), geographic forwarding with event delivery path sharing (GE/GFPS), location prediction of mobile subscribers (GE/LP), and mobility prediction of brokers and ferries (GE/MP). Pub/Sub protocols can use these different methods to satisfy the different performance requirements in different networks. We have proposed three pub/sub protocols, namely HESPER, PUMA, and GIANT, which use SD-SN-GE/GF-PS, SD-SN-GE/LP, and AD-SN-GE/MP for WSNs, WMNs, and DTNs, respectively. Next, we discuss how the four different ways of using geographic information are employed in our protocols.

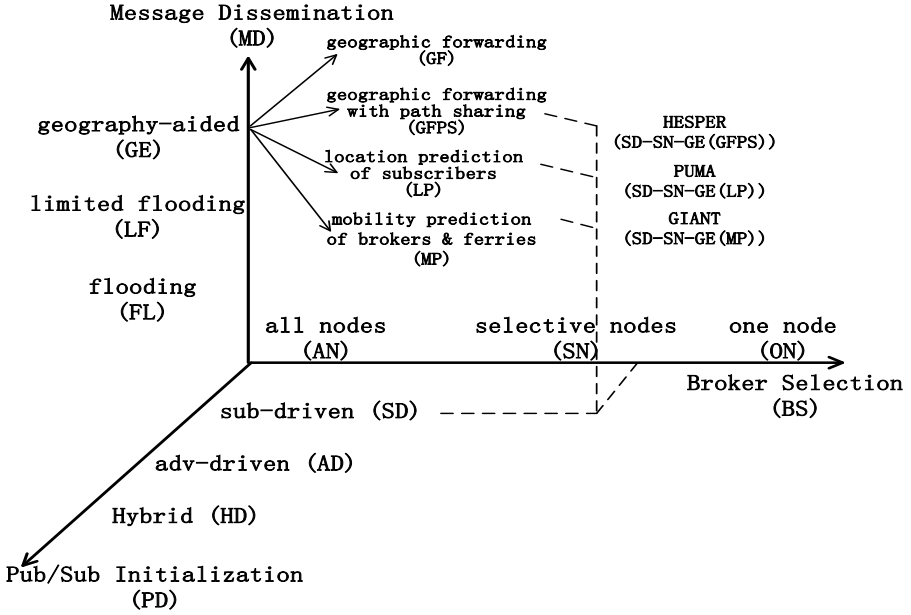


Fig. 3.2 Geographic information used in the design of pub/sub protocols

- *Geographic forwarding* (GE/GF). In a pub/sub protocol for a wireless ad hoc network, the location of an event or a publisher can be a very important property because some users of a pub/sub protocol are just interested in the events generated by the publishers at a specific location [EUG05]. For example, the drivers are interested in the traffic conditions of a junction or a main highway. Another example is that the soldiers are very interested in the situations at a specific tactical spot. According to the location of the events, pub/sub protocols can use geographic forwarding, for example GEAR[YU01], to establish the event delivery paths from publishers to subscribers. Thus, the protocols can avoid flooding and meanwhile, the performance can be largely improved.

- *Geographic forwarding with path sharing* (GE/GFPS). The subscribers may be interested in the same events. For example, all drivers are interested in the traffic condition in the main highway or road during rush hours. Another example is that different soldiers want to obtain the events from the same important tactical spot. Thus, the pub/sub protocols can use the geographic information of both events and subscribers for geographic forwarding and paths sharing among different subscribers. Using GFPS, the pub/sub protocols can effectively avoid the flooding and meanwhile, redundant event delivery. Thus, the performance of pub/sub protocol can be largely improved. In this thesis, HESPER has used the GE/GFPS strategy to disseminate events in the large-scale WSNs.

- *Location prediction* (GE/LP). The subscribers in a wireless ad hoc network can be mobile. In this case, the brokers connected with the mobile subscribers previously cannot directly forward event to the subscribers that have moved away. However, the broker can use location prediction based on the collected mobility information of the subscribers to predict the locations of the subscribers

and then, deliver the events to the predicted locations. Such a method can also avoid flooding of events and improve the performance of the pub/sub protocols. In this thesis, PUMA has used the LP strategy to deliver events to mobile subscribers in WMNs.

- *Mobility prediction* (GE/MP). In wireless ad hoc networks, the brokers or the message ferries may also move around. Thus, the pub/sub protocols can use the mobility predict to determine the locations where the brokers or the ferries will probably reach. Then, the protocols can recruit the suitable brokers and ferries to carry and forward the events to the next-hop broker or the subscribers, when the mobile brokers and ferries reach the corresponding locations. Such a manner can effectively avoid the repeated and redundant event delivery and thus, improve the performance of the pub/sub protocol. In this thesis, GIANT has used the MP strategy in DTNs.

Chapter 4

HESPER: A Highly Efficient and Scalable Pub/Sub Protocol in WSNs

In this chapter, we describe the proposed Highly Efficient and Scalable Publish/subscribe protocol for wireless sEnsoR networks, namely *HESPER*.. This chapter is organized as follows. Section 4.1 briefly introduces the work. Section 4.2 discusses the system model. Section 4.3 describes the HESPER protocol and section 4.4 presents the theoretical analysis and experimental evaluation of the HESPER protocol. Finally, Section 4.5 summarizes this chapter.

4.1 Overview

A wireless sensor network (WSN) is composed of sensor nodes attached with sensing, computation, and wireless communication devices. These sensor nodes cooperatively collect sensory data and transmit them to the applications over WSNs [AKY02]. In the recent years, WSNs have attracted attentions from both industrial and academic communities. Many WSN-based applications have been exploited in different areas, such as environment monitoring, traffic management, intelligent building management, battle-field surveillance, etc.

To meet the requirements of such applications, an important task of WSNs is to provide users event service, which can support asynchronous data exchange to users

upon the specific data interests of users. Since a WSN is uniquely characterized by the large scale and the limited energy, design and implementation of event service for WSNs face the challenges of high scalability and energy efficiency. Usually, a WSN is composed of hundreds or thousands of sensor nodes. The collection and dissemination of the global information are not available in WSNs. Thus, to provide highly scalable event service, a WSN should employ the distributed and localized mechanisms. In addition, because a sensor node is powered by the only on-board battery, the available energy of a node is very limited. It is well known that the major partial of a sensor node is consumed by wireless communication [PRA05]. Thus, to save the energy and avoid node exhaustive in a short time, the wireless communication in event services should be reduced as much as possible.

Publish/Subscribe (Pub/Sub) [BAN99][CAR00][EUG03] system has been widely used to support event service in the traditional distributed computing systems. A pub/sub system can encapsulate data into events and exchange events for applications. In a pub/sub system, the event generators and consumers are named publishers and subscribers, respectively, where the publishers use the operation of *Publish* to send events and subscribers use the operation of *Subscribe* to receive events. Usually, the publishers and subscribers will use pub/sub protocols to disseminate and subscribe the interesting events, respectively. However, most previous Pub/Sub protocols for WSNs employ subscription flooding in the whole network [HEI01][INT00][MAR02][COS05-2][SOU05] or in the existing pub/sub trees[PRA05], which suffers from poor scalability and incurs high energy costs. To avoid subscription flooding, D. Estrin, *at el*, proposed Diffusion Filters (DF) [HEI02], which employs geographic energy-aware routing (GEAR)[YU01] to disseminate event subscriptions. However, DF establishes independent event

delivery paths for different subscribers, where the events are delivered replicated and consequently, cause high energy consumption.

We address how to design an efficient and scalable pub/sub protocol for WSNs. Given a publisher and multiple subscribers, we want to establish a tree that is accommodating the publisher and subscribers and minimizes the cost of wireless event transmission from the publisher to the subscribers. Such a problem has been adequately expressed as a Stenier Minimal Tree problem, where the event delivery paths from a publisher to all subscribers are organized into a tree and the event delivery in this tree needs minimum cost. However, in a pub/sub tree, the messages transmitted include not only the events, but also pub/sub control messages, and the messages for tree establishment. We want to reduce the total amount of the wireless transmission for such messages in order to save energy. In addition, we want to use only localized topology information and algorithms to achieve high scalability.

In this chapter, we propose *HESPER*, a Highly Efficient and Scalable Pub/sub protocol in wireless sEnsoR networks, which HESPER outperforms the previous pub/sub protocols in terms of scalability and energy-efficiency by taking the following advantages. First, in HESPER, using the geographical information of the interesting events, the subscribers can efficiently subscribe the events without flooding. Second, HESPER can support event pub/sub for multiple subscribers and publishers in a distributed manner. According to the locations of the events and publishers, all subscribers use the same calculation to obtain the subscription dissemination paths. Thus, HESPER can establish the event delivery paths that greatly overlap with each other and are shared by the subscribers in event delivery. Since HESPER does not need the coordination among the subscribers to establish the delivery paths, it can significantly reduce the amount of event delivery and thus,

save energy. Third, HESPER uses the intermediate nodes rotation to balance the working loads for the intermediate nodes on the delivery paths. This can avoid the rapid exhaustions of the nodes on the delivery paths and the consequent delivery disruptions. Fourth, HESPER offers two different routing rules to balance the tradeoff between the subscription and publication costs to satisfy the different requirements of the practical applications.

We have conducted the theoretical analysis and the extensive experimental evaluation to show the performance advantages of HESPER in compared with the previous solutions, including DF and SDCT. Both of them are the famous pub/sub protocols in WSNs. Our extensive simulation results show that, in comparison with the previous work of DF and SDCT, HESPER can save event delivery energy up to 50% and 80% respectively, and meanwhile, save the event subscription energy up to 50% and 90%, respectively. In addition, our theoretical analysis shows that, measured by hop count, the length of delivery path established by HESPER between a publisher node and a subscriber node is at most 0.8 time larger than the theoretical shortest path. The experimental results show the average length of the delivery path in HESPER is less than l.5 times of the delivery path established by DF. Considering the significant energy saving in HESPER, such a little bit larger delivery latency of HESPER is still quite reasonable and attractive.

4.2 System Models and Basic Operations

In this section, we discuss the system model and the assumptions used in the rest of the chapter. We assume that all sensor nodes are randomly deployed in a

rectangular region and are aware of their own geographical locations. All sensor nodes share the same transmission range (*TR*) and any two nodes with the distance shorter than *TR* can directly communicate with each other. According to the transmission range, the region is divided into the equal-size grids, where the nodes in one grid can directly communicate with each other. Thus, the maximum length of the diagonal lines in the grid is TR and the maximum side length of a grid is about 0.7**TR*.

Now, we discuss the definitions of the event and subscription used by HESPER. An event is described by a 3-tuple as follows.

Event = <*Region*, *Topic*, *Value*>, where

- Region is the source region of the event, denoted by the publisher's grid.
- Topic is the sensory type described by the event, such as temperature, light, pressure, etc.
- Value is the sensory result, such as 30 degree, 1 lumen, 1 Pa, etc.

A subscription can be described by a 4-tuple <*Region, Topic, Value, Operation*>. If an event satisfies a subscription, the region and the topic of the events are the same as the satisfied subscription, and the relation between the values of the event and the subscription satisfies the 'operation' in the subscription. For an example, event <A, temperature, 5 > can satisfy the subscription <A, temperature, 10, '<'>. Thus, a subscription will be used as a filter to find the interesting events. In addition, if a subscription is covered by another subscription, the set of all events satisfying the first subscription is a subset of the all events satisfying the second subscription. For example, subscription A <A, temperature, 10, '>'> can be covered by

subscription B <A, temperature, 5, '>'>. We note subscription B $\supseteq$ subscription A.

Next, we describe the model used for designing our pub/sub protocol. We assume that HESPER uses broker/client architecture, where the brokers are responsible for the subscription dissemination and event delivery and the clients are the event subscribers or publishers. As for one topic of subscriptions and events, HESPER choose only one broker in a grid. The mechanism on how to use an event topic to select brokers will be discussed in the following section. In the remaining part of the chapter, we refer to a broker connected with a publisher as a publisher-hosting broker (PHB) and a broker connected with a subscriber as a subscriber-hosting broker (SHB).

Now, we discuss how to select a broker from the nodes in one grid. A broker in a pub/sub protocol is responsible for the subscription dissemination and event delivery. Such a high communication load will exhaust the broker nodes very quickly. Thus, we need to elaborately select the broker node and balance the traffic load on different sensor nodes. In HESPER, we use a hash function H to determine the broker nodes. When a node N in the gird G receives a subscription S, it will use the hash function to calculate a physical location (x, y) in the grid G as follows:

$$x \leftarrow H^i(S.Topic), \text{ and } y \leftarrow H^j(S.Topic),$$

where i and j are different positive integers and mean different times of the hash function being used. Then, node N selects the node from grid G, which is the nearest one to (x, y). Using such a hash function, a node can select different broker nodes for the events of different topics. Thus, HESPER can effectively balance the event delivery traffic loads among the nodes in a grid.

However, the selected broker nodes are usually consumed more energy to transmit

the subscriptions and events. Thus, the broker nodes are exhausted more quickly. In HESPER, we use the rotation of brokers to avoid the occurrence of the exhaustive node, where an existing broker with low remaining energy can select the new brokers with high remaining energy to perform the functions of the event delivery. The rotation of brokers has the following steps, as shown in Fig. 4.1.

1. If the remaining energy of a broker *B* is lower than a threshold value, Broker *B* will start a rotation of broker.
2. Broker *B* will use the hash function to determine the positions of new brokers corresponding to every event topic, which is responsible by Broker *B*.
3. Broker *B* determines the nodes to be the new brokers, which is in the same grid of Broker *B* and meanwhile, the nearest one to the positions calculated by the hash function.
4. Broker *B* delivers the local data structures to the new brokers according to their event topics.
5. Broker *B* sends the notifications of the broker updating to the last-hop brokers and the last-hop brokers will update the local data structures accordingly.

Fig. 4.1 the steps of the broker rotation

In HESPER, we use subscription-reverse-path approach to establish event delivery paths. Using this approach, a subscriber will first send out a subscription. Then, the subscription will be disseminated in the network until the corresponding PHB or a suitable intermediate broker receives the subscription. Next, the corresponding PHB or the intermediate broker will deliver the events to the subscribers reversely along the subscription dissemination path.

4.3 The HESPER Protocol

In this section, we describe HESPER, which uses two strategies, including

HESPER-I and HESPER-II. First, we describe the preliminary knowledge about our protocol design; next, we describe the two different proposed routing strategies. In the designing of HESPER, we aim at enabling different subscribers to share the event delivery paths as much as possible. In this way, the event delivery on the common part of the paths can serve all of the subscribers simultaneously and, therefore, the amount of event delivery can be reduced. Our protocol adopts the subscription-reverse-path mechanism. Thus, the paths for subscription dissemination determine the final event delivery paths and the SHBs should use overlapping paths as much as possible to propagate their subscriptions. According to the geographic information of the desired event publisher, the newly joining SHB can effectively discover the existing propagation path, which may already become a delivery path. Then, the SHB can take full advantages of the existing path by appending itself to the existing path. We discuss the details in this section.

For the simplicity of the discussion, we first establish a rectangular coordinate system. Assuming that a PHB is located at the center of its grid, we set the PHB's location to be the origin and add four lines passing the origin point. One of the four lines is a vertical line, and then every acute angle generated by two neighboring lines is 45 degrees, as shown in Fig.4.2. Starting from the vertical line, we name the four lines as X-Axis, leading-diagonal (L-diagonal), Y-Axis, and auxiliary-diagonal (A-diagonal), in the clockwise direction (Fig. 4.2). The grids passed by axes and diagonals are named axis and diagonal grids, respectively. The two axes together with the two diagonals divide the whole region into 8 parts, named as Part I to Part VIII in the clockwise direction, excluding the grids passed by the axis and diagonal lines. According to the partition of the region, we have the following notations for the brokers in different regions.

- B_X: the set of brokers that are in the grids passed by X-Axis. Similarly, we have B_Y.
- B_L: the set of brokers that are in the grids passed by L-diagonal. Similarly, we have B_A.
- B_I: the set of brokers that are in Region I. Similarly, we also have B_{II}, B_{III}, B_{IV}, B_V, B_{VI}, B_{VII}, *and* B_{VIII}.

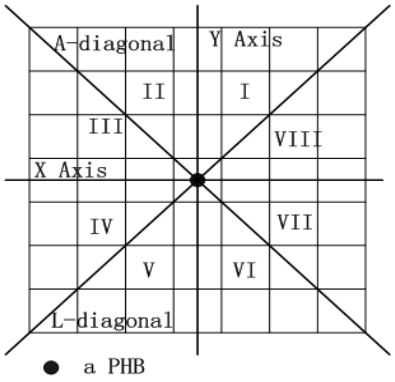


Fig.4.2 8 parts of a deploying region

We also define the following directions.

- X-direction: the direction parallel to X-Axis and towards the target PHB.
- Y-direction: the direction parallel to Y-Axis and towards the target PHB.
- L-direction: the direction parallel to L-diagonal and towards the target PHB
- A-direction: the direction parallel to A-diagonal and towards the target PHB

Accordingly, we give the following functions to obtain the neighboring brokers in different directions.

- $B^X( )$: this function is used to return the neighboring brokers in X-direction. Similarly, we have $B^Y( )$, $B^L( )$, and $B^A( )$, respectively.

Recall that the event delivery paths used in HESPER are the reverse paths of subscription disseminations. Thus, the establishment of subscription dissemination paths will determine the performance of event delivery in HESPER. Now, we discuss how to select a grid to be the one on the next hop during the subscription dissemination. To achieve high energy efficiency in the event delivery for one subscriber, one of the intuitive approach is to establish a shortest path between a subscription and a publisher, where the event will be delivered in the smallest number of times. However, it is obviously not an energy-efficient approach to establish a shortest path for every subscriber to a publisher, if there are multiple subscribers. This is because some subscribers staying very near can share event delivery paths. An optimal solution to establish the event delivery paths for multiple subscribers is to use a centralized algorithm to establish an event delivery tree that is rooted at the publisher and contains all subscribers. However, such an approach needs a large amount of subscribers' information transmission for the tree establishment. This will cause high energy consumption. Both of the above approaches are not suitable to WSNs.

In HESPER, we consider both path sharing and distributed mechanisms. To increase the possibility of different subscribers sharing the event delivery paths, HESPER makes the subscribers to disseminate the subscriptions into the same grid as early as possible, when the subscriptions have been disseminated in the WSN. Thus, the subscribers can share the delivery path from the publisher to the grid where the subscriptions have been disseminated. However, the subscribers have no knowledge about each other. In HESPER, all subscriber and intermediate nodes can use the same routing rule to calculate the subscription dissemination paths according to the physical locations of the publisher. Using such a routing rule, a subscriber can

calculate the part of the event delivery path that will be used by other subscribers. For example, considering the deployment region is a rectangle in this chapter, the sharing part of the path is in the directions parallelizing with the diagonals. Thus, the subscribers can be guaranteed to have a high possibility of sharing the event delivery paths and meanwhile, need not the information exchange among different subscribers. The details of the routing rules used in HESPER-I and II have been discussed as follows, respectively.

Using HESPER-I, a SHB greedily propagates a subscription towards the target PHB to establish the shortest subscription delivery path that is the reverse shortest event delivery path. First, the SHB and the intermediate brokers greedily forward the subscriptions to the diagonal grids along the X-direction or Y-direction. Then, the brokers in the diagonal grids continue to forward the received subscriptions along the diagonal towards the PHB. Obviously, the delivery path on the diagonal line can be shared by different SHBs and the redundant event delivery can be reduced. Fig. 4.3(a) and (b) show the examples of subscription and event delivery paths established by HESPER-I, respectively.

When a broker receives a new subscription *S*, it first checks its routing table. If an entry in the routing table can satisfy *S* or the broker is the target PHB, the broker send the reply back to the sender of *S*; otherwise, the broker employs Algorithm 4.1, shown in Fig. 4.4, to determine the next-hop broker and then, forward *S* to the selected next-hop broker.

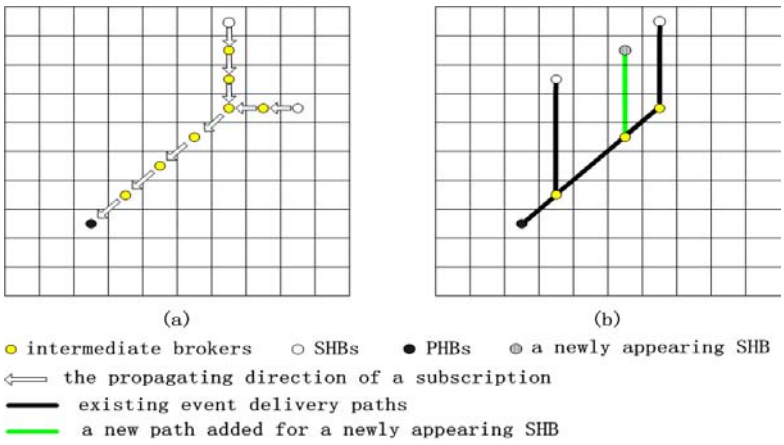


Fig. 4.3 examples of the subscription and event delivery paths established by HESPER-I

Algorithm 4.1: HESPER-I
Input::
br //this broker
Output::
B //the neighbour brokers which *S* should be forwarded
PseudoCode:

```
1:  B ← φ;
2:  if br ∈ (B_I ∪ B_II ∪ B_V ∪ B_VI ) then
3:    B ← B ∪ B^Y(br);
4:  else if br ∈ (B_III ∪ B_IV ∪ B_VII ∪ B_VIII ) then
5:          B ← B ∪ B^X(br);
6:  else if br ∈ B_L then
7:          B ←B ∪ B^L(br);
8:  else if br ∈ B_A then
9:          B ←B ∪ B^A(br);
10: else if br ∈ B_X then
11:         B ←B ∪ B^X(br);
12: else if br ∈ B_Y then
13:         B ←B ∪ B^Y(br);
14: end if
15: return B.
```

Fig.4.4 Algorithm 4.1

Using HESPER-II, the SHBs also greedily forward the subscriptions to find the shortest paths between a PHB and a SHB. However, the brokers forward the subscriptions not only along the X-direction or Y-direction, but also along the L-direction and A-direction. This allows different SHBs to share longer event delivery paths than HESPER-I with a little increase of the subscription overhead. Fig.4.5(a) and (b) show the examples of subscription and event delivery paths

established by HESPER-II, respectively.

When a broker receives a new subscription *S*, it first checks its routing table. If a routing entry in the table can satisfy *S* or the broker is the target PHB, the broker sends the reply back to the sender of *S*; otherwise, the broker employs Algorithm 4.2, shown in Fig. 4.6, to determine the next-hop broker, and then forward *S* to the selected next-hop broker.

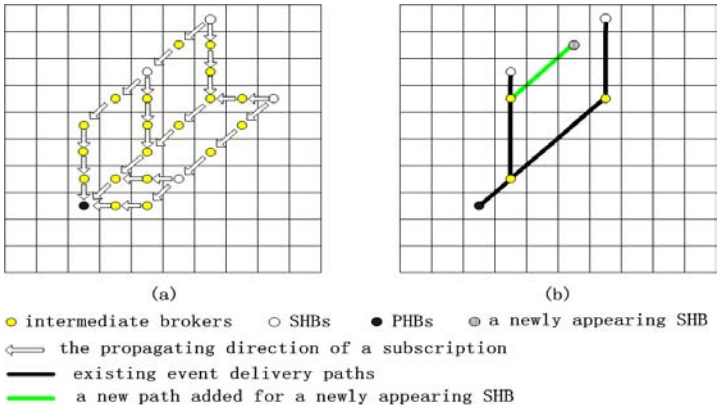


Fig. 4.5 Examples of the subscription and event delivery path established by HESPER-II

Algorithm 4.2: HESPER-II
Input::
br // this broker
Output::
B //the neighbor brokers which *S* should be forwarded
PseudoCode:

```
1:  B ← φ;
2:  if br ∈ (B_I ∪ B_II ∪ B_V ∪ B_VI ) then
3:      B ← B ∪ B^Y(br).
4:      if br ∈ (B_I ∪ B_V) then
5:          B ← B ∪ B^L(br).
6:      end if
7:      if br ∈ (B_II ∪ B_VI ) then
8:          B ← B ∪ B^A(br).
9:      end if
10: else if br ∈ (B_III ∪ B_IV ∪ B_VII ∪ B_VIII ) then
11:         B ← B ∪ B^X(br).
12:     if br ∈ (B_III ∪ B_VII) then
13:         B ← B ∪ B^A(br).
14:     end if
15:     if br ∈ (B_IV ∪ B_VIII ) then
16:         B ← B ∪ B^L(br).
17:     end if
18: else if br ∈ B_L then
19:         B ← B ∪ B^L(br);
20: else if br ∈ B_A then
21:         B ← B ∪ B^A(br);
22: else if br ∈ B_X then
23:         B ← B ∪ B^X(br);
24: else if br ∈ B_Y then
25:         B ← B ∪ B^Y(br);
26: end if
27: return B.
```

Fig. 4.6 Algorithm 4.2

Both HESPER-I and HESPER-II can achieve high efficiency and scalability. In addition, when a new arrival subscriber wants to participate in the system, it needs only to find the broker in the grid of the subscriber and, then, submit the subscriptions to the broker. If an event forwarded by the broker can satisfy the subscriptions from the subscriber, the broker just need to send an additional copy of the interesting event to new subscriber. Thus, the increase on the traffic load for one newly arrival subscriber is very limited. Because one broker need only serve the subscribers in one grid and the number of the subscribers in one grid is very limited, the performance of HESPER will not be greatly affected by the number of the

subscribers. This also shows the high scalability of HESPER-I and II.

Furthermore, we can show that the delay latency caused by HESPER is also very reasonable. For the simplicity of discussion on the delay latency, we assume that the sensor nodes are randomly distributed in a rectangular region, which composed of k*k equal-size gird. The subscriber is randomly distributed in the region and the publisher is located at grid (0, 0). According to HESPER, a delivery path from a publisher to a subscriber can be divided into two segments (see Fig. 4.7). The first segment PO is along the diagonal direction; the second segment is along the horizontal or vertical direction. Since the segments along vertical direction shares the same properties as those along horizontal direction, we just discuss the horizontal segment OS in this sub-section.

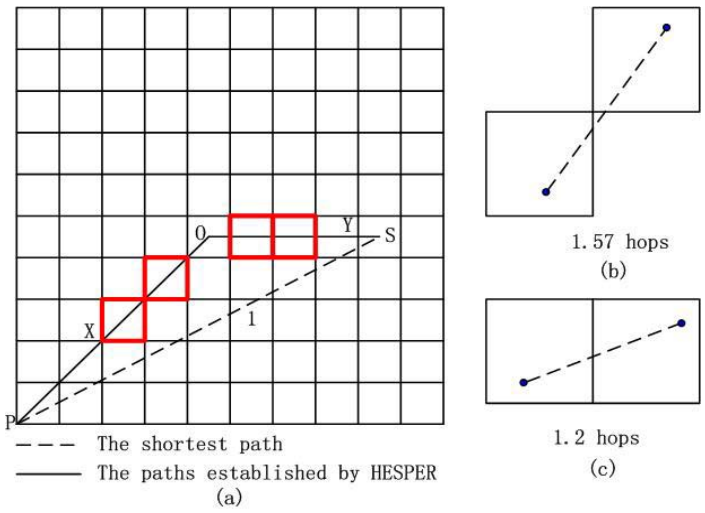


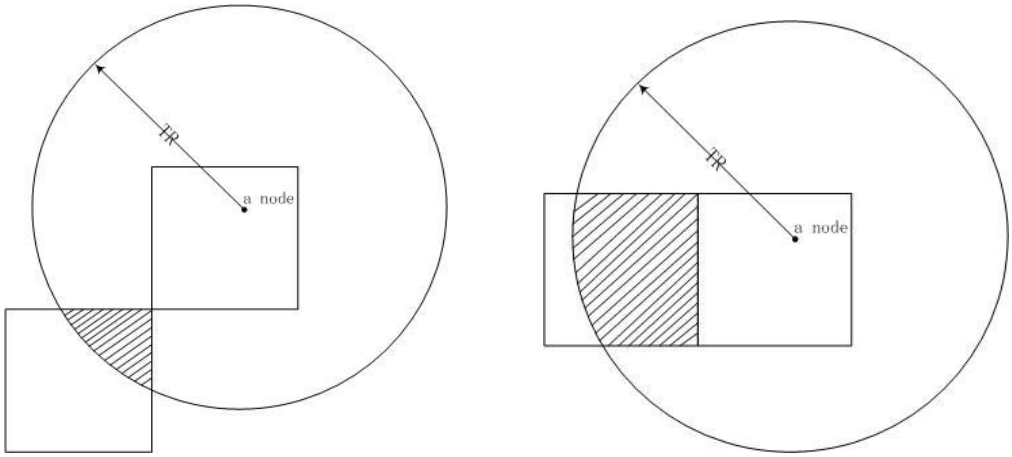
Fig. 4.7 the delivery path established by HESPER between a publisher and a subscriber

Next, we will discuss the impact of the diagonal and horizontal segments to the number of hops on a delivery path, respectively. We first consider the expected number of hops H_d from a node to another one in the diagonal neighboring grid on PO, as shown in Fig. 4.7 (b). Then, we consider the expected number of hops H_h

from a node to another one in the horizontal neighboring grid on OS, as shown in Fig. 4.7 (c). We have the following theorem.

Theorem 4.1 The value of the expected number of hops on the diagonal partial H_d of the delivery path is 1.56 and the value of the expected number of hops H_h on the horizontal and vertical partials of the delivery path is 1.21.

Proof: For the simplicity of discussion, we first show the following picture.



(a) (b)

Fig. 4.8 the hops between neighboring grids on the event delivery path

In Fig. 4.8(a), we show the grids of two sequential hops on the diagonal partial of the delivery paths. Without loss of generality, if there is a node *d* residing in a grid, only the nodes in the shadow part of the grid of the next hop can directly communicate with *d*. Since the nodes are randomly deployed in the whole region and the probability of a grid hosting at least one node is 1, we have the probability *p* of a node directly communicating with another node in the grid of next hop to be the following expression.

$$p = \frac{1}{A}\iint_{A}(A' \times 1)d\sigma\,, \tag{4.1}$$

where *A* is the size of a grid, *A'* is the size of the shadow area, $d\sigma$ is an integral element of the area. If the broker node of next hop is deployed in the shadow area, the hop number between two hops is 1; otherwise, the number is 2. Thus, we have the following expression (4.2).

$$H_d = p \times 1 + (1 - p) \times 2 \tag{4.2}$$

In fact, we can find that H_h shares the same expression with H_d. The formula expression of A' is very complex and not convenient to be used in expression derivation. Thus, we use that the numeric analysis to evaluate the values of H_h and H_d. The experimental results show that

H_h = 1.56 and (4.3)

H_d = 1.21 (4.4)

Sine the length of the grid diagonal is the transmission range, we can obtain the number of hops on PO and OS is *1.56x/TR* and $1.21y/(\frac{\sqrt{2}}{2}TR)$, respectively.

Now, we consider the total number of hops H_t of the delivery path established by HESPER and the minimum number of hops H_m of the shortest path in theory. Assuming the length of PS is 1, easily, we can have the following expressions.

$$H_t = 1.56x/TR + 1.21y/(\frac{\sqrt{2}}{2}TR). \tag{4.5}$$

$$H_m = 1/TR. \tag{4.6}$$

As for expression (4.5), *x* and *y* are the lengths of PO and OS, respectively. Next, we show the maximum number of hops of the delivery path established by HESPER in comparison with the shortest paths in theory. We need to obtain the maximum value

of the following function.

$$H_t / H_m = 1.56x + 1.71y . \quad (4.7)$$

According to the relationships among the internal angle and side length in a triangle, we have the expression (4.8).

$$\frac{x}{\sin \angle OSP} = \frac{y}{\sin \angle OPS} = \frac{1}{\sin \angle POS} \quad (4.8)$$

Obviously, $\angle POS = 135^\circ$. Then, denoting $\angle OSP$ to be α, we have expression (4.9) and (4.10).

$$x = \sqrt{2} \sin \alpha \quad (4.9)$$

$$y = \sqrt{2} \sin(45^\circ - \alpha) = \cos \alpha - \sin \alpha \quad (4.10)$$

Bring (4.9) and (4.10) into expression (4.7), we have the following function.

$$H(\alpha) = H_t / H_m = 1.71 \cos \alpha + 0.49 \sin \alpha \quad (4.11)$$

To obtain the maximum value of $H(\alpha)$, let

$$H'(\alpha)\big|_\alpha = 0. \quad (4.12)$$

Then, we obtain expression (4.13).

$$0.49 \cos \alpha - 1.71 \sin \alpha = 0 . \quad (4.13)$$

In addition, as for any α, we have the following formula:

$$\sin^2 \alpha + \cos^2 \alpha = 1 . \quad (4.14)$$

According to (4.13) and (4.14), we have

$$\sin \alpha = 0.275 \quad \text{and} \quad (4.15)$$

$$\cos\alpha = 0.96 . \quad (4.16)$$

Bringing (4.15) and (4.16) into (4.11), we have the maximum value of $H(\alpha)$ is

$$\max(H(\alpha)) \approx 1.78 . \quad (4.17)$$

Thus, we can find that the number of hop counts on the delivery path of HESPER is less than 1.78 times of that on the shortest path in theory. Considering the performance of HESPER in terms of subscribing overhead and publishing overhead, this is an attractive result. In the section V, we also show that the average number of hops on the delivery path of HESPER is no more than 1.5 times of that on the delivery path established by DF, according to our extensive simulation results.

4.4 Performance Evaluation

The performance of a pub/sub protocol can be measured by three metrics, including delivery overhead, subscribing overhead, and delivery latency, respectively. We have conducted extensive simulations to evaluate the performance of HESPER. We also compare the performance of HESPER with DF and SDCT. In this section, we report our extensive simulation results that show the significant performance advantages of HESPER in terms of the three metrics, respectively.

4.4.1 Simulation Setup and Performance Metrics

Recall that the performance of a pub/sub can be measured by delivery overhead, subscribing overhead, and delivery latency. To show the delivery overhead, we use the following two metrics to show the delivery overhead of a pub/sub protocol.

Delivery Message Cost: The number of messages sent for delivering an event to all subscribers. The number will increase one once a message has been sent from a sender to a receiver. Here, we do not consider the case where a sender can send to multiple receivers simultaneously.

Average Delivery Energy Cost: The total energy consumption for delivering an event to all subscribers divided by the number of subscribers. The energy consumption model for wireless transmission will be discussed later.

Similar to the delivery overhead, we use the following two metrics to measure the subscribing overhead.

Subscribing Message Cost: The number of messages transmitted for establishing the event delivery paths to all subscribers. The number will increase one once a message has been sent from a sender to a receiver. Here, we also do not consider the case where a sender can send to multiple receivers simultaneously.

Average Subscribing Energy Cost: The total energy consumption for establishing the delivery paths for all subscribers divided by the number of subscribers. Here, the energy consumption model for wireless transmission will be discussed later.

Many factors, such as wireless channel fading, node scheduling, the traffic load, etc usually affect the delay on a wireless delivery path. Therefore, the delay is not easily to be precisely measured. However, in a common sense, an event delivery along a path with a larger number of hops suffers from higher delivery delay. Thus,

we use the following hop count of the end-to-end delivery path to reflect the delivery delay of a pub/sub protocol.

Hop Count: The number of hops on an end-to-end delivery path from a publisher and a subscriber.

There are some existing software simulators for wireless sensor networks. The most famous one is ns2, which provides 802.11-based and 802.15.4-based simulation modules for WSNs. However, the performance of ns2 largely suffered by the network size in the simulation. Thus, ns2 is not suitable for the simulation of large-scale networks. In addition, the performance advantages of HESPER can be shown, when people use different underlying protocol and mechanisms. We have developed the simulation programs based on C language. However, the simulation parameters are referred to a well-known work [YOU04], where the authors determine the energy consumption of a packet by using only transmission range and the packet size. The major parameters used in the simulations are shown in Table 4.1.

Table 4.1 Parameters of the simulations

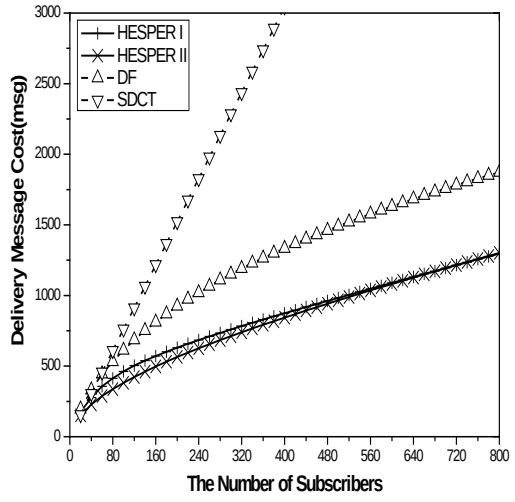
Deployment region	1000m*1000m
Network size	4000 nodes, 8000 nodes
Distribution of sensor nodes	random
Transmission range (TR)	70m, 100m
Threshold distance (TD)	75m
Number of subscribers	20 to 800
Distribution of subscribers	random
Position of the publisher	the center of grid (0, 0)
E_{elec}	50 *nJ/bit*
e_{fs}	10 $pJ/bit/m^2$
e_{mp}	0.0013$pJ/bit/m^4$
Message Size (M_b)	320 bits
Repeat times	100

In our simulation, all sensor nodes are randomly distributed in a rectangular region. To evaluate the scalability of the protocol, we use the different network sizes, including 4000 and 8000 nodes, respectively. We also varied the number of subscribers from 20 to 400. We evaluate the energy efficiency of the protocol using two different transmission ranges of nodes, which are corresponding to two energy consumption models, respectively. In general, the energy consumption on wireless transmission includes two parts [YOU04]. One is the energy consumption on the digital electronics of wireless receiver, denoted by E_{elec}; the other is used in signal transmission to overcome wireless channel fading, denoted by E_{amp}. We consider two different channel fading models, including free-space model and multi-path model. If $TR>TD$, the channel fading follows multi-path model and $E_{amp}=e_{mp}$. The energy consumption on a sender is $M_b*(E_{elec}+ e_{mp}*d^4)$. Otherwise, if $TR<TD$, the channel fading follows free-space model and $E_{amp}=e_{fs}$. The energy consumption on a sender is $M_b*(E_{elec}+ e_{fs}*d^2)$. However, in both cases, the consumption on a receiver is always M_b*E_{elec}. In addition, to obtain stable results, we repeat 100 times of the simulations with each combination of different parameter values and the average values of the results are shown in the result figures.

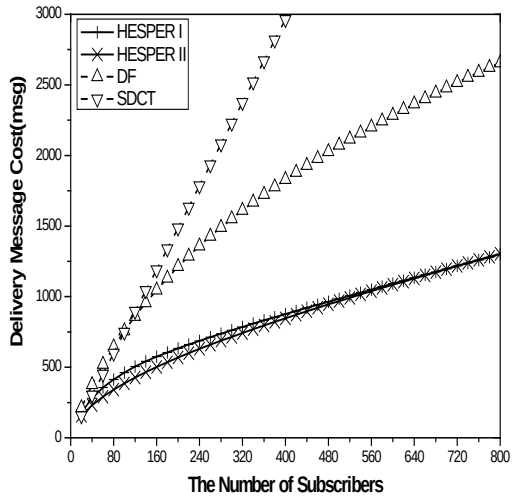
4.4.2 Simulation Results

Fig. 4.9 shows the impact of the number of subscribers to the delivery message cost of the protocols with different system parameters. In Fig. 4.9(a)(b), we can find that HESPER-II always outperform other protocols in terms of delivery message cost and the performance advantage of HESPER-II is larger when the network size

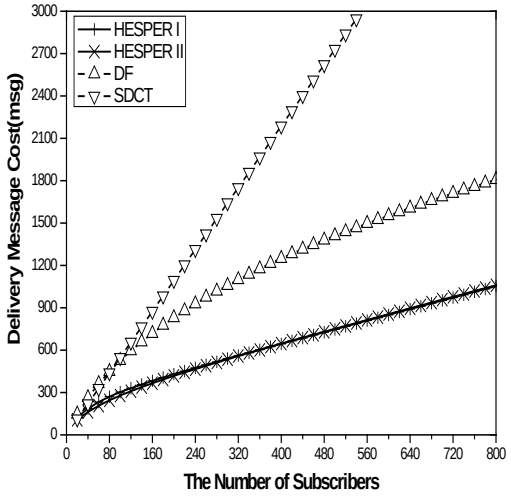
become larger. In the network of 4000 nodes, HESPER-II saves up to 30% and 80% delivery message cost compared with DF and SDCT, respectively; when the network size is 8000 nodes, it reduces 60% and 90% cost than DF and SDCT, respectively. As for HESPER-I, it cost nearly 10% higher cost than HESPER-II, when the number of subscribers is lower than 100. However, when the number of subscribers is larger than 400, both of the strategies achieve almost same performance. In Fig. 4.9(c)(d), we can see that the two HESPER strategies can achieve more significant performance advantages using the longer transmission range. More specifically, in the smaller network, HESPER-I and II outperform DF and SDCT up to 40% and 80%, respectively, while, in the larger network, they outperform DF and SDCT up to 70% and 80%, respectively.



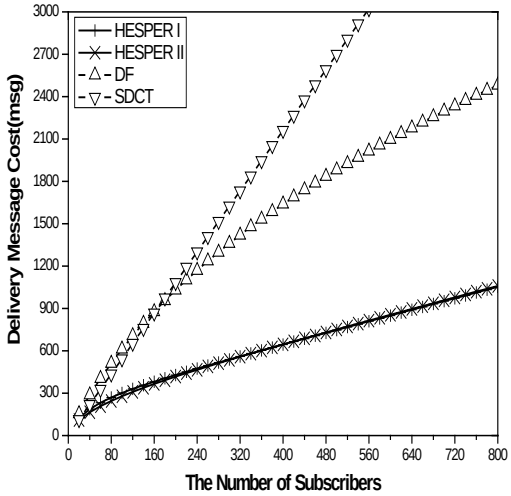
(a) 4000 nodes, TR=70m



(b) 8000 nodes, TR=70m



(c) 4000 nodes, TR=100m

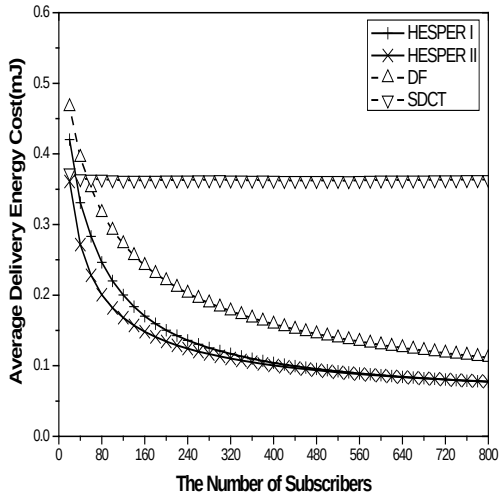


(d) 8000 nodes, TR=100m

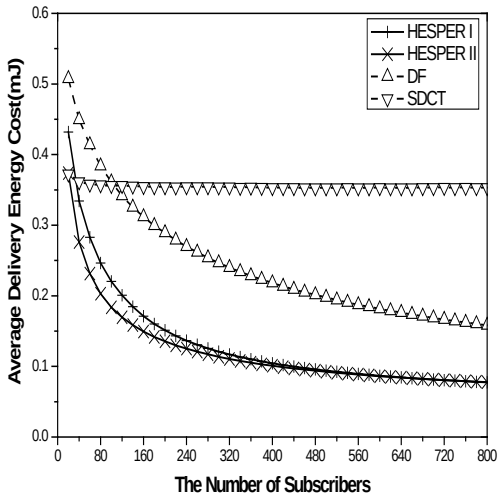
Fig. 4.9 delivery message cost of the protocols against the number of subscribers with different network sizes and transmission ranges

Fig. 4.10 shows the impact of the number of the subscribers to average delivery energy cost against the number of subscribers with different network sizes and transmission ranges. From Fig. 4.10, we can find that the average delivery energy cost caused by SDCT keeps stable. However, HESPER-I, HESPER-II, and DF perform better and better with the increase of the number of subscribers. HESPER-II always outperforms other protocols. When the number of subscribers is larger than 40, HESPER-I achieves lower energy cost than DF and SDCT, respectively. When the number of subscribers becomes 320, HESPER-I and HESPER-II keep stable, which shows good scalability of the two strategies. Although DF also shows the good scalability with large number of subscribers, DF cost higher energy cost than the proposed HESPER-I and HESPER-II. In addition, when the number of subscribers is smaller than 100, DF cost even higher energy cost than SDCT. When the transmission range is shorter, HESPER-I and HESPER-II outperform DF up to 30% and 50% in the networks with different sizes, respectively. Compared to SDCT,

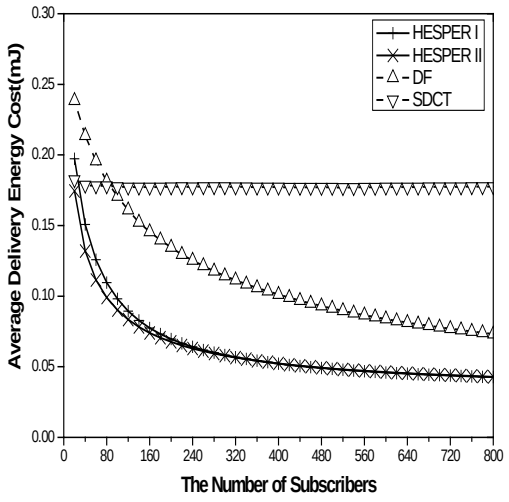
HESPER-I and HESPER-II save up to 70% energy cost in different networks. Furthermore, when the transmission range becomes longer, HESPER-I and HESPER-II outperform DF over 40% and 70% in different networks, respectively. In comparison with SDCT, HESPER outperform over 70%.



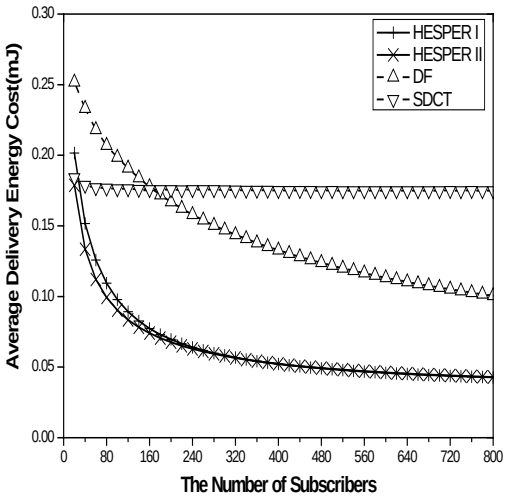
(a) 4000 nodes, TR=70m



(b) 8000 nodes, TR=70m



(c) 4000 nodes, TR=100m



(d) 8000 nodes, TR=100m

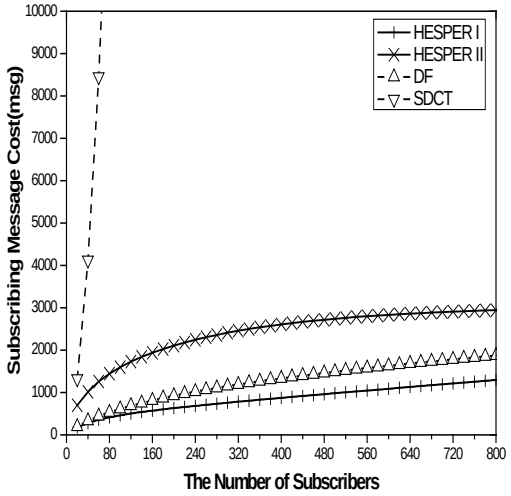
Fig. 4.10 average delivery energy cost of the protocols against the number of subscribers with different network sizes and transmission ranges

From Fig. 4.9 and Fig. 4.10, we can see that the proposed HESPER shows a lower

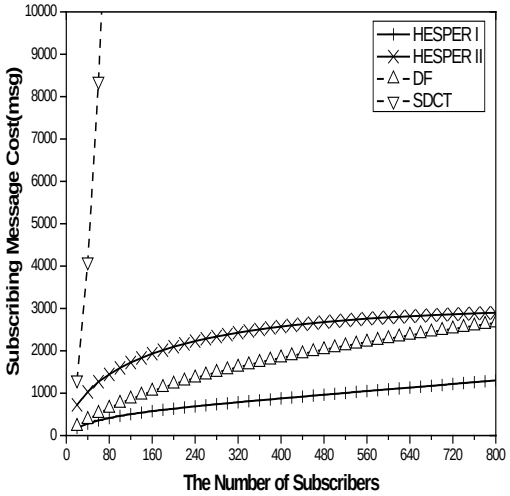
delivery message cost and energy cost in the most of the time and the energy cost caused by the HESPER rapidly becomes stable with different transmission ranges and network sizes, which shows a very attractive scalability. Compared with the performance of HESPER using a shorter transmission range, HESPER performs better when it uses a longer one. This is because that the number of hops on the end-to-end delivery paths becomes smaller with a longer transmission range and therefore, the number of messages for delivering an event from a publisher to a subscriber becomes smaller. However, we can see that two strategies of HESPER achieve larger performance advantage using longer transmission range in comparison with DF. Because GEAR used by DF always selects a neighboring node as the next-hop in a greedy geographical forwarding manner and can achieve only opportunistic overlapping among delivery path. When the transmission range becomes longer, a node in DF has more neighbor nodes to be the candidate next-hop nodes and therefore, the probability of a node selected by different nodes to be the next-hop node becomes lower. Thus, the probability of different paths overlapping with each other becomes lower and the impact of the transmission range to the performance of DF is not as significant as that to the performance of HESPER. As for SDCT, the delivery cost of it is much larger then the HESPER. Different from HESPER using the multi-tree to delivery event, SDCT just establishes a binary tree with all subscribers as leaf nodes. Thus, compared with HESPER, SDCT is not efficient, especially when there are a large number of subscribers.

Fig. 4.11 shows the subscribing message cost against the number of subscribers with different network sizes and transmission ranges. In Fig. 4.11(a)(b), we can find that HESPER-I always outperform other protocols in terms of average subscribing energy cost and the performance advantage of HESPER-I is larger when the network

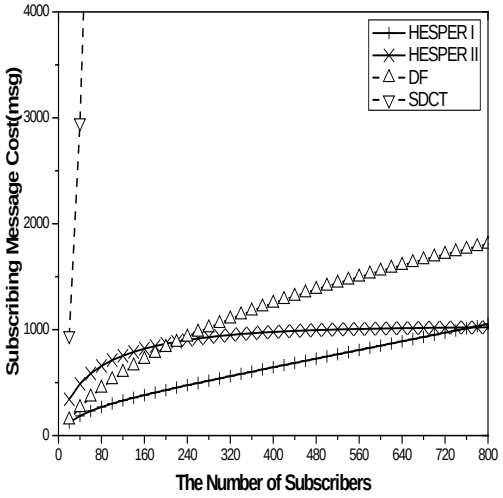
size become larger. In the network of 4000 nodes, HESPER-II saves up to 30% and 90% average subscribing energy cost compared with DF and SDCT, respectively; when the network size is 8000 nodes, it reduces 50% and 90% cost than DF and SDCT, respectively. As for HESPER-II, it cost nearly 3 times cost than DF, when the number of subscribers is lower than 80. However, with the increase of the number of subscribers, performance difference between HESPER-II and DF gradually drops and HESPER-II costs 1.5 times than DF with 800 subscribers. In Fig. 4.11(c)(d), we can see that HESPER can achieve more significant performance advantages using the longer transmission range. More specifically, in the smaller network, HESPER-I outperforms DF and SDCT up to 40% and 90%, respectively, while, in the larger network, HESPER-I outperforms DF and SDCT up to 60% and 90%, respectively. As for HESPER-II, it costs 2 times than DF when the number of subscribers is 20 in the smaller network. However, the performance difference between the DF and HESPER-II drops with the increase of number of subscribers. When the number of subscribers is larger than 220, HESPER-II outperforms DF and the cost of HESPER-II is only 60% of the DF's cost. In the larger network, when the number of subscribers is larger than 140, HESPER-II costs less than DF. The performance advantage of HESPER-II becomes more and more obvious with the increase of the number of subscribers. When there are 800 subscribers in the network, HESPER-II saves 60% message cost, compared with DF.



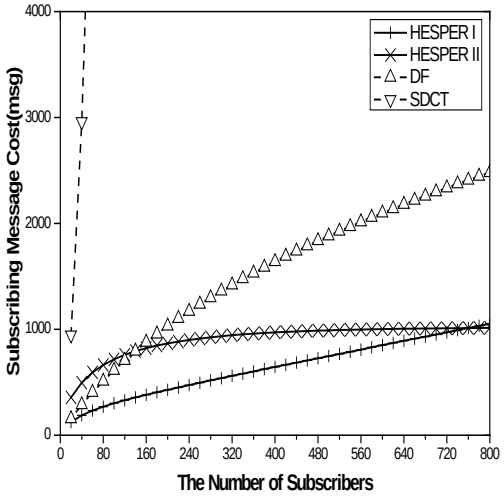
(a) 4000 nodes, TR=70m



(b) 8000 nodes, TR=70m



(c) 4000 nodes, TR=100m

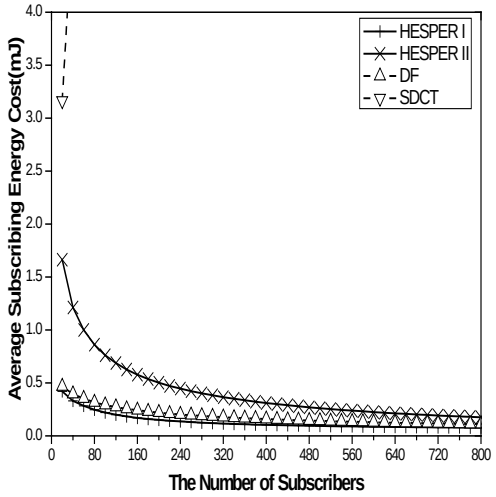


(d) 8000 nodes, TR=100m

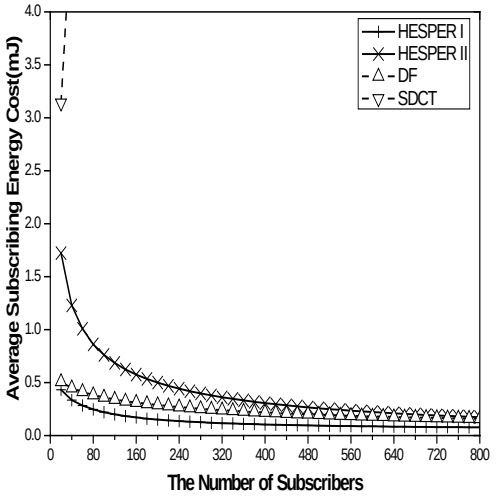
Fig. 4.11 subscribing message cost of the protocols against the number of subscribers with different network sizes and transmission ranges

Fig. 4.12 shows the average subscribing energy cost against the number of subscribers with different network sizes and transmission ranges. In Fig. 4.12(a)(b), we can find that HESPER-I always outperforms other protocols in terms of average subscribing energy cost and the performance advantage of HESPER-I is larger when the network size become larger. In the network of 4000 nodes, HESPER-I saves up

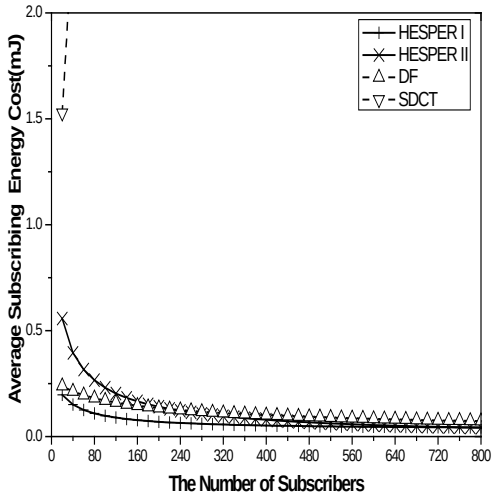
to 30% and 95% delivery message cost compared with DF and SDCT, respectively; when the network size is 8000 nodes, it reduces 50% and 97% cost than DF and SDCT, respectively. As for HESPER-II, it usually cost 3 times cost than DF, when the number of subscribers is lower than 80. However, with the increase of the number of subscribers, performance difference between HESPER-II and DF gradually drops and HESPER-II costs 1.5 times than DF with 800 subscribers. In Fig. 4.12(c)(d), we can see that HESPER achieves more significant performance advantages using the longer transmission range. More specifically, in the smaller network, HESPER-I outperforms DF and SDCT up to 40% and 90%, respectively, while, in the larger network, HESPER-I outperforms DF and SDCT up to 60% and 97%, respectively. As for HESPER-II, it costs over 2 times than DF when the number of subscribers is 20 in the smaller network. However, the performance difference between DF and HESPER-II drops with the increase of number of subscribers. When the number of subscribers is larger than 220, HESPER-II outperforms DF and the cost of HESPER-II is less 40% than DF. In the larger network, when the number of subscribers is larger than 140, HESPER-II costs less than DF. The performance advantage of HESPER-II becomes more and more obvious with the increase of the number of subscribers. When there are 800 subscribers in the network, HESPER-II saves 60% message cost, compared with DF.



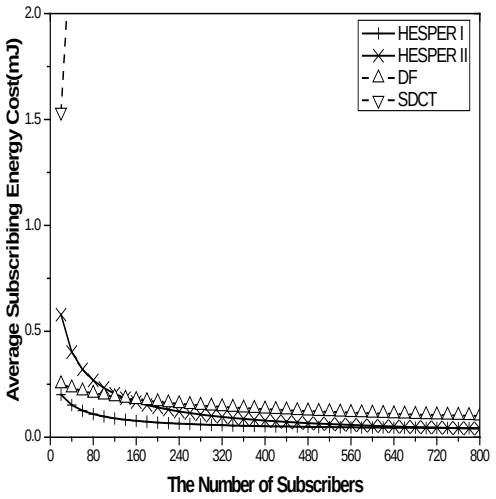
(a) 4000 nodes, TR=70m



(b) 8000 nodes, TR=70m



(c) 4000 nodes, TR=100m



(d) 8000 nodes, TR=100m

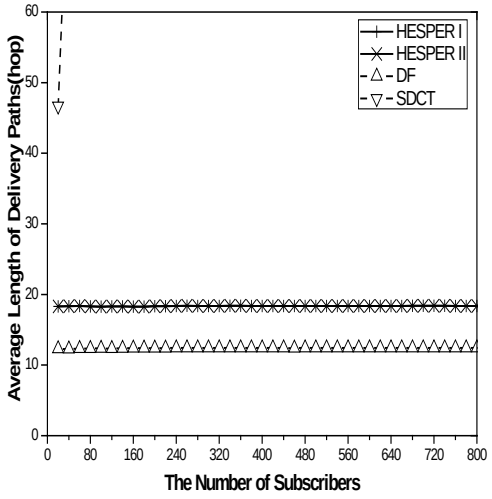
Fig. 4.12 average subscribing energy cost of the protocols against the number of subscribers with different network sizes and transmission ranges

From Fig. 4.11 and Fig. 4.12, we can see that the proposed HESPER protocol shows a lower subscribing message cost and energy cost in the most of the time. In addition, the subscribing energy cost caused by HESPER rapidly becomes stable with different transmission ranges and network sizes, which also shows a very attractive scalability. Compared with the performance of HESPER using a shorter

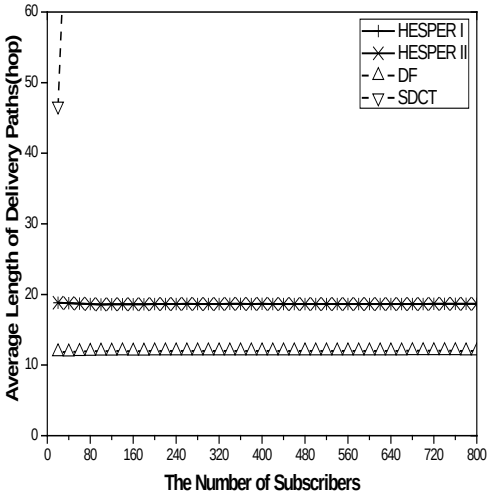
transmission range, HESPER perform better when they use a longer transmission range. This is because the number of hops to disseminate the subscriptions from a subscriber to a publisher becomes smaller with a longer transmission range and therefore, the number of messages for disseminating a subscription from a subscriber to a publisher becomes smaller. However, we can see that HESPER achieve larger performance advantage using longer transmission range in comparison with DF. Because GEAR used by DF always selects a neighboring node as the next-hop in a greedy geographical forwarding manner and can achieve only opportunistic overlapping among delivery path. When the transmission range becomes longer, a node in DF has more neighbor nodes to be the candidate next-hop nodes and therefore, the probability of a node selected by different nodes to be the next-hop node becomes lower. Thus, the probability of different paths overlapping with each other becomes lower and the impact of the transmission range to the performance of DF is not as significant as that to the performance of HESPER. As for SDCT, the subscribing cost of it is much larger than HESPER. Because SDCT should flood in the existing pub/sub tree for a newly emerging subscriber joining in the tree, the subscribing cost of SDCT is very high.

Fig. 4.13 shows the average length of delivery paths against the number of subscribers with different network sizes and transmission ranges. From Fig. 14, we can find that the average lengths of the delivery paths caused by HESPER-I, HESPER-II, and DF all keep stable. This is because both HESPER and DF use the geographical forwarding and thus, the number of hops of the established paths is only dependent on the transmission range of the nodes and the distance between a publisher and a subscriber. However, SDCT performs worse and worse with the increase of the number of subscribers. With the increase of the number of

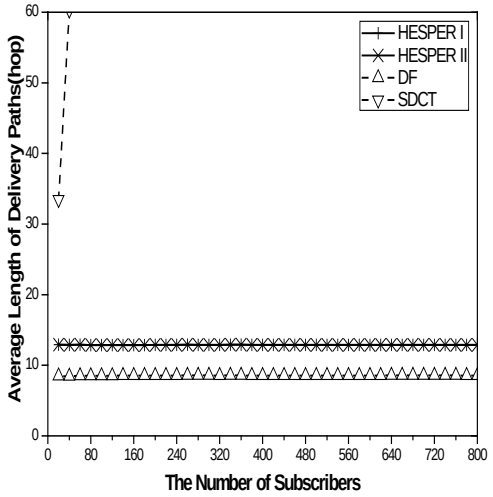
subscribers, the length of the pub/sub tree established by SDCT becomes larger and larger. HESPER-I and HESPER-II achieve almost the same performance. The average length of delivery paths caused by DF is shorter than that of HESPER and SDCT up to 50% and 90%, respectively. Obviously, DF uses the greedy geographical forwarding and thus, can achieve the smallest number of hops on every delivery path from a subscriber to a publisher. Different from DF, the HESPER, in fact, follow the guided geographical forwarding to achieve the overlapping among different delivery paths and thus, cause a larger number of hops on the delivery paths. In wireless sensor networks, many applications target at the long-term data collection and the nodes cannot be recharged easily. Thus, compared with the significant energy saving of HESPER, the higher delivery latency still can be very acceptable and attractive.



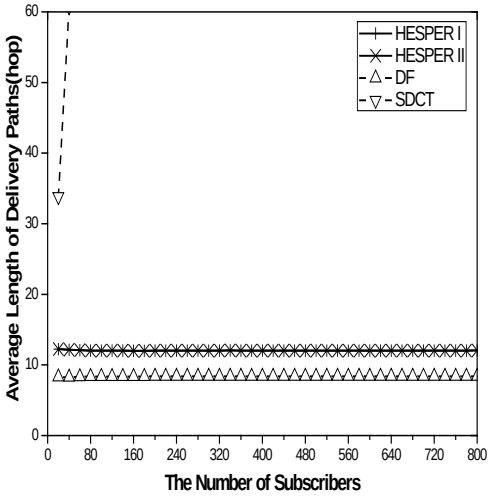
(a) 4000 nodes, TR=70m



(b) 8000 nodes, TR=70m



(c) 4000 nodes, TR=100m



(d) 8000 nodes, TR=100m

Fig. 4.13 average length of delivery paths of the protocols against the number of subscribers with different network sizes and transmission ranges

In the simulations, we have set the value of the transmission range to be 70m, which is referred to [YOU04] as mentioned previously. Using shorter or much shorter transmission ranges, all of the protocols, including SDCT, DF, HESPER-I and II, certainly cause higher cost of subscribing and event delivery. This is because the numbers of hops on the subscription dissemination and event delivery paths become large or much larger, respectively. However, when the simulations use lower value of the transmission range, the performance of HESPER can still keep the advantages in comparison with SDCT and DF. This is because HESPER-I and II need not use flooding and the change of transmission range will not involve flooding.

4.5 Summary

In this chapter, we describe *HESPER*, a Highly Efficient and Scalable Publish/subscribe protocol for wireless sEnsoR networks. HESPER uses the geographical information of the interesting events to establish the overlapping event delivery paths, which can greatly reduce the amount of event delivery and then, save the energy consumptions. Since the establishment of the delivery paths need not the coordination between subscribers, HESPER can further reduce the energy consumptions and achieve high scalability. In addition, HESPER offers two different strategies to balance the tradeoff between the subscription and publication costs to satisfy the different requirements of the practical applications. Our extensive simulation results show that, in comparison with the previous work of DF and SDCT, HESPER can save event delivery energy up to 50% and 80% respectively, and meanwhile, save the event subscription energy up to 50% and 90%, respectively.

Chapter 5

PUMA: A Reliable and Efficient Pub/Sub Protocol for WMNs based on Location Prediction

In this chapter, we describe the proposed reliable and efficient PUblish/subscribe protocol for wireless Mesh networks based on locAtion prediction, namely *PUMA*. This chapter is organized as follows. Section 5.1 briefly introduces this work. Section 5.2 describes the system model and basic assumptions. Section 5.3 describes the design of PUMA for WMNs in details. Section 5.4 proposes the event delivery protocol based on mobility prediction. Section 5.5 summarizes this chapter.

5.1 Overview

A wireless mesh network (WMN) is composed of a collection of base stations inter-connected by wireless links and cooperatively transmitting data for users [AKY05]. Taking the advantages of rapid deployment and flexible topology, this emerging networking paradigm is regarded as a promising solution to provide wireless Internet services to users in a large area, such as a high way, a large-scale community, and a metropolitan.

One of the major objectives of WMNs is to provide users data services, such as data transmission and data storage, on the Internet through wireless communications.

Publish/Subscribe (pub/sub) protocols are a widely used for supporting event services, which can encapsulate data into events and asynchronously exchange the events between distributed event publishers (generators) and subscribers (consumers) [BAN99][CAR01][EUG03]. Many works have been done in the context of the traditional distributed computing environments [BAN99][CAR01][EUG03]. However, these previous solutions are not suitable to WMNs, because WMNs are quite different from the traditional distributed systems due to the existence of a large number of mobile clients. Recently, some solutions have been proposed for designing Pub/Sub systems in the cellular networks to support mobile clients [BUR04][FIE03][ION04][POD04][SUT01][TAR03][WAN05]. However, all these works address only the problem of the clients' reconnection, where the clients were disconnected from the base station and re-connect to another base station in a new place after movement. Different from cellular networks, WMNs need to handle the problem of how to provide continuous event service to the mobile clients, which is much more challenging. This is because of the following reasons.

- *Reliability.* In a WMN, a client moves around in the network and meanwhile, requires continuous wireless communication service. The event delivery path should be changed along with the changing of the client's location. However, the changing of event delivery path may not follow the movement of the client. The event cannot be successfully delivered to the client and the client will lose the event. Thus, the reliability of the event delivery should be guaranteed, if the pub/sub can support mobile clients.

- *System Message Cost.* In a WMN, the base stations in a WMN are connected by wireless links. Thus, the bandwidth in the network is limited. A pub/sub system over a WMN needs to serve a large number of subscribers. The events in a

pub/sub system can cover many topics, such as news, weather forecast, traffic information, etc. Thus, the number of the event will be also very large. Considering the limited bandwidth in a WMN, the message cost caused by a pub/sub protocol should be as low as possible.

- *Energy Efficiency of Mobile Clients.* Usually, the users of WMNs use the hand-held devices, such as, notebook, PDA, mobile phone, etc. Most of such devices are powered by the attached batteries. Due to the limitation on the size of the hand-held devices, the size of the attached battery is very limited and the mobile clients likely to be exhausted quickly. Thus, a pub/sub protocol should try it best to reduce the message cost of the mobile clients and then, the energy consumption of the clients can be reduced.

In this chapter, we address the problem of how to support continuous event delivery for mobile clients in a pub/sub system for WMNs. Aiming at solve the problem, we propose a reliable and efficient PUb/sub protocol for wireless Mesh networks based on locAtion prediction, namely *PUMA*. Using the location prediction, PUMA can estimate all of the candidate locations where the interested client possibly stays and then, deliver the events to all of such candidate locations. We show that such an event delivery with minimum delivery cost is a minimum stenier tree (MST) problem, which is NP-hard. However, according to the calculation of the possibility of a client staying in a grid, we can find that a client has a low possibility to stay in partial r of the candidate locations, namely cold locations. Delivery events to the cold locations may cause high unnecessary message cost due to the large number of the cold locations. To further reduce the message cost caused by cold locations, PUMA defines a probability threshold and at first, delivers the events to the locations, namely hot locations, where the sum of the possibility of a

client staying in is larger than the probability threshold. Once the client is not hot locations, PUMA will then deliver the events to the cold locations. Therefore, PUMA can guarantee the reliability of the event delivery and meanwhile, achieve high efficiency. We can show that, given a probability threshold, delivering an event to satisfy the threshold is still a NP-hard problem. Also, we proposed an algorithm for the problem with polynomial time complexity. We conduct extensive simulations to evaluate the performance of our protocol and algorithm. The simulation results show that, in comparison with the paging algorithms for clients' mobility management [PIT97], PUMA can save up to over 90% system message cost and over 60% client message cost, respectively.

5.2 System Model and Assumptions

In this section, we first describe the network model and the mobility model of the mobile clients used in this chapter. Then, we describe the model and assumptions of events and filters in our pub/sub system. We also discuss the pub/sub architecture model.

We assume that the WMN is deployed in a rectangular region, which is composed of grids with identical size. Every grid hosts only one mesh base station and one base station resides in only one grid. A base station can directly communicate with only the base stations, which stay in the vertical and horizontal neighboring grids of the base station's grid. In addition, a mobile client can directly with only the base station in the grid where the client stays.

We assume that the movement of clients follows the random walker model (RWM)

[CAM02], where a client moves from its current grid to a neighboring grid with a probability *p*. The value of *p* is equal to 1/*n*, where *n* is the number of candidate neighboring grids. The candidate neighboring grids will be both vertical and horizontal neighbor grids nearest to the destination. Thus, *n* is 4. Also, for the simplicity of the discussion, we assume that a client moves with the same constant speed. We also assume that, in a unit time, a client can move from its current grid to one of its neighboring grid. In practice, many clients in WMNs are moving along the streets in cities or towns. The streets usually divide the cities and towns into grids. In addition, although there are some other mobility models, for example random waypoint model, these models can also help people to predict the mobility status of a node. Thus, these models can also be used in the PUMA. In this chapter, we use the random walk model to describe the mobility of the clients and discuss the performance of PUMA.

Similar to the existing content-based Pub/Sub system [BAN99][CAR01], an event is described by a tuple of Attribute-Value, while a subscription filter is described by a tuple of Attribute-Value-Operator. For example, event *E* <stock price, 10.9> means that the stock price is $10.9 and filter *F* < stock price, 10, '>'> means that the subscribers need to receive the event where the stock price is higher than $10. We say that an event can match a filter when the topic of the filter and the event are the same and the comparison between the values of the filter and the event satisfies the "operator" in the filter [CAR01]. Obviously, event *E* matches filter *F* in the above example.

We assume that the Pub/Sub system uses the broker/client architecture, where brokers perform the filter/event transmission and management. The clients in the system subscribe or un-subscribe their interests to the system by just registering the

corresponding subscription or un-subscription to their brokers. Every mesh base station is a broker and every mobile user is a client. The functions of the broker and client are implemented by running corresponding programs on the base stations and user devices, respectively.

5.3 Protocol Description of PUMA

In this section, we describe our PUMA protocol. First, we discuss the preliminaries used in the description of PUMA. Second, we present how to calculate the probability of a mobile client staying in a grid at moment *T*. Third, we describe PUMA using all-location delivery strategy. Forth, we describe PUMA using hot-location delivery strategy.

We assume that each grid have a system-wide unique ID. Also, we assume that the total number of the grid is $n=l*l$ and for the simplicity of discussion, L is an odd integer. We denote $l = 2*k+1$.We also assume that a mobile client starts its movement from the central grid of the region and accordingly, the client is served by the broker staying in the central grid. For the simplicity of discussion, we first denote the location of a grid as a 2-tuple $<x, y>$, where x and y keep the locations of a grid in the vertical and horizontal directions, respectively. In addition, the value of x of a grid is 1 larger than that of the left neighbor grid, while the value of y of a grid is 1 larger than that of the lower neighbor grid. In this chapter, the location of the grid at the bottom-left corner is noted as <0, 0>. Thus, the location of the upper neighbor grid of grid<0, 0> is <0, 1> and that of the right neighboring grid is <1, 0>.

Then, we assign a system-wide unique ID to every grid. The ID's value of grid at <x, y> is equal to $x*l+y$.

Similar to most of the existing Pub/Sub systems [CAR01], all brokers in our system are organized into an acyclic undirected graph (AUG), where every broker uses the reverse-path-forwarding protocol to receive the events subscribed by its clients. A broker will propagate the subscriptions received from its clients in the AUG to all other brokers, establishing a data collection tree rooted at this broker. The events matching the subscription will be reversely forwarded along the tree until they reach the root, i.e., the subscribing broker. The subscribing broker will further deliver these events to the corresponding clients.

5.3.1 Description of Protocol

First, we introduce the basic idea of PUMA. In PUMA, a client first submit its subscription to the broker which can directly communicate with the client and the broker, namely previous broker, will help the client finish the event subscription. Then, the client starts its movement in the network by following the random walk model. During the movement of the client, the broker may receive an interesting event subscribed by the mobile client but may not deliver the events directly to the mobile client, because the client already moves out of the direct communication range of the broker. The previous broker will calculate the probabilities of the client staying in each grid. Then, the previous broker has two choices. One is that PUMA delivers the receive event to all of locations with the probability larger than 0 using a

minimum message cost. Another one is that PUMA select some of the grids, namely hot locations, which have a higher probability to host the client and then, first deliver the event to all of the hot locations with a minimum message cost. After that, if the event has not been delivered successfully, the broker will deliver the event to the remaining grids which has a probability larger than 0, but are not hot ones.

Upon receiving an interesting event, the previous broker will calculate the length of time that the client has left. Assuming the time is *T*, the broker can calculate the probability of a mobile client staying in a grid at time *T* as follows. According to [NEL98], the random walk model of a client moving in the grids can be formulated as a *Markov Chain model*. In the markov chain model, the probability of the initial location of a client can be description by a vector, denoted as P^0 as follows.

$$P^0 = (p_0^0, p_1^0, p_2^0, ..., p_{x*l+y}^0, ..., p_n^0), \tag{5.1}$$

where p_{x*l+y}^0 denotes the probability of the client staying at grid <*x*, *y*> before the movement. Obviously, we have $p_{k*l+k}^0 = 1$, because the client starts from the central grid according to our assumption. Also, we have the transition matrix of the random walk model as follows.

$$T = \begin{pmatrix} p_{0,0}, & p_{0,1}, & ..., & p_{0,j}, & ..., & p_{0,n} \\ p_{1,0}, & p_{1,1}, & ..., & p_{1,j}, & ..., & p_{1,n} \\ ... & ... & ... & ... & ... & ... \\ p_{i,0}, & p_{i,1}, & ..., & p_{i,j}, & ..., & p_{i,n} \\ ... & ... & ... & ... & ... & ... \\ p_{n,0}, & p_{n,1}, & ..., & p_{n,j}, & ..., & p_{n,n} \end{pmatrix}, \tag{5.2}$$

where $p_{i,j}$ denotes the probability of the client moving from grid *i* to grid *j* during one unit time. According to our assumption on the random walk model, we can easily obtain the following expressions.

$$p_{i,j} = \begin{cases} 1/4, & if\ |i-j| = 1; \\ 1/4, & if\ |i-j| = l+1; \\ 1/4, & if\ |i-j| = l-1; \\ 0, & others. \end{cases} \tag{5.3}$$

Expression (5.3) shows that a client has a probability of 1/4 to move from a grid to this grid's left, right, upper, and lower neighbor grids, respectively. Here, we ignore the edge effect. Thus, we can obtain the probability vector, denoted as P^t which shows the probability of a client staying in every grid at time *t*. Here, *t* is an integral times of one unit time.

$$P^t = (p_0^t, p_1^t, p_2^t, ..., p_{x*l+y}^t, ..., p_n^t) = P^0 * T^t = (P_0^0, P_1^0, ..., P_n^0) * \begin{pmatrix} p_{0,0}, & p_{0,1}, & ..., & p_{0,j}, & ..., & p_{0,n} \\ p_{1,0}, & p_{1,1}, & ..., & p_{1,j}, & ..., & p_{1,n} \\ ... & ... & ... & ... & ... & ... \\ p_{i,0}, & p_{i,1}, & ..., & p_{i,j}, & ..., & p_{i,n} \\ ... & ... & ... & ... & ... & ... \\ p_{n,0}, & p_{n,1}, & ..., & p_{n,j}, & ..., & p_{n,n} \end{pmatrix}^t \tag{5.4}$$

Upon obtaining the probability of the client staying in each grid, the broker will learn about the grids where the client may stay at time *t*. Then, the broker establishes a minimum event delivery tree and this is rooted at the broker and accommodates all the brokers staying in the grids the client may stay in. To find such a delivery tree, the broker first establishes a graph $G(t)=<V, E>$ for the event delivery at time *t*, which has the following properties.

a) a vertex *v* in *G* is corresponding to a broker in the WMN and the vertex corresponding to the central broker is denoted as *v*'.

b) If and only if two base stations in the WMN can directly communicate with each other, then there is an edge *e* between two vertices corresponding to the two brokers, respectively.

c) The weight of an edge *e* is the length of it, noted as |*e*|. Let |*e*|=1.

d) The weight of a vertex *v*, denoted as |*v*|, is the probability of the client staying in the grid corresponding to *v*. Thus, $\left|v_{x*l+y}\right| = p^{t}_{x*l+y}$.

Now, we formally define the problem of how to deliver the event to all possible locations with minimum message cost, namely all-location delivery (ALD) problem, as follows.

Problem 5.1 (**ALD problem**):

Find a tree *tr* <*V'*, *E'*>, s.t.,

a) $v' \in V'$, $V' \subset V$, and $E' \subset E$;

b) $\forall v \in V$ and |*v*|>0, $v \in V'$

minimize $\sum_{e \in E'} |e|$.

In fact, the ALD problem is an example of Rectilinear Minimum Stenier Tree problem, which has been shown as a NP-complete problem [GAR77]. To solve the ALD problem, we give an approximate algorithm, namely ALD-algorithm, which is a variant of the algorithm in our previous work[DEY06] and this algorithm has been proved with an approximate ratio of 4. The description of our heuristic algorithm as follows.

ALD-Algorithm
Input: a graph G=(V, E)
Output: a tree for ALD spanning on tr=(V', E')

Step1: Construct an auxiliary graph which is a complete weighted graph H = (V_H, E_H). where $\forall v \in V_s, |v| > 0$, and $\forall e_{i,j} \in E_s$, $i = x_1 * l + y_1$, $j = x_2 * l + y_2$, $|e_{i,j}| = |x_1 - y_1| + |x_2 - y_2|$. Here, i and j are the IDs of the brokers which are corresponding to the two vertices of $e_{i,j}$ in G, respectively.

Step2: Compute a minimum spanning tree T in H.

Step3: Replace each edge of T with a corresponding path in G. Then delete cycles to obtain an tree for ALD.

Fig. 5.1 ALD Algorithm

After obtaining the delivery tree ALD algorithm, the previous broker can use the delivery tree to deliver the event to all of the brokers which are in the grids with the probability larger than 0.

However, according to the calculation results of the probability, we can find that, in fact, most of the locations where a client possibly stays has a very low probability to host the client. We call such locations as cold locations. If the event is delivered to the cold locations, most of messages are very likely to be wasted. To further reduce the message cost, PUMA can select the locations with high probability to host the client, namely hot locations, and first deliver the event to them with a minimum message cost. If the event has not been delivered successfully, then PUMA deliver the event to the remaining cold locations with a minimum message cost. However, it is not easy to determine the hot locations. Thus, we define a threshold *th* of the sum of the probability values of the hot locations. Then, the broker can first select a subset of the grids as hot locations and the sum of the probability values of the grids

is larger than *th*. Then, the broker deliver the event to the grids in these hot locations. Now, we formally define the problem of delivering an event to hot locations with a minimum message cost, namely HLD problem, as follows.

Problem 5.2(HLD problem):

Find a tree *tr* <*V'*, *E'*>, s.t.,

a) $v' \in V'$, $V' \subset V$, and $E' \subset E$;

b) $\sum_{v \in V'} |v| \geq th$, where $0 < th \leq 1$;

minimize $\sum_{e \in E'} |e|$.

Now, we prove that HLD problem is NP-complete. We first prove that the decision version of HLD problem is NP-hard by giving a polynomial time reduction from the Discrete Rectilinear Steiner Minimum Tree problem, which is known to be NP-complete [NEL77]. Let us first formally define the two decision problems.

Problem 5.3 (Rectilinear Steiner minimum tree). Given a set *X* of integer-coordinate nodes in the rectilinear plane and a positive integer *L*, does there exist a set $Y \supseteq X$ of integer-coordinate nodes such that some spanning tree *T* of *Y* satisfies the condition $l(T) \leq L$, where *l*(*T*) is the total length of *T*?

Problem 5.4 (HLD tree with a bounded length): Given a *G'* = (*V'*, *E'*), where *V'* is a set of weighted nodes in the rectilinear plane, a non-negative integer *L'*, a non-negative number *n*, a subset *S* of *V'*, where any vertex has a weight larger than 0 and the weight sum of all vertices is 1, and a subset *S'* of *S*, where the weight sum of all vertices is larger than *n*, does there exist a tree spanning $Q \supseteq S'$, where $S \supseteq S'$, such that $l(Q) \leq L'$, where *l*(*T*) is the total length of *T*.

Theorem 5.1 There is a polynomial time reduction from Problem 5.3 to Problem 5.4. Therefore, Problem 5.4 is NP-hard.

Proof. Let *I* be an instance of Problem 1. We construct an instance *I'* of Problem 5.2 by letting *V*' be all nodes in the rectilinear plane, and *G'* be the graph *G* corresponding to Problem 1, and by setting $L' = L$ and $n = 1$. Thus, $S' = S$. It is clear that this construction of *I'* from *I* takes polynomial time. It is easy to prove that *T* is a solution for *I* if and only if *T* is a solution for *I'*. Therefore, Problem 5.4 is NP-hard.

Theorem 5.2 The HLD problem is NP-complete.

Proof. Obviously, the HLD problem belongs to the class NP. Thus, together with the result of Theorem 5.1, we can conclude that the HLD problem is NP-complete.

To solve the HLD problem, we give a heuristic algorithm, namely HLD-algorithm, which is based on the ALD-algorithm previously mentioned. Figure 5.2 shows the details of the HLD algorithm.

HLD-Algorithm
Input: a graph G=(V, E)
Output: a tree for HLD
Step1: Sort all v in a descending order according to $|v|$.
Step2: Obtain a subset V_s of V, s.t.,

a). $\sum_{v \in V'} |v| \geq th$;

b). Denote the ranking of v as $r(v)$. For $\forall v_1 \in V_s, \forall v_2 \in V - V_s$, $r(v_1) > r(v_2)$.

c). Denoting the number of vertices in V_s as $|V_s|$, minimize $|V_s|$.

Step3: Construct an auxiliary graph which is a complete weighted graph H = (V_s, E_H). where $\forall e_{i,j} \in E_H$, $i = x_1 * l + y_1$, $j = x_2 * l + y_2$, $|e_{i,j}| = |x_1 - y_1| + |x_2 - y_2|$. Here, i and j are the IDs of the brokers which are corresponding to the two vertices of $e_{i,j}$ in G, respectively.

Step4: Compute a minimum spanning tree T in H.

Step5: Replace each edge of T with a corresponding path in G. Then delete cycles to obtain a tree for HLD.

Fig. 5.2 HLD Algorithm

Obviously, the most difficult work in the solution of HLD problem is how to determine the hot locations. In HLD algorithm, the broker first ranks the vertices according to the value of the probabilities of grids corresponding to the vertices. According to the ranking result, the broker just selects the minimum number of vertices with the highest ranking and the sum of the weights of the vertices is larger than *th*. After obtaining the tree for HLD, the broker first delivers the event to the hot locations along the tree. Once the event has not been successfully delivered, the broker then deliver the event to the remaining cold locations, where the broker can use the ALD but the V_H should only include the vertices corresponding to the cold locations in step 2.

5.4 Performance Evaluation

We have conducted extensive simulations to evaluate the performance of PUMA. In this section, we will report these simulation results and compare the performance of PUMA with the previous work. First, we introduce the simulation setup. Then, we discuss the performance metrics, including average system message cost and average client message cost. Then, we show the simulation results of PUMA and the comparison to the previous work of paging algorithm [PIT97]. The results show that our algorithm can significantly outperform the paging algorithms in terms of the average system message cost and average client message cost.

5.4.1 Simulation Setup and Performance Metrics

In the simulations, the region where a client moves is a rectangular region, which is composed of 25*25 grids. In the region, a grid hosts only one base station and a base station resides in only one grid. A base station can directly communicate with only the base stations that reside in the vertical and horizontal neighboring grids. A client starts its movement from the central grid of the region and moves by following a random walk model, where the client can move from one grid to one and only one vertical or horizontal neighboring grid within a unit time. Here, we have used only one client. When there are many clients in an application system, the system can handle the clients one by one. However, if there are multiple clients being interested in the same events simultaneously, the definition of the problem has changed. We will discuss it the future work.

The duration of the whole simulation is 12 unit times. The client can directly communicate only with the base station residing in the grid that the client moves in. We set the temporal interval, denoted as I_{event}, between two sequentially arriving events to be 2, 3, 4, 5, 6, and 7 unit times. The different intervals have different application background, which means different applications have specific intervals in the event generations. For example, the real-time traffic events, such as car collisions, traffic congestion, traffic light switches, can occur very frequently and the arrival interval of the events can be very short. However, the email notifications usually will come less frequently or seldom. Thus, the arrival interval of events will be longer. In the simulation, we assume that the event transmission delay between the base station residing in the central grid and any of other base stations in the region can be ignored. This is very reasonable because the duration of a client moving from one grid to another is much longer than that of an event being transmitted from the central grid to another one.

In addition, we set the value of the threshold, denoted as *th*, used by PUMA are 40%, 50%, 60%, 70%, 80%, 90%, and 100%, respectively. Using different values of *th*, we can determine the impact of *th* to the performance of PUMA and find the optimal value of *th* when using different system parameters. We also provide the paging algorithm as the performance benchmark. In the paging algorithm, a client will proactively report its location to the previous broker once the client moves from one grid to another one and the distance, denoted as *dist*, between the two grids is beyond a certain value. The value of *dist* between two grids is equal to the shortest duration when a client moves from one to the other. In the simulations, we set the value of *dist* to be 2, 3, 4, 5, 6, 7, and 8. Table 5.1 shows the values of the parameters used by our simulations.

Table 5.1 Parameters of the simulations

size of the movement region	25*25 grids
the number of base stations	25*25
deployment of base stations	at least and only one in one grid
number of clients	1
mobility model	random walk
starting point of the movement	the central grid of th;e region
the length of the simulation (unit time)	12 (unit times)
I_{event}	2, 3, 4, 5, 6, 7 (unit times)
value of *th*	40%, 50%, 60%, 70%, 80%, 90%, 100%
value of *dist*	2, 3, 4, 5, 6, 7, 8

We have developed our own simulator based on C language. There are some existing simulators, for example ns2, which have provided the simulation modules for wireless network, which focus on the mechanisms of the underlying layers, such as physical layer, MAC layer, etc. However, as we have mentioned in Chapter 4, due to the efficiency of simulations, ns2 cannot support the simulation based on a large number of wireless nodes. Furthermore, ns2 cannot support the simulation of WMNs very well. In fact, PUMA can work on the networks with different underlying network details. Thus, in the simulations of PUMA, we have not involved many specific issues of the underlying layer in a network. We just focus on the number of message transmitted on the upper layer. Thus, we just use our own simulator to simulate the routing layer of a WMN.

The performance of a pub/sub protocol can be measured by message cost. In a WMN, the message cost has two different categories, including system message cost and client message cost. As for the PUMA protocol, the messages include the location reports and the events. Using the paging algorithm, a client needs to generate the location reports and then, send the reports to the base station that is the nearest one to the client. Then, the base station will use the multi-hop relay to send

the reports to the base station that hosts the previous broker serving the client. In addition, upon receiving an event, the previous broker needs to send the event to the base station that forward the latest location report of the client. Then, that base station need to broadcast the event to the other base stations that may directly communicate with the client. As for PUMA, the broker previously serving a client also need send the receiving the interesting event to the base stations that may directly communicate with the client. After receiving the event, the client needs to send the broker a reply, which includes an acknowledgement to the event and the current location of the client. In the simulations, we assume that one message can contain an event, a location report, or a reply.

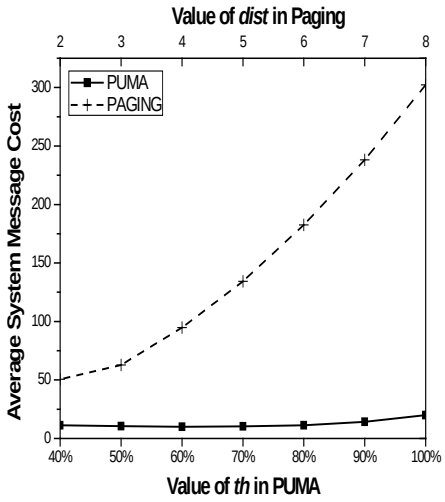
Usually, we consider the two costs averaged by the number of the delivered events. Thus, we have average system message cost and average client message cost, respectively. The specific definitions of the two costs are as follows.

Average System Message Cost. The number of messages that are sent by the base stations and clients for delivering one event. This metric can effectively measure the overhead of the system on delivering an event. Thus, the low cost of the system shows the high efficiency of a pub/sub protocol.

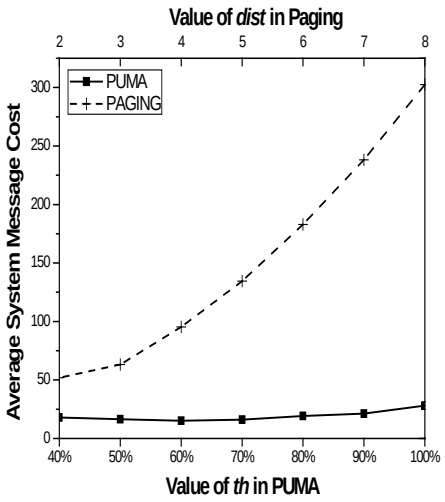
Average Client Message Cost. The number of messages that are sent by a client for receiving one event. This metric can effectively measure the overhead of a client upon receiving an event. Usually, the power and computation resources of a client are much less than a base station. Thus, the cost of a client should be as low as possible.

5.4.2 Simulation Results

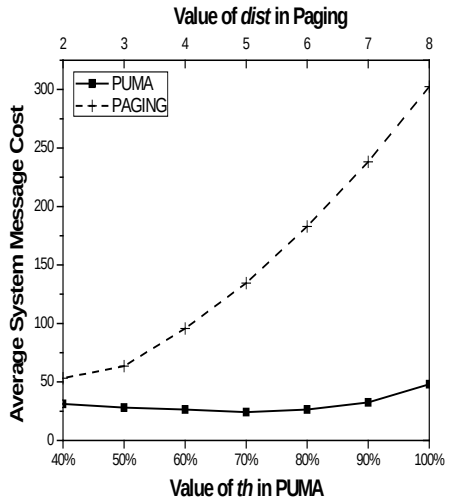
Fig. 5.3 shows the impact of the *th* value to average system message cost of the pub/sub protocol when the temporal interval between two sequentially arriving events is 2, 3, 4, 5, 6, and 7 unit times. From all of sub-figures, we can find that the minimum value of average system message cost is 10, 15.2, 24.2, 31.4, 45.2, and 52.6, when I_{event} is 2, 3, 4, 5, 6, and 7, respectively. In comparison with the paging algorithm, the lowest average system message cost of PUMA is only about 20%, 29%, 45%, 57%, 81%, and 91% of the cost incurred by the paging algorithm, when I_{event} is 2, 3, 4, 5, 6, and 7 unit times, respectively. Also, we can find the optimal value of *th* is 60% when I_{event} is 2 and 3 unit times. The optimal value of *th* is 70% when I_{event} is 4 unit times. Then, the optimal value of *th* becomes 80% when I_{event} is 5, 6, and 7 unit times. From 6 sub-figures, we can find that the average system message cost are raised from 20 to 124 along with the value of I_{event}, when the value of *th* is 100%. From all of the sub-figures, we can find that the best performance of paging algorithm are achieved when the value of *dist* is 2.



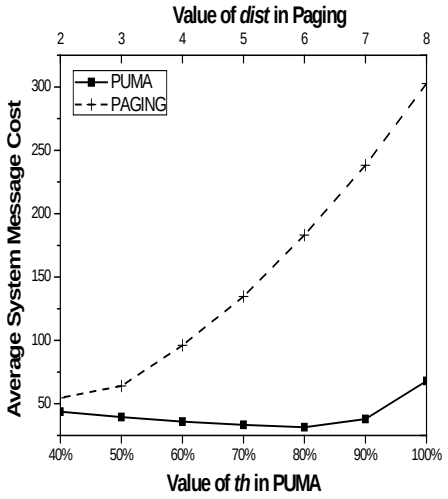
(a) I_{event} = 2 unit times



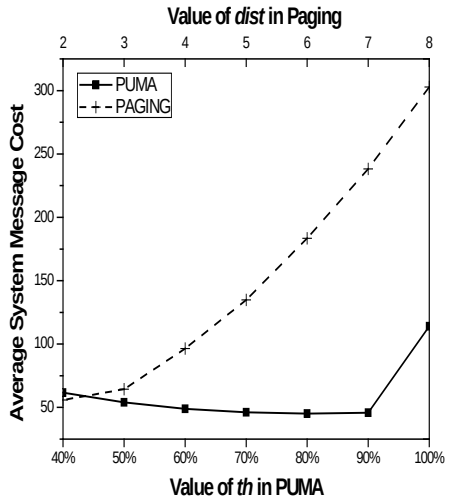
(b) I_{event} = 3 unit times



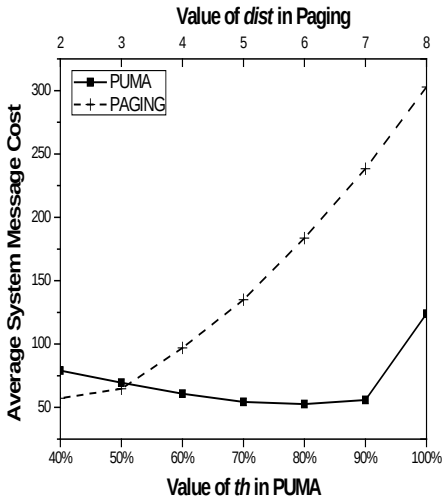
(c) I_{event} = 4 unit times



(d) I_{event} = 5 unit times



(e) I_{event} = 6 unit times

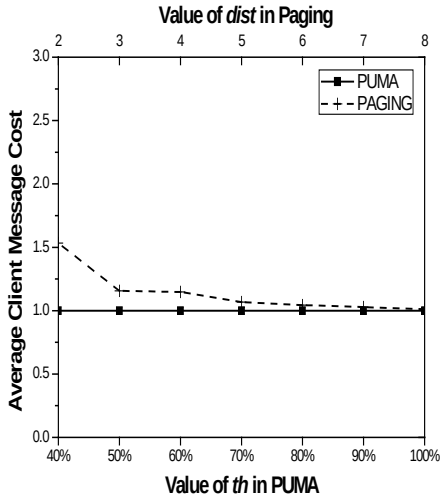


(f) I_{event} = 7 unit times

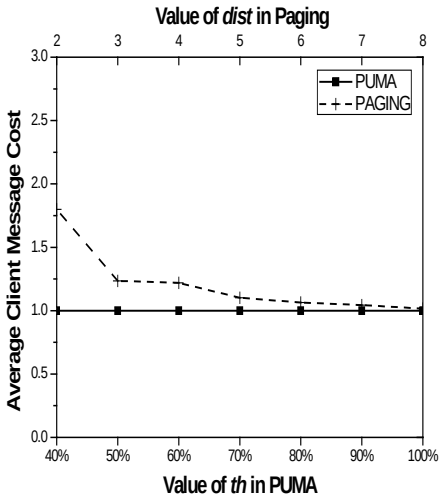
Fig. 5.3 the impact of the value of *th* to the average system message cost of PUMA with different I_{event}

From Fig. 5.3, we can find that PUMA have a very attractive performance advantage in comparison with the paging algorithm. This is because, unlike the paging algorithm, PUMA need not the proactive location reports sent by the client. Thus, a large amount of message cost can be reduced. In addition, when the arrival interval of the event is large, the performance advantage is more obvious. This is because, using the paging algorithm, a client still need to send the reports even without any new event. We can also find that the optimal performance of PUMA are closely relevant to the value of *th*. More specifically, when the value of I_{event} is larger,

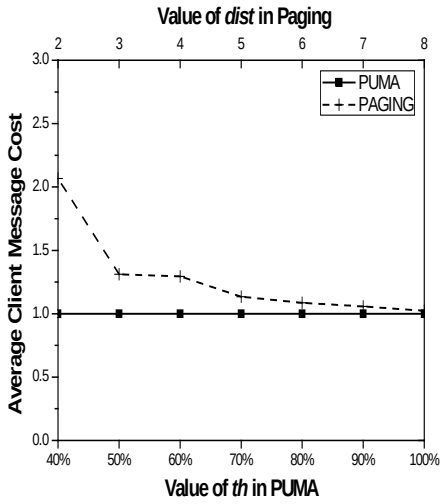
the optimal value of *th* should become larger. This is because the number of the grids, where the client may appear, become larger, when the value of *th* is large and the broker can obtain an estimation with low accuracy. On the contrary, when the value of I_{event} is small, the broker can obtain a more accurate estimation. Thus, the value of *th* can be lower. Considering an extreme case where the value of *th* is 100%, the broker will disseminate the received events to all of the grid where the client may appear. Obviously, along with the increase of the value of I_{event}, the number of such grids are raises. Thus, when the arrival interval of the events is very large, the dissemination with the *th* value of 100% should not be used.



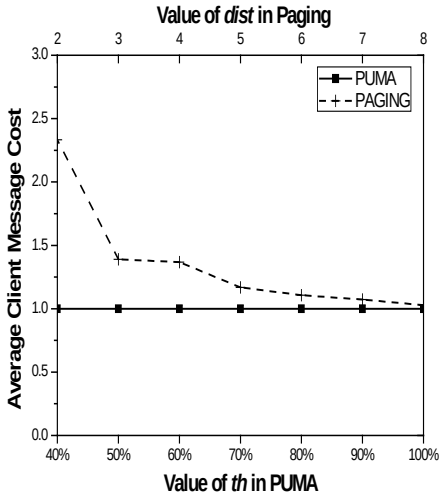
(a) I_{event} = 2 unit times



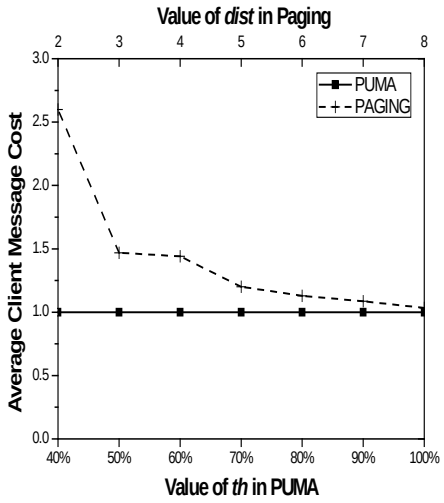
(b) I_{event} = 3 unit times



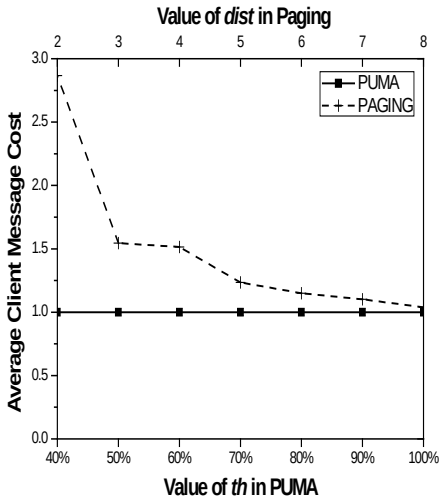
(c) I_{event} = 4 unit times



(d) I_{event} = 5 unit times



(e) I_{event} = 6 unit times



(f) I_{event} = 7 unit times

Fig. 5.4 the impact of the value of *th* to the average client message cost of PUMA with different I_{event}

Fig. 5.4 shows the impact of the value of *th* to the performance of PUMA when the arrival interval of events is 2, 3, 4, 5, 6, and 7 unit times, respectively. From all sub-figures, we can find that the average client message cost of PUMA just keeps one. In comparison with the paging algorithm, PUMA has lower average client message cost with different system parameters. More specifically, PUMA can outperform the paging algorithm up to 30%, 44%, 50%, 57%, 62%, and 65%, when

the arrival interval of events is 2, 3, 4, 5, 6, and 7 unit times, respectively.

From the figures, we can find that a client using PUMA will send just one message only when receiving one event. This allows a client to reduce the message cost as much as possible and a large amount of energy can be saved accordingly. On the contrary, a client using paging algorithm need keep reporting its locations, even without any newly arrived events. This causes a large amount of client message cost and wastes much energy of the clients. In addition, considering Fig. 5.3 and Fig. 5.4 simultaneously, we can find that, using the paging algorithm, we can find that, to achieve low system message cost, a client need more frequently report its location and then, the events broadcasting can be conducted in a smaller region. If a client wants to reduce its own message cost to save the energy, the system needs to afford higher cost. This because the broadcast region will become larger and the number of base stations involved will also become larger. Thus, there is a trade-off between the system cost and the client cost. However, PUMA can effectively reduce the system cost and client cost simultaneously and thus, can achieve a very attractive performance.

In addition, we can see that different intervals has different impact to performance of the protocols in terms of system and client message cost. When the interval is short, the system and client message costs certainly will become larger for delivering more events in a certain duration. However, because PUMA does not involve any proactive report of clients, the client message cost per event caused by PUMA will not change. Thus, the performance advantages of PUMA will change with different arrival intervals of events. The threshold value *th* is determined by the settings of the parameters. We believe that according to different parameters, such as the region size, the arrival interval of events, the mobility model of clients, etc, the optimal

threshold value should be different. Thus, in practice, the application system should follow the system parameters to determine the optimal threshold value. As we have mentioned in Section 5.2, PUMA can use different client mobility model to predict the mobility status of client. Obviously, if the mobility model can provide higher accuracy, PUMA can achieve larger performance advantages.

5.5 Summary

In this chapter, we studied the problem of how to support mobile clients in Pub/Sub systems for WMNs in a reliable and efficient manner. We proposed the first event delivery protocol for mobile clients in WMNs by using the location prediction. Our protocol, namely PUMA, can guarantee the reliability of the event delivery, and meanwhile, achieve low both system message cost and client message cost, which can effectively save the bandwidth in the WMNs and the energy of the mobile clients. We conducted extensive simulations to evaluate the performance of PUMA, and also compared it with the paging algorithms, which have been used in cellular networks for clients' mobility management. The simulation results demonstrate the performance advantages of PUMA. Compared with the paging algorithms, PUMA can save up to 91% and 65% of average system message cost and average client message cost, respectively.

Chapter 6

GIANT: An Efficient Pub/Sub Protocol for Event Delivery in DTNs

In this chapter, we describe the proposed efficient publish/subscribe protocol for event delivery in delay-tolerant networks. This chapter is organized as follows. Section 6.1 briefly introduces this work. Section 6.2 discusses the system model and basic assumptions. Section 6.3 introduces the data structures and message types. Section 6.4 describes the GIANT protocol in details. Section 6.5 presents our simulation results. Section 6.6 summarizes this chapter.

6.1 Overview

Delay-tolerant network (DTN) is an emerging networking paradigm that can provide opportunistic wireless communication services over partitioned networks[FAL03][ZHA06-1]. In a DTN, mobile nodes, also called message ferries, are recruited to deliver messages in a controlled carry-move-forward manner. There are many applications of DTNs, including inter-planetary satellite networks[LEG05], vehicle-based message ferries[SET06], and Data-muling[JEA05] in wireless sensor networks.

Like in other kinds of networks, event service in DTNs can provide support for applications in DTNs to asynchronously exchange data among communication participants. Pub/sub protocols are designed to implement the event subscription and delivery operations. Pub/sub[BAN99][CAR00][EUG03] is an event service

paradigm that has been widely used in traditional distributed system to provide support for users to disseminate and collect data using the simple publish and subscribe operations. However, the design and implementation of Pub/Sub protocols for DTNs face new challenges. The first challenging issue is *Reliability.* In DTNs, the links on a delivery path are intermittently connected or even do not exist contemporarily. Existing protocols designed for traditional distributed systems just deliver events along connected paths and cannot be used in DTNs.

The second challenging issue is *Resource Consumption.* Message ferries are an important resource in a DTN because they are usually specially designed mobile objects with limited storage space. To guarantee the network performance, a DTN usually needs to employ sufficient number of message ferries to deliver messages. However, with the increase of the network scale, a DTN usually needs to recruit more and more ferries and thus, causes the rapid increase of resource consumptions.

The third challenging issue is *Delivery Delay.* Since a DTN is frequently partitioned, message ferries cannot immediately forward the received events. They usually need to carry the events temporarily and move around. Only when broken links have been recovered, other suitable links have been discovered, or the destination of events has been reached, the ferries can forward the events. Such a carrying-and-moving fashion of message delivery often incurs high delivery delay.

In this chapter, we propose GIANT, a Geography-aIded pub/sub protocol in delAy-tolerant NeTworks, which address all the above issues. Unlike the previous solutions that employ ferries only moving in the specific directions, GIANT can recruit randomly moving ferries to deliver events without disruption of ferry mobility. This makes GIANT more flexible in supporting applications. Another

feature of GIANT is that it can guarantee user specified event delivery ratio with minimum number of ferries that are necessary. We know that recruiting multiple ferries can cause redundant event delivery and, thus, incur high resource consumption. GIANT reduces the redundancy by first constructing an event delivery tree with optimal number of hops from the publishers to the subscribers, and then using the location information of the intermediate nodes on the delivery paths to minimize the number of recruited ferries on each hop. The optimality on the number of hops is derived from the observation on our extensive experiments. Using the location information of the intermediate nodes on each hop and the information of the ferries previously recruited, GIANT can determine whether more ferries need to be recruited for event delivery. In this way, GIANT can avoid the recruiting of unnecessary ferries.

. We have conducted simulations to evaluate the performance of GIANT, and compared it with event delivery using a minimum spanning tree (MST) generated by a well-known MST algorithm. Conventional solutions for establishing a minimum spanning tree (MST) only consider how to reduce the total length of the tree, but may have a large number of hops on the paths from the root to the leave nodes. This will cause much more ferries to be recruited and longer delivery latency. The results show that, in terms of the important metrics of resource saving and delivery latency, GIANT recruits 30% less number of ferries and reduce delivery latency up to 50%, respectively. Therefore, GIANT not only achieves greater flexibility, but also has high reliability and high efficiency.

6.2 System Model and Assumptions

In this section, we discuss the system model and the assumptions used in the rest of the chapter. A DTN consists of multiple static nodes and mobile nodes, where the mobile nodes, also called message ferries, move in the network and can carry and forward data among the static nodes. In practical applications, the static nodes can be categorized into two classes according to the functionality of the nodes. In first class, a static node is just a single node, for example, a roadside unit in intelligent transportation systems, which need to employ the bypassing vehicles to disseminate and obtain information. The second class of static nodes can also be brokers to serve a group of client nodes, for example, the broker nodes in the intra-village networks [SET06]. Without losing generality, in this chapter, we will discuss the brokers belonging to the second class. We also assume that the static nodes have the same computing and communicating capabilities and the mobile node are homogeneous. We assume that the DTN is deployed in a rectangular region, which is divided into identical-size grids according to the wireless communication range of the static nodes. A broker can directly communicate only with the message ferries or the client in the broker's hosting grid but there is no direct communication between a mobile node and a client. We also assume that one grid can host only one static node and one static node can reside in only one grid. Therefore, there is no communication between two brokers without the aid of the message ferries. We assign a system-wide unique ID to each broker. This ID is composed of the vertical and horizontal coordinates of the grid where the broker resides. In this chapter, we define the coordinates of the grid at the lower-left corner is (0, 0). The grid (0, 1) and

(1, 0) are the immediate upper and right neighbor of grid (0, 0), respectively.

The above model is reasonable for practical applications on data dissemination in DTNs. For example, [SET06] described the Internet access service for remote villages. Since the villages far from the metropolitans have no infrastructures to access Internet, they need to recruit the shuttle buses to carry and forward data. Another example is the community networks. The houses along the streets in a community cannot directly communicate with each other. They can also recruit the bypassing vehicles to exchange data. In these examples, an Intra-village network or a local network in a building forms a sub-network in our model. The sub-networks are distributed in a certain area and cannot communicate with each other or Internet directly. Usually, a sub-network has a broker to directly communicate with the mobile nodes and recruit these mobile nodes to exchange data with other networks.

Many existing works [JAI04] [KEV03][ZHA06-1] in DTNs recruit the ferries that move instructed by specific application requirements. In contrast, in GIANT, ferries are ordinary mobile nodes that can move arbitrarily and no specific requirements are enforced on mobility of the ferries. Relaxing the assumptions on the ferries can greatly reduce the recruiting cost. In this chapter, we assume that ferries movement follows the random walker model (RWM) [CAM02], where a ferry moves from its current grid to a neighboring grid with a probability p. The value of p is equal to $1/n$, where n is the number of neighboring grids. Here, as for using RWM, we have the same consideration as mentioned in the last chapter. The mobile nodes in a DTNs usually are vehicles or moving people and they will follow the streets or roads which usually divides the cities and towns into grids. Thus, the RWM can effectively describe the properties of the networks. The candidate neighboring grids will be both vertical and horizontal neighbor grids nearest to the destination. Thus, n is 4. Also,

for the simplicity of the discussion, we assume that all ferries move with the same constant speed, i.e., in a unit time, a ferry can move from its current grid to one of its neighboring grid.

Next, we describe the model used for designing our pub/sub protocols. We assume that GIANT uses broker/client architecture, where the brokers are responsible for the subscription dissemination and event delivery and the clients are the event subscribers or publishers. In our model, we choose the proxy node in a sub-network as the broker. We refer to a broker connected with a publisher as a publisher-hosting broker (PHB) and a broker connected with a subscriber as a subscriber-hosting broker (SHB).

The pub/sub system employs advertisement-driven mechanism [CAR01], which can effectively reduce the events delivery and subscriptions dissemination in the network, especially when the publishers and the publication topics are not changed frequently. An advertisement issued by a publisher contains the publication topics, which will be discussed in the following section in details. A PHB first disseminates the advertisements from the publishers to all sub-networks. If there is a subscriber interested in the events with the topics contained in the advertisements, the corresponding SHB replies the interesting advertisements by generating and disseminating the corresponding subscriptions.

6.3 Data Structure and Message Types

In this section, we first describe the data structures used by brokers. Next, we

discuss the four types of messages used by GIANT.

A broker node needs to maintain two tables, including *filter_subscriber* and *routing* tables. As shown in Fig. 6.1, an entry in a filter-subscriber table includes 2 fields, filter and callbackinfo_list, where the callbackinfo_list records the list of the event notification information of applications which shares the same subscription and their subscription is kept in the filter field of the entry. The operations on a filter-subscriber table include appending a new entry to the table and purging the existing table, which are performed for the client's operations of subscription and un-subscription, accordingly.

Data Structure 6.1: filter_subscriber table

```
struct filter_subscriber_entry
{
   subscription filter;
   callback *callbackinfo_list[MaxNo_clients];
} filter_subscriber_table[MaxNo_entry];
```

Fig. 6.1 filter_subscriber table

An entry in the routing table on a broker node has four fields, including last_hop_list, filter, next_hop_list, and forwarded_event_list, as shown in Fig. 6.2 The last_hop_list field records the information list of the brokers from which the subscribed events will be delivered. Every entry in the last_hop_list field has four fields, including ID, status, distance, and hopcount. The ID field keeps the broker's ID. The value of status can be inactive or active, which shows that broker can deliver the event or not, respectively. The distance and hopcount field record the length and the number of hops of the event delivery path from the PHB to this broker, respectively. The filter field of the routing entry keeps the received subscription; the next_hop_list field keeps an ID list of the brokers or clients, where

the received events will be forwarded. The forwarded_event_list field is used to keep the events that have already been forwarded. An entry in forwarded_event_list has three fields, including the forward_event, forwarded_time_list, and unready_broker_list fields. The event field keeps the event cached in this broker. The forwarded_time_list field keeps the times when the event has been forwarded. The unready_broker_list field keeps the IDs of the next-hop brokers which the event has not been delivered with the specified probability.

Data Structure 6.2: routing_table

```
struct routing_entry
{
        subscription filter;
        struct broker
        {
                broker_ID ID;
                broker_Status status;
                float distance;
                integer hopcount;
        }broker last_hop_list[MaxNo_broker];
        node_ID next_hop_list[MaxNo_broker];
        struct forwarded_event
        {
                event for_event;
                sys_time forwarded_time [MaxNo_MN];
                broker_ID unready_broker_list[MaxNo_broker];
        } forwarded_event_list[MaxNo_Event];
}routing_table[MaxNo_entry].
```

Fig. 6.2 routing_table

Messages types used in GIANT includes events, subscription, unsubscription, and advertisement. An event is described by a 5-tuple.

Event = < *ID*, *Sender_ID*, *Receiver_ID*, *Topic*, *Value* >, where

- *ID* is of an event, which is composed of two parts. The first part is the ID of the broker that the publisher of the event connects to. The second part is a

sequence No. maintained by the PHB. Every time when the PHB publishes a new event, the sequence No. will be increased by 1. Thus, the ID can be used to identify the system-wide uniqueness of an event.

- *Sender_ID* uses to record the ID of the event sender of one hop on an event delivery path.
- *Receiver_ID* is the ID of the event receiver specified by the sender.
- *Topic* is the information topic described by the event, such as stock price, temperature, etc.
- *Value* is the sensory result, such as 20 dollar, 23 degree, etc. For example, an event, <temperature, 23> means that the today's temperature is 23 degree.

In GIANT, a subscription and a un-subscription are composed of the same fields. Thus, we just describe the subscription only. A subscription (sub) used by GIANT is defined by a 5-tuple:

Subscription = <*ID, Sender_ID, Receiver_ID, Flag, Filter*>, where

- *ID* is of a sub is very similar to the ID of an event and used to uniquely identify a subscription in the system.
- *Sender_ID* and *Receiver_ID* are used to record the ID of the subscription sender and the receiver of one hop in subscription delivery, respectively, where the sender specifies the receiver.
- *Flag* is set equal to 1, if this message is a subscription; otherwise, the message is a un-subscription.
- *Filter* is composed of three fields, including *Topic*, *Value*, and *Operator*. *Topic* and *Value* are the same as the counterparts in an event, respectively.

These two fields combined with *Operator* are used to obtain the interesting events specified by this subscription. For example, a filter <temperature, 25, '<' > specifies all the events, where the temperature is below than 25 degree.

A broker handles only those received events that satisfy a subscription. The satisfying means that the topic of the received event is the same as that in the filter of the satisfied subscription, and the relationship between the values of the event and the subscription satisfies the 'operator' in the filter of the subscription. For an example, event E <temperature, 23 > can be successfully matched to the subscription S, where the filter is <temperature, 25, '<'>, because the topics are same and 23 degree is lower than 25 degree. It is denoted as $E \in S$. A broker usually uses a subscription to obtain the desired events and then, forward them.

In addition, a subscription is covered by another subscription, if all of the events matched by the first subscription are also matched by the second subscription. For example, the filter of subscription S1 = <A, temperature, 20, '<'> is covered by the filter of the subscription S2 = <A, temperature, 25, '<'>, denoted $S2 \supset S1$.

An advertisement (ADV) is described by a 3-tuple:

Advertisement = <*ID*, *Topic*, *Sender* >, where

- *ID* is of an advertisement is very similar to the ID of an event or subscription and used to uniquely identify an advertisement in the system.
- *Topic* is the topic of the events generated by a publisher, such as stock price, temperature, etc.
- *Sender* is composed of three fields, including *Sender_ID*, *Distance*, and *Hopcount*. *Sender_ID* is used to record the ID of the sender of one hop in advertisement dissemination. *Distance* records the length of the part of the

advertisement delivery path from the advertisement source PHB to the sender, while *Hopcount* records the number of hops from the source PHB to the sender. Upon receiving an advertisement, a broker can use the sender information kept in the advertisement to determine whether or not take the sender as an upstream broker for event delivery.

6.4 The GIANT Protocol

In this section, we present the design of the GIANT protocol. We first informally describe the overall idea of GIANT and then describe the protocol in details, including four basic operations, namely advertising, subscribing, publishing, and unsubscribing respectively.

In GIANT, as shown in Fig. 6.3, a PHB recruits the ferries to disseminate its advertisements. The ferries move and meanwhile, carry and forward the advertisements to other brokers and also cache the IDs of the visited brokers. Upon receiving an advertisement from a ferry, a broker records the brokers visited by the ferry as candidate upstream brokers and disseminate the received advertisement into its own sub-network. Once the broker receives a subscription from a client corresponding to the received advertisement, it will select an upstream broker from the candidate ones and then, recruit ferries to forward the subscription to the upstream broker. Then, after the upstream broker receives the subscription, it will also forward the subscription to its own upstream broker via ferries. GIANT uses the above operations until the corresponding PHB receives the subscription or an intermediate broker can satisfy the subscription. Then, the paths for subscription

forwarding are the reverse paths for event delivery. A PHB will publish its events reversely along the subscription forwarding paths with the aid of the ferries. In addition, GIANT uses the RecruitFerry algorithm to guarantee the event delivery ratio specified by applications and meanwhile, minimize the number of ferries to be recruited. RecruitFerry uses the physical locations of subscribers and publishers to calculate the event delivery ratio achieved by the recruited ferries and then, can further determine whether or not to recruit more ferries to deliver events. We discuss the details of the above operations in the following sub-sections.

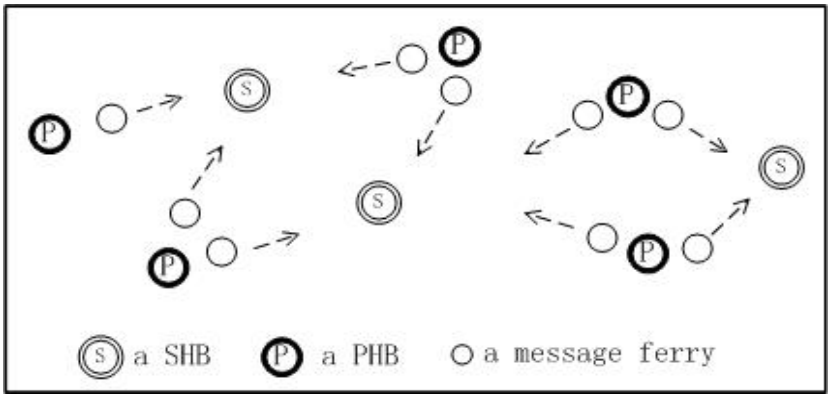


Fig. 6.3 the SHBs, PHBs, and message ferries in GIANT

6.4.1 Advertisement

In GIANT, publishers first disseminate the advertisements to whole network and then, the subscribers can subscribe the interesting events according to the advertisements. The operation of advertisement includes the following steps.

1. A publisher submits the advertisement to the sub-network broker.
2. The PHB generates an advertisement with the Sender_ID, Distance, and Hopcount equal to the *ID* of the broker, 0, and 0, respectively.
3. The PHB forwards the ferry nearby.
4. The ferry moves.
5. Upon finding a broker nearby, the ferry will forward the cached advertisement entry, *E*, in the local advertisement_list table to the found broker.
6. Upon receiving *A* from a ferry, the broker will update its local routing table. according to *A*.

Fig. 6.4 the steps in the operation of advertisement

6.4.2 Event Subscription

After receiving the advertisements, an interested subscriber will send its subscription to the sub-network broker. The broker recruits ferries to forward the subscriptions until the publisher or the suitable intermediate broker receive the subscription. The path for subscription forwards then becomes the reverse path for event delivery. The details of event subscription are as follows.

1. Upon receiving a subscription *S* from a client or a ferry, a broker will update its local *routing_table*, respectively.
2. If *S* is from a client, the broker updates its local *filter_subscriber_list* table to record the callback information of the client.
3. If the existing event subscriptions in local routing table cannot cover *S*, the broker calls *Broker_SendSub* (see Fig. 6.6) to select the upstream broker, *UP*, and forward *S* and the ID of *UP* to the ferry nearby.
4. The ferry caches the received *S* and moves around.
5. Upon finding the *UP* selected by the sender of *S*, the ferry forward the cached *S* to the broker.

Fig. 6.5 the steps in the operation of subscription

```
Function 6.1:  Broker_SendSub(S)
//for broker to handle the subscription entry received from a ferry
Input :
 S    //the subscription forwarded by a ferry
Pseudocode:
 1:   Search local routing table.
 2:   if there is a routing entry RE, s.t. RE.filter.topic = S.filter.topic and
        the status of every entry in RE.last_hop_list is inactive and
        RE.filter.value ≠ φ  then
 3:     select the brokers {B} from RE.last_hop_list with minimum value of distance;
 4:     if |B| > 1 then     //|B| obtains the number of elements in set B.
 5:       select the broker b from B with the number of hops equal to 2 or 3;
 6:     else
 7:       b ← the element in B;
 8:     end if
 9:     S.Sender_ID ← ID of this broker;
10:     S.Receiver_ID ← ID of b;
11:     forward S to a ferry nearby;
12:  end if
```

Fig. 6.6 the Broker_SendSub function

In Function 6.1, the broker needs to determine the most suitable upstream broker from the corresponding intermediate brokers. Since the ferries follow the random walk mobility model, the shortest path from the subscriber to the publisher is not the best one for event delivery. We find that both the total length and the number of hops of the delivery path can affect the number of recruited ferries and delivery latency of the event delivery on the path. In Section 6.5, we will show that the path between a subscriber and a publisher with the shortest length and only 2 or 3 hops is best delivery path.

6.4.3 Event Publication

In GIANT, the brokers will disseminate the events according to the subscriptions

from subscribers. After a publisher generates an event, it will first submit the event to the broker. Upon receiving the event, the broker will recruit the ferries to deliver the event to the brokers on the next hop, which can be obtained from the local routing table. The operation will be repeated until the subscribers receive the event. The steps of operation of event publication are shown in the following figure.

1. Upon finding a ferry nearby, a broker will call *Broker_SendEvent* function (see Fig. 6.8) to recruit the ferry and forward the events and target brokers to the ferry.
2. The ferry caches the received events.
3. The ferry moves.
4. Upon finding a target broker nearby, the ferry forwards the corresponding event *E* to the target broker.
5. Upon receiving *E*, a broker will update its local routing table and if necessary, the broker will disseminate *E* to the clients served by it.

Fig. 6.7 the steps in the operation of event publication

Function 6.2: Broker_SendEvent( )
//for broker to recruit the ferries nearby to deliver events
Pseudocode:

```
1:  Search its local routing_table.
2:  for every routing entry RE in routing_table do
3:      for every entry FE in RE.forwarded_event_list do
4:          if FE.unready_broker_list ≠ ( then
5:              for every broker id, BID, in FE.unready_broker_list do
6:                  if RecruitFerry(BID, FE.time,Prob) = false then
                            // the details of algorithm RecruitFerry is shown in Fig. 9.
7:                      FE. unready_broker_list ← FE. unready_broker_list - BID;
8:                  else
9:                      generate a new event NE;
10:                     NE.event ← FE.event;
11:                     NE.Sender_ID ← ID of this broker ;
12:                     NE.Receiver_ID ← BID;
13:                     forward NE to the mobile node;
14:                     FE.forwarded_time ← FE.forwarded_time ∪ systime;
15:                 end if
16:             end for
17:         end if
18:     end for
19: end for
```

Fig. 6.8 the Broker_SendEvent function

Algorithm 6.1: Broker_RecuitFerry(*BID, FE.time, Prob*)
// for broker to determine whether or not to recruit a ferry to deliver event
Input :
BID // the ID of the downstream broker
FE.time // the list of times when the event has been forwarded.
Prob // the target hop-by-hop reliability specified by applications or users.
Output :
RESULT // if the result is ***true***, the broker needs to forward the event; otherwise, the broker does not need to forward the event.
Pseudocode:
1: $i \leftarrow$ |the X-coordinate of *BID* – the X-coordinate of this broker's ID|;
2: $j \leftarrow$ |the Y-coordinate of *BID* – the Y-coordinate of this broker's ID|;
3: $p \leftarrow 1$;
4: **for** every entry *TE* in *FE.time* **do**
5: $T \leftarrow$ ***systime*** $- TE + i + j$;
6: Calculate the probability, p', that a random-walking ferry ever visits BID within time *T*;
// The derivation of *p'* is in *Theorem 1* .
7: $p \leftarrow p * (1- p')$;
8: **end for**
9: **if** $(1-p) > Prob$ **then**
10: $RESULT \leftarrow$ ***false***;
11: **else** $RESULT \leftarrow$ ***true***;
12: **end if**

Fig. 6.9 the RecruitFerry algorithm

In the RecruitFerry algorithm, the broker needs to calculate the probability, *p*', that a random-walking ferry ever visits grid(*i*, *j*) within time *T*. As for the *p*', we have the following theorem.

Theorem 6.1 The probability $P^T(G_{0,0}, G_{i,j})$ that a random-walking ferry starting from grid(0,0) ever visits a grid(*i*, *j*) within time *T* is as the expressions in the recursive pattern.

$P^T(G_{0,0}, G_{i,j}) = \sum_{m=i+j}^{\lfloor (T-i-j)/2 \rfloor} p_m(G_{0,0}, G_{i,j})$, where $p_{i+j}(G_{0,0}, G_{i,j}) = p^{i+j}(G_{0,0}, G_{i,j})$ and $p_{i+j+2k}(G_{0,0}, G_{i,j}) = P^{i+j+2k} - \sum_{m=0}^{k} p_{i+j+2m}(G_{0,0}, G_{i,j}) * p^{2*(k-t)}(G_{i,j}, G_{i,j})$. Here, the denotation of $\lfloor (T-i-j)/2 \rfloor$ means the largest integer no larger than (*T*-*i*-*j*)/2 and $p^t(G_{i1,j1}, G_{i2,j2})$ means the probability of a ferry starting from grid(i_1,j_1) visiting

grid(i_2,j_2) at time t.

The proof of *Theorem 1* and how to obtain the value of $p^t(G_{0,0}, G_{i,j})$ are in Appendix 6A. In theory, an intermediate broker can always guarantee the reliability of event delivery to the one-hop downstream broker, if the number of recruited ferries is sufficiently large. However, this approach is not efficient in terms of network resources, especially the memory space of the ferries. Thus, a broker needs to determine the minimum number of recruited ferries. In the Broker_SendEvent function, a broker uses the ***RecruitFerry*** algorithm (see Fig. 6.9) to determine whether or not to recruit the ferry nearby to deliver an event. The RecruitFerry algorithm uses the historical records of event delivery, including the number of delivery times and the system time of the delivery, to determine whether or not the previous event delivery can achieve the delivery ratio specified by applications. If the delivery ratio can be achieved, the ferry will not be recruited; otherwise, the ferry will be recruited.

6.4.4 Event Un-subscription

After receiving some interesting events, a subscriber maybe need not obtain the subscribed events any more. GIANT also provide the operation of un-subscription to the user to avoid deliver unsubscribed events. GIANT takes the following steps to conduct event un-subscription.

1. Upon receiving a un-subscription *US*, a broker will update its local *routing_table* accordingly.
2. If *S* is from a client, the broker removes the corresponding entry from its local *filter_subscriber_list* table.
3. The broker recruits the ferries nearby to forward *US* and the IDs of upstream brokers corresponding to *US*.
4. The ferry caches the received *US*.
5. The ferry moves.

Fig. 6.10 the steps in the operation of un-subscription

6.5 Performance Evaluation

We have conducted simulations to evaluate the performance of GIANT. In this section, we discuss the results of our simulations. We study the impact of the distance between a sender and a receiver, the number of hops in a delivery path on the performance metrics, namely number of the ferries recruited, and delivery delay, respectively. Also, we compared the performance of event delivery in the delivery tree established by GIANT with the tree established by Prime algorithm. The comparative evaluation demonstrates the performance advantages of GIANT.

6.5.1 Simulation Setup and Performance Metrics

We have conducted two groups of simulations. The first group is to show the impact of the distance between a sender and a receiver to the number of the ferries recruited. In this group of simulations, the sender is located at (0, 0), while the

receiver is at (2, 2), (3, 3), (4, 4),…, (9, 9), (10, 10). Also, we specify the target delivery ratio on these paths to be 0.5, 0.6, …, and 0.9. We test the number of recruited ferries and the delivery delay with the combinations of the different values of the parameters. Every result in our figure for the above simulations is an average of the results generated by 10000 times experiments with the same parameters.

The second group is to show the impact of the number of hops in a delivery path on the number of the ferries recruited and delivery delay. In this group of simulations, the sender is located at (0, 0), while the receiver is at (12, 12). We use the multi-hop paths between the sender and the receiver with 2, 3, 4, 6, and 8, hops, respectively, where the corresponding physical distance of every hop is 12, 8, 6, 4, and 3, sequentially. These parameters setup can ensure the total distances of all these paths are same. Also, we specify the target delivery ratio on these paths to be 0.5, 0.6, …, and 0.9 and the arriving intervals of ferries are 2, 4, 6, and 8 unit times. We test the delivery delay and the number of recruited ferries with the combinations of the different values of the parameters. Every result in our figure for the above simulations is an average of the results generated by 10000 times experiments with the same parameters.

The third group is to compare the performance of GIANT with the PRIME algorithm for minimum spanning tree establishment. Although there are existing protocols mentioned in Chapter 2, these existing protocols are not suitable for the performance simulation due to the total different network model used by these previous protocols. In this group, we use two regions of different sizes, including 10*10 and 20*20 grids, respectively. We randomly deploy 4, 5, 6, 7, 8, 9, and 10 subscribers in the smaller region and deploy 8, 10, 12, 14, 16, 18, and 20 subscribers in the larger region. We set the target delivery ratio for every subscriber to be 0.5,

0.6, 0.7, 0.8, or 0.9, randomly. In addition, we use our GIANT algorithm and the Prime algorithm [COR01] for minimum spanning tree to establish the pub/sub tree, respectively. Then, we compare the number of the recruited ferries and the delivery delay of GIANT with the minimum spanning tree. Every result on our figures is the average value of the result of 100 times simulations with the same parameters.

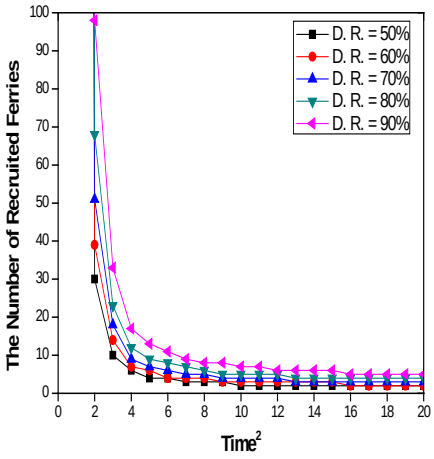
In our simulations, we use the following two metrics to evaluate the performance of our protocol.

- *Minimum Number of Recruited Ferries.* This metric is the minimum number of recruited ferries, which achieve the target delivery ratio within a certain length of duration. This metrics can show the resources cost by a pub/sub system over a DTN.
- *Delivery Delay.* This metric is the length of the duration from the sender recruiting the first ferry to the recruited ferries achieving target delivery ratio. This metric can measure the event delivery latency cost by a pub/sub system in a DTN.

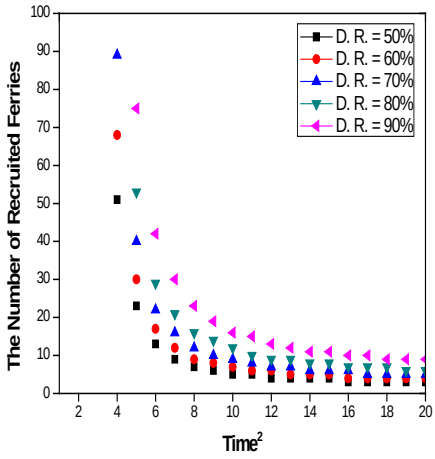
We have developed our own simulator by using C language. As we have mentioned the last two chapters, ns2 cannot efficiently support the simulation of wireless networks with a large number of nodes. In addition, ns2 has not provide the modules specifically for the simulation of DTNs. The performance of GIANT are determined by the arrival interval of ferries, the number of hops, and the target delivery ratio but not the details of underlying layers, such as MAC layer, physical layer, etc. Thus, we use our own simulator that can support the simulation of GIANT more easily and efficiently.

6.5.2 Simulation Results

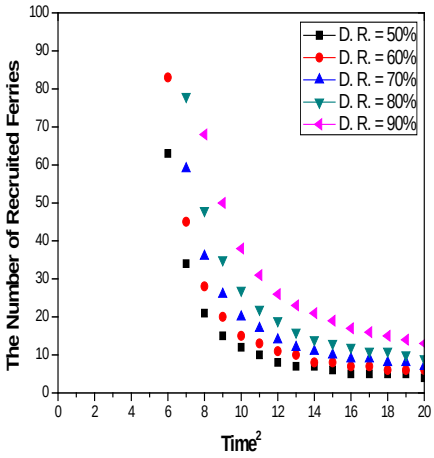
Fig. 6.11 shows the impact of the physical distance between a sender and a receiver to the minimum number of the ferries recruited to guarantee different target delivery ratio (D.R.) within different time. We can find that the minimum number of recruited ferry is obviously increasing along with the distance between the sender and the receiver. When the receiver is at (2, 2), the sender need only recruit no more than 100 ferries to guarantee the delivery ratio from 50% to 90% within 4 unit times. However, when the receiver is at (8, 8), the sender cannot guarantee the reliability high than 50% within 64 time units by recruiting even more than 100 ferries. When the time is long enough, for example 400 unit times, the number of recruited ferries can decrease to 5 and 20 to guarantee the 0.9 delivery ratio for the receivers at (2, 2) and (8, 8), respectively.



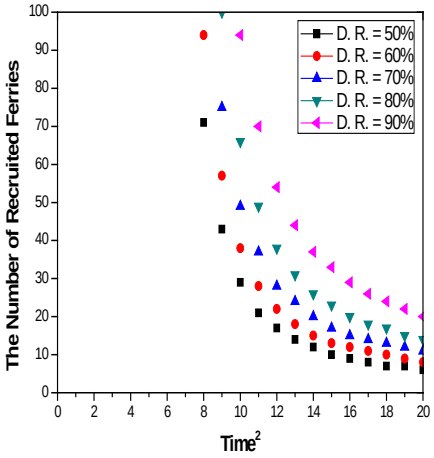
(a) the receiver at (2, 2)



(b) the receiver at (4, 4)



(c) the receiver at (6, 6)



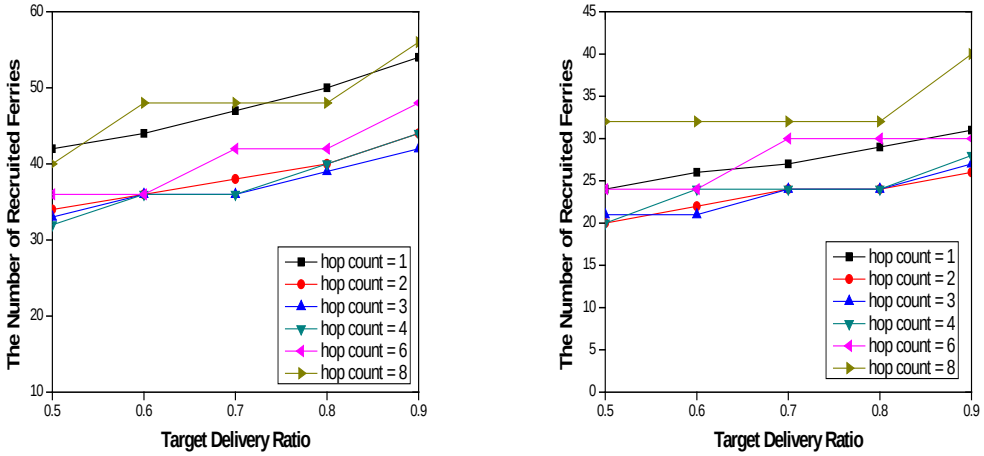
(d) the receiver at (8, 8)

Fig. 6.11 the minimum number of recruited ferries to guarantee the specified target delivery ratio between a sender and a receiver

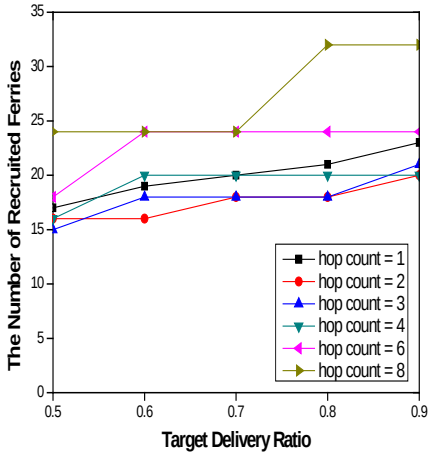
From the results shown in Fig. 6.11, we can obtain the following observations. First, the distance between a pair of sender and receiver can significantly affect the performance of event delivery between them in terms of data delivery ratio, the minimum number of recruited ferries. More specifically, within a certain temporal duration, the delivery ratio is decreased significantly along with the increase of the distance, while the number of the recruited ferries is considerably increased along with the distance. Thus, in application systems, the distance for data delivery should be as short as possible. Second, the number of recruited ferries is remarkably decreased along with increase of the delivery delay and the decrease of target delivery ratio. Thus, the application systems can guarantee the delivery ratio and meanwhile, avoid recruiting a large number of ferries by increasing delivery delay.

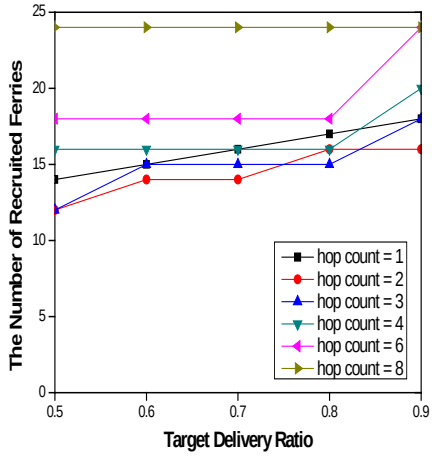
Fig. 6.12 and 6.13 show the impact of the number of hops of a delivery path to the performance of GIANT in terms of the minimum number of recruited ferries and the delivery delay. From Fig. 6.12, we can find that the sender using the delivery paths

of 2 and 3 hops can recruit smaller number of ferries than the sender using delivery paths of 1 and other numbers of hops. For example, to guarantee the reliability of 0.9, the sender needs to recruit 31, 26, 27, 28, 30, and 40 ferries with the arrival interval of 2 unit times along the paths of 1, 2, 3, 4, 6, and 8 hops, while the sender needs to recruit 18, 16, 18, 20, 24, and 24 ferries for the receiver with the arrival interval of 8 unit times along these paths, respectively. From Fig. 6.13, we can observe that the delivery paths of 2 or 3 hops can also achieve the lowest delivery delay compared with the paths of 1 hop or other number of hops. For example, to achieve the reliability of 0.9, the delivery delay is 130, 110, 105, 104, 108, and 120 unit times with the arrival interval of 2 unit times on the paths of 1, 2, 3, 4, 6, and 8 hops, respectively and the delay of delivery to the receiver are 165, 146, 147, 152, 168, and 200 unit times with the arrival interval of 8 unit times on these, respectively.



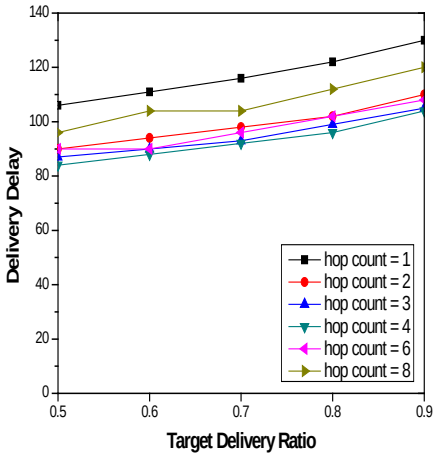
(a) ferries' arrival interval of 2 unit times (b) ferries' arrival interval of 4 unit times

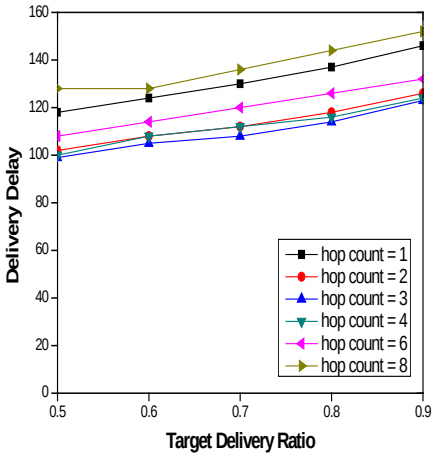




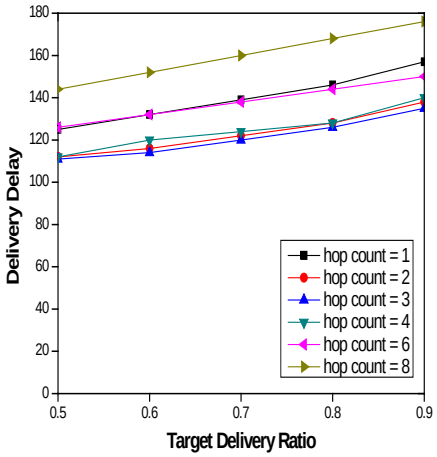
(c) ferries' arrival interval of 6 unit times (d) ferries' arrival interval of 8 unit times

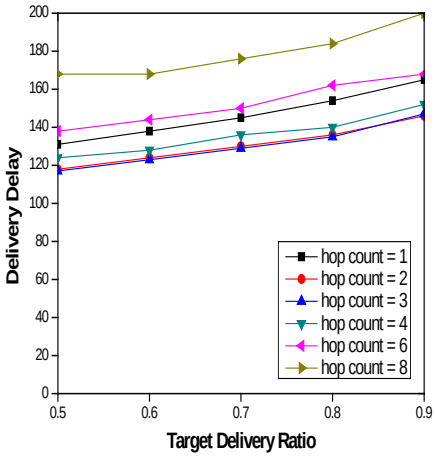
Fig. 6.12 the number of recruited ferries with different hop counts





(a) ferries' arrival interval of 2 unit times (b) ferries' arrival interval of 4 unit times





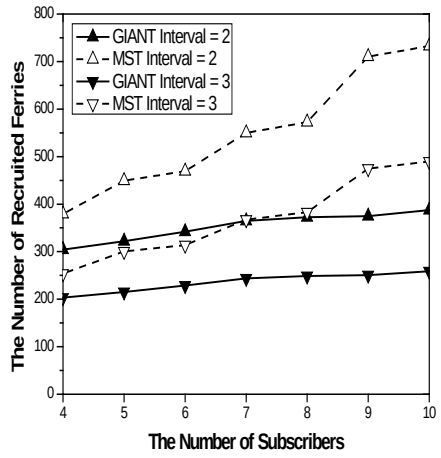
(c) ferries' arrival interval of 6 unit times (d) ferries' arrival interval of 8 unit times

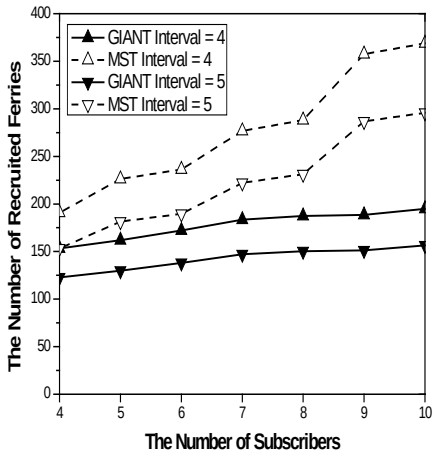
Fig. 6.13 the delivery delay in the multi-hop delivery with different hop counts

From Fig. 6.12 and 6.13, we can find that the minimum hop count can still be an important metrics for establishing the optimal event delivery path between a sender and a receiver, even in delay-tolerant networks. However, an interesting exception is that the paths of 2 or 3 hops can always achieve better performance than the paths of only single hop in terms of the number of recruited ferries and the delivery delay. Although the higher hops can decrease the average distance of the event delivery on every hop, the delivery ratio on every hop should also increase along with the hops to guarantee the delivery ratio. It is a tradeoff between the distance and the delivery ratio on every hop. When the number of hops is 2 or 3, the distance on every hop can be greatly shortened; meanwhile, the reliability on every hop has not been significantly increased. Thus, the paths of 2 or 3 hops can achieve the best performance compared with the paths of 1 hop and other larger number of hops. Thus, when there are multiple delivery paths with same length, the practical systems should select the delivery paths with few hops but avoid the paths of single hop.

Fig. 6.14 and 6.15 show the number of ferries recruited by pub/sub systems when using GIANT and the Prime algorithm for MST, respectively. From Fig. 6.14, we can observe that the number of ferries recruited by GIANT is at least 30% less than the Prime algorithm in the region of 10*10 grids. In Fig. 6.15, we can find the performance advantage of GIANT is also significant in a larger region, compared with the optimal MST algorithm. When the arrival interval of the ferries is 2, the number of the ferries recruited by GIANT is also 30% less than the Prime algorithm with different number of subscribers. More specifically, when the number of subscribers becomes 20, the number of the ferries recruited by GIANT is 70% less than the Prime algorithm. Also, performance of our protocol drops slowly along with the increase of subscribers compared with the Prime algorithm. This demonstrates a

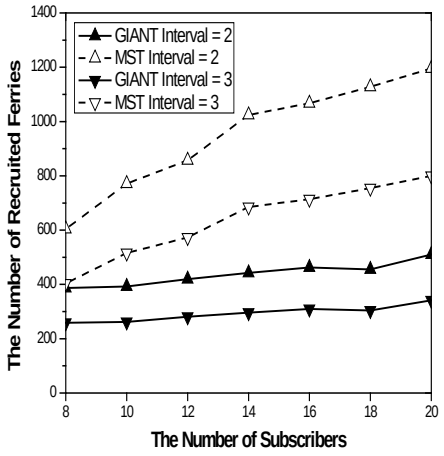
high scalability of GIANT.

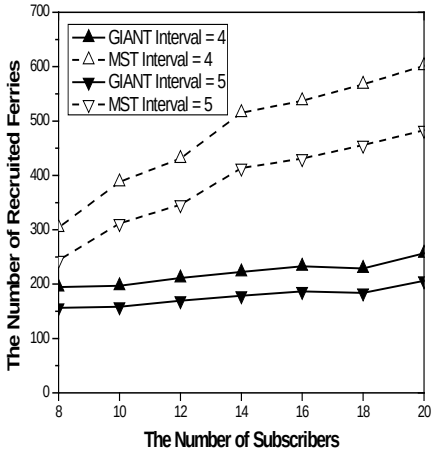




(a) arrival interval of 2, 3 unit times (b) arrival interval of 4, 5 unit times

Fig. 6.14 the number of ferries recruited by pub/sub systems in the region of 10*10 grids



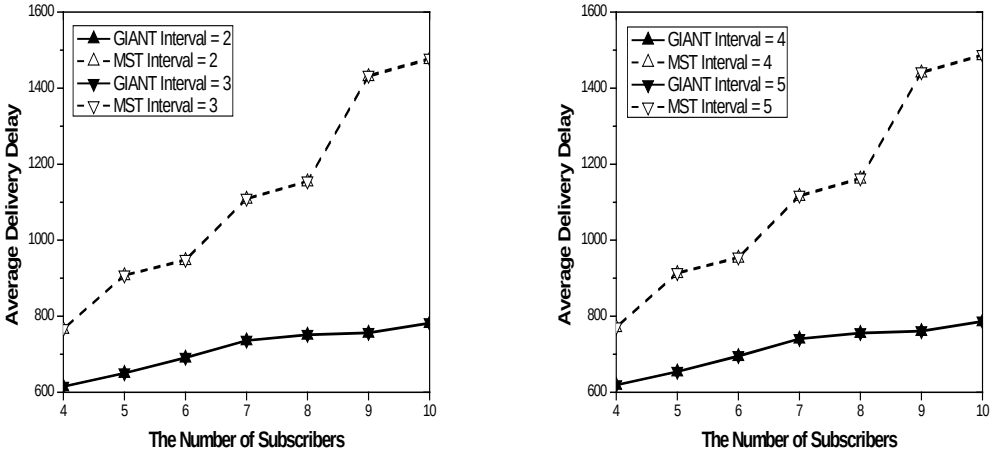


(a) arrival interval of 2, 3 unit times (b) arrival interval of 4, 5 unit times

Fig. 6.15 the number of ferries recruited by pub/sub systems in the region of 20*20 grids

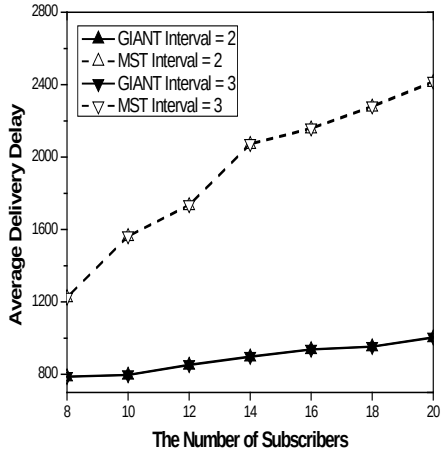
Fig. 6.16 and 6.17 show the delivery delay of the Pub/Sub systems upon using GIANT and the Prime algorithm, respectively. In Fig. 6.16, we can find that the delivery delay caused by GIANT is much lower than the Prime algorithm in the

smaller region. The delay of GIANT is at least 20% lower than the Prime algorithm with different number of subscribers and the performance advantage is up to 45% with 10 subscribers. In Fig. 6.17, the performance advantage of GIANT is more obvious in a larger region. GIANT outperforms 30% compared with the Prime algorithm with different number of subscribers. When the number of subscribers becomes 20, the delay of GIANT is less than half of the delay incurred by the Prime algorithm. Meanwhile, GIANT also shows a good scalability in terms of delivery delay in both of the regions. The delivery delay only increases less than 25%, while the number of subscribers becomes doubled from 10 to 20. However, the delay incurred by the Prime algorithm becomes almost doubled when the number of the subscribers becomes from 10 to 20. This shows the higher scalability of GIANT.

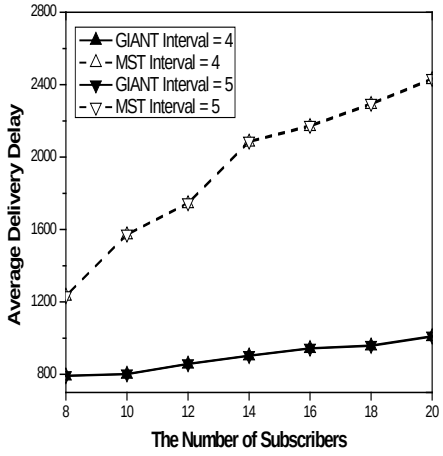


(a) arrival interval of 2, 3 unit times (b) arrival interval of 4, 5 unit times

Fig. 6.16 delivery delay of the Pub/Sub systems in the region of 10*10 grids



(a) arrival interval of 2, 3 unit times



(b) arrival interval of 4, 5 unit times

Fig. 6.17 delivery delay of the Pub/Sub systems in the region of 20*20 grids

From Fig. 6.14, 6.15, 6.16, and 6.17, we find that, to achieve identical delivery ratio, GIANT significantly outperforms the Prime algorithm for MST in term of the number of the recruited ferries and the delivery delay. Although the length of event delivery paths still greatly determines the performance of pub/sub systems for DTNs, the target event delivery ratio also generates a great impact to the performance of pub/sub for DTNs. The Prime algorithm can achieve the lowest length of total event delivery paths in the pub/sub tree. However, the algorithm also causes the delivery path with larger hop counts. Therefore, to guarantee the end-to-end delivery ratio, the delivery ratio on every hop becomes much higher. The nodes on every hop need to recruit much more ferries and also, cause much higher delivery delay. Because of the independence of every hop, the cost on every hop rises exponentially along with the number of hops. Although GIANT establishes the delivery paths with smaller hop counts, the cost on every hop can be greatly reduced. Thus, the performance can be significantly improved. In addition, if we use different mobility model to conduct simulation, we can also show the performance advantages

of GIANT. Because we use the theoretical analysis based on RWM to design the RecruitFerry algorithm, we can also conduct the corresponding theoretical analysis based on other mobility models to design the corresponding algorithm to minimize the number of ferries recruited.

6.6 Summary

In this chapter, we have proposed GIANT, a highly efficient publish/subscribe protocol for DTNs. Comparing with the previous solutions, the proposed GIANT protocol can provide more flexibility to support applications in the sense that GIANT can recruit randomly moving ferries other than the ferries moving in a specific direction to deliver the events. GIANT also can guarantee the user specified event delivery ratio by recruiting multiple message ferries. To reduce redundant event delivery caused by recruiting multiple ferries, GIANT first constructs an event delivery tree with the optimal number of hops on the paths from the publishers to the subscribers and, then, uses the location information of forwarding nodes to minimize the number of ferries recruited on each hop. Results of our extensive simulations show that, comparing with event delivery using the minimum spanning tree (MST) generated by an existing well-known MST algorithm, GIANT can recruit 30% less number of ferries and reduce delivery latency up to 50%, respectively.

Appendix 6.A

In the following part, we prove Theorem 6.1. Recall that Theorem 6.1 is

Theorem 6.1 The probability $P^T(G_{0,0}, G_{i,j})$ that a random-walking ferry starting from grid(0,0) ever visits a grid(*i*, *j*) within time *T* is as the expressions in the recursive pattern.

$P^T(G_{0,0}, G_{i,j}) = \sum_{m=i+j}^{\lfloor (T-i-j)/2 \rfloor} p_m(G_{0,0}, G_{i,j})$, where $p_{i+j}(G_{0,0}, G_{i,j}) = p^{i+j}(G_{0,0}, G_{i,j})$ and $p_{i+j+2k}(G_{0,0}, G_{i,j}) = P^{i+j+2k} - \sum_{m=0}^{k} p_{i+j+2m}(G_{0,0}, G_{i,j}) * p^{2*(k-t)}(G_{i,j}, G_{i,j})$. Here, the denotation of $\lfloor (T-i-j)/2 \rfloor$ means the largest integer no larger than (*T*-*i*-*j*)/2 and $p^t(G_{i1,j1}, G_{i2,j2})$ means the probability of a ferry starting from grid(i_1,j_1) visiting grid(i_2,j_2) at time *t*.

Proof: For the simplicity of our discussion, we first establish a rectangular coordinate, where the origin is at the location of a sending broker, for example a PHB or an intermediate broker for event delivery. The X-Axis and Y-Axis are parallel to the horizontal and vertical directions, respectively. The unit length of the coordinate is the lateral length of a grid, as shown in Fig. 6A.1. Thus, the location of broker is (0, 0). We also assume that there is a subscribing broker, which is at (*i*, *j*). We assume that *i* and *j* are larger than 0. The maximum values of *i* and *j* are *l* and there are total *n* grid. In every unit time, a ferry can only move from its current grid to any of the neighboring grids which are in the same row or column as the current grid.

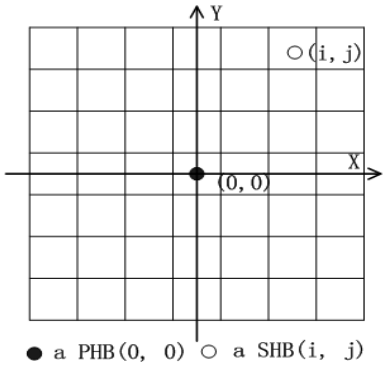


Fig.6A.1 the rectangular coordinate for Pub/Sub system

In order to determine whether to recruit a newly arrival ferry, we first need to calculate the probability $P_T(G_{0,0}, G_{i,j})$ of a ferry starting from grid(0,0) ever visiting grid(*i, j*) within time *T*, if the ferry starts from (0, 0) at time 0. Let $U_T(G_{0,0}, G_{i,j})$ denote the event of a ferry from grid(0, 0) ever visiting grid (i, j) within the temporal duration of length *T*. Then, we have

$$U_T = \bigcup_{t=0}^{T} U_t \ , \tag{6A.1}$$

where $U_t(G_{0,0}, G_{i,j})$ denotes the event that the ferry firstly visits (i, j) at the time *t*. Let $p_t(G_{0,0}, G_{i,j})$ denote the probability of a ferry from (0, 0) firstly visiting (*i, j*) at time *t*. Because the U_t s with different *t* are independent from each other, we have the following expression.

$$P_T = \sum_{t=0}^{T} p_t(G_{0,0}, G_{i,j}) \, . \tag{6A.2}$$

Recall that we have already discussed the probability calculation of the mobile client staying a certain location at time *t* in Chapter 5. Here, we repeat the same content. According to [NEL98], the random walk model of a client moving in the grids can be formulated as a *Markov Chain model*. In the markov chain model, the

probability of the initial location of a client can be description by a vector, denoted as P^0 as follows.

$$P^0 = (p_0^0, p_1^0, p_2^0, ..., p_{i*l+j}^0, ..., p_n^0), \tag{6A.3}$$

where p_{i*l+j}^0 denotes the probability of the client staying at grid(i, j) before the movement. Obviously, we have $p_0^0 = 1$, because the client starts from grid(0, 0). Also, we have the transition matrix of the random walk model as follows.

$$T = \begin{pmatrix} p_{0,0}, & p_{0,1}, & ..., & p_{0,j}, & ..., & p_{0,n} \\ p_{1,0}, & p_{1,1}, & ..., & p_{1,j}, & ..., & p_{1,n} \\ ... & ... & ... & ... & ... & ... \\ p_{i,0}, & p_{i,1}, & ..., & p_{i,j}, & ..., & p_{i,n} \\ ... & ... & ... & ... & ... & ... \\ p_{n,0}, & p_{n,1}, & ..., & p_{n,j}, & ..., & p_{n,n} \end{pmatrix}, \tag{6A.4}$$

where $p_{i,j}$ denotes the probability of the client moving from grid i to grid j during one unit time. According to the random walk model, we can easily obtain the following expressions.

$$p_{i,j} = \begin{cases} 1/4, & if\ |i-j| = 1; \\ 1/4, & if\ |i-j| = l+1; \\ 1/4, & if\ |i-j| = l-1; \\ 0, & others. \end{cases} \tag{6A.5}$$

Expression (6A.5) shows that a client has a probability of 1/4 to move from a grid to this grid's left, right, upper, and lower neighbor grids, respectively. We can obtain the probability vector, denoted as P^t which shows the probability of a client staying in every grid at time t. Here, t is an integral times of one unit time.

$$P^t = (p_0^t, p_1^t, p_2^t, ..., p_{i*l+j}^t, ..., p_n^t) = P^0 * T^t = (P_0^0, P_1^0, ..., P_n^0) * \begin{pmatrix} p_{0,0}, & p_{0,1}, & ..., & p_{0,j}, & ..., & p_{0,n} \\ p_{1,0}, & p_{1,1}, & ..., & p_{1,j}, & ..., & p_{1,n} \\ ... & ... & ... & ... & ... & ... \\ p_{i,0}, & p_{i,1}, & ..., & p_{i,j}, & ..., & p_{i,n} \\ ... & ... & ... & ... & ... & ... \\ p_{n,0}, & p_{n,1}, & ..., & p_{n,j}, & ..., & p_{n,n} \end{pmatrix}^t \quad \text{(6A.6)}$$

For the uniform format of the expressions, we denote the element of P_{i*l+j}^t in P^t to be $p^t(G_{0,0}, G_{i,j})$, which the probability of a ferry starting from grid(0,0) visiting grid(i, j) at time t.

According to [HOE72], we have the following expression.

$$P^T(G_{0,0}, G_{i,j}) = \sum_{t=0}^{T} p_t(G_{0,0}, G_{i,j}) p^{T-t}(G_{i,j}, G_{i,j}). \quad \text{(6A.7)}$$

Since a ferry can only move from its current grid to one of the directly neighboring grids in every unit time, a ferry cannot visit grid(i, j) at time (i+j-1). Thus, we have expression 6A.9 as follows.

$$\sum_{t<i+j} p_t(G_{0,0}, G_{i,j}) = 0. \quad \text{(6A.8)}$$

Obviously, the ferry can visit grid(i, j) at time i+j, and the ferry cannot visit grid(i, j) before time i+j. Thus, we have expression (6A.9).

$$p_{i+j}(G_{0,0}, G_{i,j}) = p^{i+j}(G_{0,0}, G_{i,j}) \quad \text{(6A.9)}$$

Then, according to the expression (6A.7), we have the following expression (6A.10).

$$p_{i+j+1}(G_{0,0}, G_{i,j}) = (P^{i+j+1}(G_{0,0}, G_{i,j}) - \sum_{t=0}^{i+j+1} p_t(G_{0,0}, G_{i,j}) * p^t(G_{0,0}, G_{i,j})) / p^0(G_{i,j}, G_{i,j}) \quad \text{(6A.10)}$$

As for any grid (i, j), the value of $p^0(G_{i,j}, G_{i,j})$ is 1. This is because the probability of a ferry starting from grid(i, j) and then, return to (i, j) within 0 unit time is always

1.

According to the expression (6A.8), we obtain the expression of (5A.11) as follows.

$$p_{i+j+1}(G_{0,0},G_{i,j}) = P^{i+j+1}(G_{0,0},G_{i,j}) - \sum_{k=0}^{1} p_{i+j+k}(G_{0,0},G_{i,j}) * p^{k}(G_{0,0},G_{i,j}). \qquad (6A.11).$$

According to the properties of random walk, a ferry cannot start from grid(*i*, *j*) and then return to grid(i, j) within one unit time. Thus, we can obtain

$$p_{i+j+1}(G_{0,0},G_{i,j}) = 0. \qquad (6A.12).$$

In fact, we can see values of $p_{i+j+2k-1}(G_{0,0},G_{i,j})$ are all equal to 0, where *k* is a positive integer.

In the same way, we have the expression of $p_{i+j+2k}(G_{0,0},G_{i,j})$ as follows.

$$p_{i+j+2k}(G_{0,0},G_{i,j}) = P^{i+j+2k} - \sum_{t=0}^{k} p_{i+j+2t}(G_{0,0},G_{i,j}) * p^{2*(k-t)}(G_{i,j},G_{i,j}) \qquad (6A.13)$$

Finally, we can obtain the expression of $P^{T}(G_{0,0},G_{i,j})$ as follows.

$$P^{T}(G_{0,0},G_{i,j}) = \sum_{t=i+j}^{T} p_t(G_{0,0},G_{i,j}), \text{ where } p_{i+j}(G_{0,0},G_{i,j}) = p^{i+j}(G_{0,0},G_{i,j}) \text{ and}$$

$$p_{i+j+2k}(G_{0,0},G_{i,j}) = P^{i+j+2k} - \sum_{t=0}^{k} p_{i+j+2t}(G_{0,0},G_{i,j}) * p^{2*(k-t)}(G_{i,j},G_{i,j}).$$

Thus far, we have proved the theorem 6.1, that is, we have found the formula expression for the probability of delivering an event within a duration *T*. Using this expression, a broker can determine whether or not to recruit a ferry to forward an event by using the GIANT algorithm. When a broker detects a ferry nearby, the broker will use the RecruitFerry algorithm to obtain the probability of an event being delivered to the next-hop broker before the ferry possibly earliest visits the next-hop

broker. The probability is dependent on the location of the destination broker relative to the source broker and historic times when the ferries have been used to forward the events.

Chapter 7

Conclusions and Future Works

In this chapter, we first summarize the works included in this thesis. Then, we discuss the lessons that we have learned in this research. Finally, we discuss the future work.

In this thesis, we have addressed how to design high performance pub/sub protocols for wireless ad hoc networks. Based on the literature review, we have proposed a framework for designing pub/sub protocols in wireless ad hoc networks. The framework is composed of the three orthogonal dimensions, including message dissemination, broker selection, and pub/sub initialization strategies. According to the proposed framework, we propose the design of three pub/sub protocols, namely HESPER, PUMA, and GIANT, for WSNs, WMNs, and DTNs, respectively, by using geographic information.

In wireless sensor networks, we propose *HESPER*, a Highly Efficient and Scalable Publish/subscribe protocol for wireless sEnsoR networks. HESPER uses the geographical information of the interesting events to establish the overlapping event delivery paths, which can greatly reduce the amount of event delivery and then, save the energy consumptions. Since the establishment of the delivery paths need not the coordination between subscribers, HESPER can further reduce the energy consumptions and achieve high scalability. In addition, HESPER offers two different strategies to balance the tradeoff between the subscription and publication costs to satisfy the different requirements of the practical applications.

Aiming at supporting mobile clients in WMNs, we proposed a high reliable and efficient pub/sub protocol for mobile clients in WMNs by using the location prediction. Using the location prediction for a mobile client, our protocol, namely PUMA, can deliver the interesting events to the location where a mobile client may stay. We show that the problem of delivery events to the locations with a minimum message cost, namely all-location delivery problem, is a NP-complete problem. To further reduce the message cost, PUMA just disseminate the events to the locations, namely hot locations, where a mobile client has a higher probability to stay. If the client has not received the event, PUMA will then deliver the event to the remaining location, where the client may stay. We show that the problem of delivering an event to hot locations with a minimum message cost, namely hot-location delivery problem, is also a NP-complete problem. We propose two algorithms for the both of two NP-complete problems.

In DTNs, we have proposed GIANT, a highly efficient publish/subscribe protocol. Comparing with the previous solutions, the proposed GIANT protocol can provide more flexibility to support applications in the sense that GIANT can recruit randomly moving ferries other than the ferries moving in a specific direction to deliver the events. GIANT also can guarantee the user specified event delivery ratio by recruiting multiple message ferries. To reduce redundant event delivery caused by recruiting multiple ferries, GIANT first constructs an event delivery tree with the optimal number of hops on the paths from the publishers to the subscribers and, then, uses the location information of forwarding nodes to minimize the number of ferries recruited on each hop.

Extensive performance evaluations, including theoretical analysis and experimental simulations, have been conducted to valid the performance of our

proposed protocols and algorithms. The results show the performance advantages of the protocols and algorithms.

7.1 Experiences Learned

In this study, we have addressed how to design high-performance pub/sub protocols for wireless ad hoc networks. Due to the unique characteristics of wireless ad hoc networks, the existing protocols used in wired networks and traditional mobile networks become unsuitable and unfeasible and the protocol design is very challenging. In order to design the high performance protocols in a systematic manner, we have proposed a framework for the design of pub/sub protocols in wireless ad hoc networks composed of three different dimensions, namely message dissemination, pub/sub initialization, and broker selection. We have shown that the existing protocols are the combinations of the specific strategies selected from different dimensions. We believe that thek method of establishing a framework for protocol design is very helpful to understand the existing protocols and find the remaining design space. In addition, such a method is also helpful to identify the most important issues for the performance improvement of the novel protocols. Thus, we believe that such a method can be used in the similar research on different kinds of protocol design for different networks

In addition, we have designed three high performance pub/sub protocols for wireless ad hoc networks by taking advantages of geographic information. As shown in the thesis, a node can obtain the location information of itself and other nodes in the networks. In addition, combining the location information with the temporal

information, the node can effectively derive the mobility information of the nodes in networks. Using the location information and mobility information, the node can establish routing path with very low cost and therefore, significantly save the network resources. At the present, the cost of obtaining the geographic information cannot be ignored. However, we believe that, with the development of locating technologies, the cost of obtaining location information will become very low and most commercial communication products will be equipped with positioning devices. Thus, the geographic information will become more and more important in the protocol design and network performance improvement.

7.2 Future Works

There are still some issues and topics that can be addressed in the future work. Our future works mainly includes three aspects as follows.

1) In WSNs, the energy efficient detection for composite events is still a challenging problem. Since a composite event is usually defined by several primitive events and the temporal and spatial relationships among the primitive events, the detection of such events is highly cost in terms of bandwidth and energy. In addition, more and more applications over WSNs are required the support of mobile subscribers. However, the energy and communication capability of a sensor node are very limited. Thus, how to support mobile clients is still a challenging problem.

2) In the design of PUMA, we considered only one client. In a real application,

there are maybe several clients sharing the same interest simultaneously, we will consider handle these clients simultaneously. I believe that, using geographic information, the performance of the new protocol for handling multiple clients simultaneously can be further improved compared with PUMA for handling the clients one by one. In addition, we still need consider how to model and evaluate the impact of the imprecision of the mobility prediction on the performance of our protocol. We are also going to alleviate the effect of the imprecise prediction by designing new algorithms and protocols. We have

3) To further improve the reliability of the pub/sub protocols for DTNs, we can use messages acknowledgement in GIANT. On each hop of the event delivery path, a sender will continuously recruit the ferries to send the events until the corresponding ACK messages from the receivers have been received. However, how to efficiently and effectively send the ACK message is still a challenging problem. In addition, to further improve the performance of the pub/sub protocols, we need consider the message exchange among the ferries. A ferry can obtain messages from another one and help the message delivery.

References

[AHM07] N. Ahmed, K.Jamshaid, O. Khan. *SAFIRE: A Self-Organizing Architecture for Information Exchange between First Responders*. In 2nd IEEE Workshop on Networking Technologies for Software Define Radio Networks (SDRN), 2007.

[AKK05] K. Akkaya and M. Younis. *A Survey of Routing Protocols in Wireless Sensor Networks.* In Elsevier Ad Hoc Network Journal, 3(3): 325-349, 2005.

[AKY02] I. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci. *A survey on sensor networks*. In *IEEE Communication Magazine, 40 (8):102-114,* 2002.

[AKY05] I.F.Akyildiz, X. Wang. *A survey on wireless mesh networks*. In IEEE Communications Magazine, 43(9):S20-S30, 2005.

[BAC00] J. Bacon, K. Moody, J. Bates, R. Hayton, C. Ma, A. McNeil, O. Seidel, M. Spiteri. *Generic support for distributed applications*. In IEEE Computers Vol. 33(3): 68-76, 2000.

[BAL05] R. Baldoni, R. Beraldi, G. Cugola, M. Migliavacca, L. Querzoni. *Structure-less content-based routing in mobile ad hoc networks*. In IEEE International Conference on Pervasive Services (ICPS), 2005.

[BAN99] G. Banavar, G., T. Chandra, B. Mukherjee, J. Nagarajarao, R.E. Strom, D. C. Sturman. *An efficient multicast protocol for content-based publish-subscribe systems.* In IEEE International Conference on Distributed Computing Systems (ICDCS), 1999.

[BUR04] I. Burcea, H.-A. Jacobsen, E. DeLara, V. Muthusam,and M. Petrovic. *Disconnected Operation in Publish/Subscribe Middleware*. In IEEE International Conference on Mobile Data Management (MDM), 2004.

[CAM02] T. Camp, J. Boleng, and V. Davies. *A Survey of Mobility Models for Ad Hoc Network Research*. Wireless Communication & Mobile Computing (WCMC) 2(5):483-502, 2002.

[CAR00] A. Carzaniga, D. S. Rosenblum, A. L. Wolf. *Achieving scalability and expressiveness in an Internet-scale event notification service*. In ACM symposium on Principles of distributed computing (PODC), 2000.

[CAR01] A. Carzaniga, D. S. Rosenblum, A. L. Wolf. *Design and evaluation of a wide-area event notification service.* ACM Transactions on Computer Systems (TOCS) Vol.19(3): 332-383, 2001.

[CEN03] U. Centitenmal, A. Flinder, and Y. Sun. Power-Efficient Data Dissemination in Wireless Sensor Networks. In *ACM Workshop on Data*

Engineering for Wireless and Mobile Access (MobiDE), 2003.

[CHA01] S. Chakrabarti, A.Mishra. *QoS issues in ad hoc wireless networks.* In IEEE Communications Magazine, 39(2):142-148, 2001.

[COR01] T. H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein. *Introduction to Algorithms*, 2nd Edition. MIT Press.

[COS04] P. Costa, M. Migliavacca, G. P. Picco, G. Cugola. *Epidemic Algorithms for Reliable Content-Based Publish-Subscribe: An Evaluation.* In IEEE International Conference on Distributed Computing (ICDCS), 2004.

[COS05-1] P. Costa and G. Picco. *Semi-probabilistic Content-based Publish/subscribe*. In IEEE International Conference on Distributed Computing (ICDCS), 2005.

[COS05-2] P. Costa, *at el*. *Publish-Subscribe on Sensor Network: A Semi-probabilistic Approach*. In IEEE Conference on Mobile Ad Hoc and Sensor Systems (MASS), 2005.

[CUG05] G. Cugola, J.E.M. de Cote. *On introducing location awareness in publish-subscribe middleware.* In 25th IEEE International Conference on Distributed Computing Systems Workshops (ICDCSW), 2005.

[DEY06] Deying Li, Jiannong Cao, Ming Liu, Yuan Zheng. *Construction of Optimal Data Aggregation Trees for Wireless Sensor Networks*. In IEEE International Conference on Computer Communications and Networks (ICCCN), Oct. 2006.

[EUG03] P.Th. Eugster, *at el*. *The many faces of Publish/Subscribe.* In ACM Computing Surveys, 35(2): 114 - 131, 2003.

[EUG05] P.Th. Eugster, B. Garbinato, A. Holzer. *Location-based Publish/Subscribe*. In Fourth IEEE International Symposium on Network Computing and Applications (NCA), 2005.

[FAL03] K. Fall. *A delay-tolerant network architecture for challenged internets.* In ACM conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM), 2003.

[FAR06] S. Farrell, V. Cahill. *Delay- and disruption-tolerant networking*. 1st edition, Artech House, 2006.

[FIE03] L. Fiege, F. C. Gartner, O. Kasten, and A. Zeidler. *Supporting mobility in content-based publish/subscribe middleware*. In ACM/IFIP/USENIX International Middleware Conference (Middleware), 2003.

[GAR77] M. Garey and D. Johnson. *The rectilinear Steiner tree is NP-complete.* SIAM J. Appl. Math., 32:826-834, 1977.

[HE03] T. He, C. Huang, B. M. Blum, J. A. stankovic, T. Abdelzaher, *Range-Free Localization Schemes for Large Scale Sensor Networks*, ACM International Conference on Mobile Computing and Networking (MobiCom), August, 2003.

[HEI01] J. Heidemann, F. Silva, C. Intanagonwiwat, R. Govindan, D. Estrin, and D. Ganesan. *Building Efficient Wireless Sensor Networks with Low-Level Naming*. In ACM Intl Symposium on Operating System Principles (SOSP), 2001.

[HEI02] J. Heidemann, F. Silva, D. Estrin, P. Haldar. *Diffusion Filters as a Flexible Architecture for Event Notification in Wireless Sensor Network.* Technical Report ISI-TR-556, USC/Information Sciences Institute, 2002.

[HIG01] Jeffrey Hightower, Gaetano Borriello, *Location Systems for Ubiquitous Computing*, IEEE Computer, 34(8):57-66, 2004.

[HOE72] P. Hoel, S. Port, C. Stone. *Introduction to Stochastic Processes.* 1st edition, Boston, Houghton Mifflin,1972.

[HU04] Lingxuan Hu, David Evans, *Localization for Mobile Sensor Networks*, ACM International Conference on Mobile Computing and Networking (MobiCom), September, 2004.

[ION04] M. Ionescu and I. Marsic. *Stateful Publish-Subscribe for Mobile Environments.* In ACM International Workshop on Wireless Mobile Applications and Services on WLAN Hotspots (WMASH), 2004.

[INT00] C. Intanagonwiwat, R. Govindan, D. Estrin. *Directed diffusion: A scalable and robust communication paradigm for sensor networks.* In ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom), 2000.

[JAI04] S. Jain, K. Fall, and R. Patra. Routing in a delay tolerant network. In ACM Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM), 2004.

[JEA05] D. Jea, A. A. Somasundara, and M. B. Srivastava. *Multiple controlled mobile elements (data mules) for data collection in sensor networks*. In 3rd IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS), 2005.

[JON95] C. Jones. Yachtsman's GPS handbook : a guide to the Global Positioning System of Satellite Navigation. Shrewsbury, England : Waterline Books, 1995.

[KRA96] E. Kranakis, D. Krizanc, S.S.Ravi. *On multi-label interval routing schemes*. In Oxford Computer Journal, 39:133-139, 1996.

[LEG05] J. Leguay, T. Friedman, V. Conan. *DTN routing in a mobility pattern space*. In ACM SIGCOMM workshop on Delay-tolerant networking (WDTN), 2005.

[MAL04] D. Malan, T. Fulford-Jones, M. Welsh, S. Moulton. *CodeBlue: An Ad Hoc Sensor Network Infrastructure for Emergency Medical Care*. MobiSys 2004 Workshop on Applications of Mobile Embedded Systems (WAMES), June, 2004.

[MAR02] V.J. Marathe, T. Herman. *The Design of an Interval Routing Enabled Publish/Subscribe Communications Protocol for Ad Hoc Sensor Networks*. In Midwest Society of Programming Languages and Systems Workshop, November 2002.

[MUH02] G. Muhl. *Large-Scale Content-Based Publish/Subscribe Systems.* PhD thesis. Darmstadt University of Technology, Germany. 2002.

[NEL98] R. Nelson. *Probability, Stochasitic processes, and queueing theory.* 2nd edition, Springer Verlag, 1998.

[NIC04] Dragoş Niculescu and Badri Nath, *Position and Orientation in Ad hoc Networks*, Elsevier Journal of Ad Hoc Networks, 2(2): 133-151, 2004.

[PET05] M. Petrovic, M. Vinod, H.-A. Jacobsen. *Content-based routing in mobile ad hoc networks*. In IEEE Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous), 2005.

[PIE02] P. Pietzuch, B. Shand, J. Bacon. *Hermes: A Distributed Event-Based Middleware Architecture*. In IEEE International workshop on Distributed Event-Based Systems (DEBS), 2002.

[PIT97] E. Pitoura, G. Samaras. *Data Management for Mobile Computing*.1st edition, Springer, 1997.

[POD04] I. Podnar and I. Lovrek. *Supporting Mobility with Persistent Notifications in Publish/Subscribe Systems*. In International Workshop on Distributed Event-Based Systems (DEBS), 2004.

[PRA05] K.S. Prabh, T.F. Abdelzaher, Energy-conserving data cache placement in sensor networks, In ACM Transactions on Sensor Networks (TOSN) 1(2): 178-203, 2005.

[PRI00] Nissanka B. Priyantha, Anit Chakraborty, Hari Balakrishnan, The Cricket Location-Support system, In 6th ACM International Conference on Mobile Computing and Networking (MobiCom), MA, August 2000.

[ROY99] E.M.Royer, Chai-Keong Toh. *A review of current routing protocols for ad hoc mobile wireless networks*. In IEEE Personal Communications, 6(2):46-55, 1999.

[SEG00] B. Segall, D. Arnold, J. Boot, M. Henderson, and T. Phelps. *Content based routing with elvin4*. In AUUG2K conference. 2000.

[SET06] A. Seth, D. Kroeker, M. Zaharia, S. Guo, S. Keshav. *Low-cost communication for rural internet kiosks using mechanical backhaul*. In 12th ACM International Conference on Mobile Computing and Networking (MobiCom), 2006.

[SOU05] E. Souto, *at el*. *Mires: a publish/subscribe middleware for sensor networks*. In ACM Personal and Ubiquitous Computing, 10(1): 37-44, 2005.

[SUT01] C.P. Sutton, R. Arkins, and B. Segall. *Supporting disconnectedness – Transparent information delivery for mobile and invisible computing.* In IEEE International Symposium on Cluster Computing and the Grid (CCGrid), 2001.

[TAR03] S. Tarkoma, J. Kangasharju and K. Raatikainen. *Client Mobility in Rendezvous-Notify*. In International Workshop on Distributed Event-Based Systems (DEBS), 2003.

[WAN05] J. Wang, J. Cao, and J. Li. *Support Mobile Clients in Publish/Subscribe Systems*. In IEEE International Workshop on Mobility in Peer-to-Peer Systems (MPPS), 2005.

[WU04] H. Wu, R. Fujimoto, R. Guensler, R. Hunter. MDDV: a mobility-centric data dissemination algorithm for vehicular networks. In ACM International Workshop on Vehicular Ad hoc Networks (VANET), ACM International Conference on Mobile Computing and Networking (MobiCom), September, 2004.

[YON07] E. Yoneki, P. Hui, S. Chan, J. Crowcroft. *A Socio-Aware Overlay for Publish/Subscribe Communication in Delay Tolerant Networks*. 10th ACM/IEEE International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM), 2007.

[YOU04] O. Younis, S. Fahmy. *HEED: A Hybrid, Energy-Efficient, Distributed Clustering Approach for Ad-hoc Sensor Networks*. IEEE Transactions on Mobile Computing, 3(4):366-379, 2004.

[YU01] Y. Yu, D. Estrin, R. Govindan. *Geographical and energy aware routing: a recursive data dissemination protocol for wireless sensor networks*. UCLA Computer Science Department Technical Report, UCLA-CSD TR-01-0023, 2001.

[ZHA06-1] Z. Zhang. *Routing in intermittently connected mobile ad hoc networks and delay tolerant networks: overview and challenges.* In IEEE Communication Surveys, 8(1):24-36, 2006.

[ZHA06-2] J. Zhao, G. Cao. *VADD: Vehicle-Assisted Data Delivery in Vehicular Ad Hoc Networks.* In IEEE International Conference on Computer Communications (INFOCOM), April, 2006

[ZHE07-1] Y. Zheng, J. Cao, A. Chan, K. Chan. *Sensors and Wireless Sensor Networks for Pervasive Computing Applications*. Invited paper by *Journal of Ubiquitous Computing and Intelligence*. 1(1): 17 – 34, 2007.

[ZHE07-2] Y. Zheng, J. Cao, M. Liu, J. Wang. *Efficient Event Delivery in Publish/Subscribe Systems for Wireless Mesh Networks.* In IEEE Wireless Communications and Networking Conference (WCNC), March, 2007.

[ZHE07-3] Y. Zheng, J. Cao. *Highly Scalable and Efficient Publish/Subscribe Protocols Using Geographic Information for Wireless Sensor Networks.* In IEEE Symposium on Middleware for Sensor Systems (MiSS), 3rd International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), December, 2007.