Efficient Information Discovery and Retrieval in Wireless Ad Hoc Networks

by

Zhao Cheng

A Thesis Submitted in Partial Fulfillment
of the
Requirements for the Degree
Doctor of Philosophy

Supervised by
Professor Wendi B. Heinzelman
Department of Electrical and Computer Engineering

The College
School of Engineering and Applied Sciences

University of Rochester
Rochester, New York

2006

Curriculum Vitae

Zhao Cheng attended the Radio Engineering Department at SouthEast University, Nanjing, China from 1996 to 2000 where he received his B.S. degree in 2000. He came to the University of Rochester on August 2001 and began graduate studies in Electrical and Computer Engineering. He received the Master of Science degree from the University of Rochester in 2003. He is currently working towards his Ph.D. degree in the area of wireless communications and networking. He worked with Xerox Corporation at Webster, NY during the summer of 2003. His primary research interests include wireless communications, ad hoc and sensor networks, and peer-to-peer networks.

Acknowledgement

In following my own path toward completion of the doctorate, I have learned many lessons from so many people, both in academy and in life. Among these people, I would like to particularly thank my advisor, Professor Wendi Heinzelman. I thank her for bringing me into this country and giving me the chance to grow and gain experience in a rich research environment. I thank her for giving me large freedom in my research, guiding my research in the right direction and being supportive and positive all the time. I thank her for providing me a live example on establishing healthy yet professional relations with surrounding people and achieving success on both career and family with tremendous harmony.

I would like to thank my thesis committee Professor Gaurav Sharma, Professor Shanchieh Yang for their sincere suggestions and valuable insights. I also would like to thank my colleagues at the University of Rochester. Specifically, I would like to thank Mark Perillo, Bulent Tavli, Soro Stanislava, Tolga Numanoglu, Chris Merlin and Lei Chen for their valuable help.

I also would like to thank Xerox Corporation for providing the funding for my research and providing me the opportunity to act as an intern and access the industry world. Especially, I would like to thank Dr. Naveen Sharma for his support to my research and thesis as well as being one of the member of my thesis committee.

On my road to the doctorate, I have been blessed with my friends, and I give my sincerest thanks to each of them: Jing Dong, Mark Perillo, Zhengyu Yin, Xin Lu, Weilie Yi, Xiaoxu Yi, Zunli Lu, Jian Mao, Yi Cheng, Bing Han, Wei Chai, Zuxuan Deng, Hua Wang and Vito Pandolfo.

My final thanks go to my family. Without the breeding and the education from my mother Peixiang Zhang and my father Shumin Cheng, there would be one fewer doctorate in this world. I thank my brother, Kai Cheng, who takes care of my parents and supports my study all along when I am abroad. I also thank my cousin Feng Dong and Ying Cheng and my uncle Shuren Cheng for their helping me continue my research when I was stuck in Canada for one month.

This research was made possible in part by Xerox Corporation.

Abstract

Information, in the form of data and services, pervasively resides in the vast number of nodes in wireless ad hoc networks and sensor networks. To obtain these data and services from the hosting nodes, two procedures, peer discovery and data routing, are executed. Nodes that contain desired data/services are first discovered through peer discovery. After revealing the identity of these peers, data routing transports the data/service from these peers to the requesting node. As nodes in ad hoc networks are generally constrained in resources such as energy and processing power, it is essential to maximize the efficiency of information discovery and retrieval.

While existing solutions mostly focus on improving the efficiency for specifically chosen application requirements, my thesis is that intelligent models of complex networks are needed to provide a better understanding of the general factors that contribute to efficiency, and that analyzing these models can lead to the design of much more efficient information discovery and retrieval schemes.

In the first part of the dissertation, we mathematically model ad hoc networks to find the optimal information discovery parameters such as the total number of searching attempts and the searching radius of each attempt. We first study a general scenario where nodes are uniformly distributed and targets are identical. We then study a special scenario where route caches cause nodes to be non-uniformly distributed and create non-identical targets. In the second part of the dissertation, we develop approaches to improve the efficiency of data routing. For mobile ad hoc networks, we propose a scheme that discovers routes with long lifetimes rather than random routes. For sensor networks, we provide a general data routing model to evaluate different sensor deployment strategies from the perspective of network lifetime and monetary cost. Finally, we look at a concrete peer discovery and service retrieval example by designing a smart document system using a peer-to-peer architecture.

By using the techniques developed in this dissertation, information discovery and retrieval will be much more efficient than what is possible today, enabling the realization of ad hoc networks for important applications, such as real-time audio/video, sensor networking, and peer-aware systems of devices.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Recent advances in microchip and wireless technology have boosted the research in wireless ad hoc networks to a new level. In wireless ad hoc networks, devices equipped with wireless transceivers communicate with each other and organize their own networks without the necessity of infrastructure support. These devices generally have limited energy reserves and limited processing capabilities. Bandwidth is also a scarce resource, limited by the nature of the wireless medium and the fact that this limited resource has to be shared by nearby devices. To improve resource usage efficiency, nodes usually transmit using a low power to reach a short distance, saving energy as well as enabling better channel spatial reuse. To propagate information to reach a farther distance than the transmission range, devices must forward packets for other devices. Devices cooperate with each other in this manner to allow information to spread within the network. Since information can be quickly obtained by setting up new ad hoc networks, various types of information-based applications, such as military communications, emergency communications and environment monitoring, can thus be implemented through ad hoc networks.

To retrieve information from other devices in a network, two procedures are generally executed: peer discovery and data routing. Through peer discovery, the nodes/peers that contain the desired information are first discovered. Through data routing, information is forwarded from the host nodes/peers to the requesting node. Unlike wired networks with infrastructure support, limited bandwidth and energy resources are the main challenges for ad hoc networks. Therefore, improving the overall network effi-

ciency becomes the primary design concern for ad hoc networks.

My thesis is that rather than inventing new schemes and examining their performance through experiments and simulations, it is necessary to model the peer discovery and data routing problems, which enables us to analyze the impact of the essential model factors that have the largest impact on the resolution of these problems. Since modelling cannot cover every aspect of these complex networks, we use simulation to verify the solutions and to discover how to micro-adjust the solutions to compensate for any missing factors in the model.

In this dissertation, we describe several new ideas that use the above approach to improve the efficiency of information retrieval, i.e., improving the efficiency of both the peer discovery and the data routing procedures. Since query packets, used to discover peers, are generally broadcasted by intermediate nodes, reducing redundant query retransmissions becomes the major goal of our peer discovery study. On the other hand, discovering and applying good routes during data routing, either from the individual nodes' perspective or from the global network's view, are the major goals for data routing.

1.1 Overview of Wireless Ad hoc Networks

The study of wireless ad hoc networks is not new. There has been much research by DARPA throughout the 70's and into the early 80's. In the mid 90's, interest in and research on ad hoc networks for personal or industrial applications blossomed, thanks to advances in micro-fabrication and wireless technology. Some of the applications of wireless ad hoc networks include: military communications (establish communications among soldiers for tactical operations), emergency systems (establish communications for rescue teams in disaster areas) and wireless sensor networks. Wireless sensor networks, as a branch of ad hoc networks, have attracted considerable research attention recently. In these networks, a large number of low-cost sensor devices equipped with wireless network interfaces are deployed to execute certain applications.

In general, an ad hoc network is comprised of mobile computing devices that are equipped with wireless tranceivers for communication, without support from a fixed infrastructure (such as a base station in cellular networks or access points in wireless

local area networks.) To support portability and mobility, devices tend to be battery-powered and must transmit using a minimal power and thus reach only a short distance. Data packets have to be forwarded in order to reach destination nodes that are further away. Different applications and scenarios have other different assumptions as well. For example, for mobile ad hoc networks, nodes move according to a certain pattern, while for sensor networks, nodes are generally assumed to be static in position.

There are many challenges that need to be addressed during the design of ad hoc networks. The major areas undergoing research are: Media Access Control (MAC), routing, multicasting, quality of service, self-organization, service discovery, and scalability. For different applications, different criteria may be stressed during network design. These criteria include energy efficiency, fairness, throughput efficiency and network latency.

1.2 Research Motivation

The top challenges for ad hoc network design are the limited node energy reserves and the limited communication bandwidth. Due to the ad hoc nature of this type of network, network devices are often required to be battery-operated, and thus limited in their energy supply. Bandwidth, on the other hand, is limited by communication through wireless channels in several ways. First, the bandwidth is limited by the physical operating frequency of the wireless channel. In wired networks, bandwidth can be doubled by adding another cable. This solution, however, is not feasible in the wireless territory. Second, wireless resources have to be shared by devices in the vicinity, and some resources have to be utilized to coordinate these devices. Finally, wireless communications are unstable in nature. Packet retransmission due to packet failure further reduces the already limited bandwidth. To compensate for the resource scarcity, improving efficiency unavoidably becomes the top issue for ad hoc network design.

Despite the variety of ad hoc network applications, the ultimate purpose of an ad hoc network, as with other types of networks, is to convey information. Information, which resides in devices as data and services, can be retrieved by other devices. This information retrieval procedure is generally composed of two phases: peer discovery and data routing. Through peer discovery, the devices/peers that contain the desired

information are first discovered. Data routing is responsible for finding routes between the peers and transporting the information in the form of data packets using these routes.

This dissertation focuses on improving the efficiency of information retrieval for ad hoc networks. In other words, we strive to improve the efficiency of both the peer discovery and the data routing procedures. We are especially interested in answering the following questions in terms of efficiency:

1. Peer discovery: How do we search for one specific peer that contains the desired information? How do we discover several peers that contain similar data information or services? How do we take full advantage of node caches to achieve the same goal while reducing the negative effects of stale caches?

2. Data routing: How should data be efficiently routed from the source node to the requesting node? Since mobile ad hoc networks and wireless sensor networks have different underlying networking assumptions, how should we address data routing in these networks differently?

   In mobile ad hoc networks, data routing problem is studied from the per node perspective, and we are interested in the following questions. How is a wireless link affected by node mobility, and how is a route affected correspondingly? How much data routing overhead is brought by route discovery, and how much data routing overhead is brought by excessive data forwarding? What type of routes should we look for to reduce the data routing overhead from both peer discovery and data forwarding? What is the major difference between these routes and the traditional shortest-path routes? How can we discover these routes in a distributed manner, and how close to optimal are these routes? How much routing performance improvement can we achieve using these types of routes? What are the benefits and drawbacks of these types of routes, and what tradeoff is involved to achieve the desired improvement?

   In wireless sensor networks, data routing in wireless sensor networks is essentially different from that in mobile ad hoc networks. Rather than maintaining energy efficiency for each individual node as in mobile ad hoc networks, it is more important to maintain a balance of power consumption in sensor networks so that certain nodes do not die much earlier than others, leading to unmonitored

areas in the network. Therefore, our questions are: what is a common goal of a typical sensor network? How can data routing be generalized in wireless sensor networks, and how can we determine the optimal data routing schemes? How does the solution vary for different sensor network scenarios and deployment strategies? How do we evaluate different strategies when they are all possible solutions? What should be the general evaluation terms and methodology?

3. The application perspective: During the design of an information/service discovery and retrieval system, what should be the design goals? Correspondingly, what architecture should we choose for building this system, the client/server architecture or the peer-to-peer architecture? How can we improve the efficiency of the system when abundant resources are available to choose from? How should we implement these resource management modules, and where do we store them? The answers to these questions will enable us to build better information retrieval applications.

1.3 Research Contributions

This dissertation investigates information discovery and retrieval in ad hoc networks from the efficiency perspective. We will discover general trends for common scenarios of peer discovery and data routing through modelling. Specific information retrieval cases will be studied after the factors that have the most impact are discovered and resolved. Specific contributions to the wireless ad hoc network research community include:

- We formulate the single peer discovery and multi-peer discovery problem based on general ad hoc network assumptions. We demonstrate how we can apply this model to find solutions in real applications. The optimal searching strategies for different scenarios are thereby proposed.

- We propose an adaptive local searching scheme for a popular peer discovery case: discovering a route towards a specific node when there are route caches in the network. This case is different from the previous case in that there may be false route information returned unintentionally by nodes that have stale routes

in their caches. This problem is also part of the data routing problem, and it is worth a separate study.

- For data routing in mobile ad hoc networks, we propose a distributed routing scheme that discovers long lifetime routes. We demonstrate that our routing scheme is able to achieve better performance by effectively reducing routing overhead and maintaining efficiency during data forwarding without the need for complex cross-layer designs.

- For data routing in wireless sensor networks, the goal shifts to maintaining a balance of energy in the entire network so that the network lifetime does not degrade from early energy drain in some "hot spots." We model the problem and provide a global formula that can determine the optimal data routing method for a given network deployment scenario. More importantly, we present a general methodology to evaluate different sensor deployment strategies from both the lifetime and the monetary cost perspective.

- Finally, we design a smart document system that is composed of printing devices equipped with network interfaces that can communicate with other devices in the network through a peer-to-peer architecture. During the design of this system, we reveal which criteria should be considered for this ad-hoc type network and how the design choices are made to meet these design requirements.

1.4 Dissertation Structure

Related work from the current literature is first presented in Chapter 2. Chapter 3 illustrates how we determine the optimal peer discovery strategies for general network scenarios through modelling. In Chapter 4, we propose some local searching strategies that adapt to the current route caching conditions in ad hoc network routing. Chapter 5 tackles data routing in mobile ad hoc networks by proposing a distributed routing discovery scheme that can find long lifetime routes with short route lengths. An advanced routing scheme based on these long lifetime routes is also proposed. The results show the effectiveness of our long lifetime routing scheme. In Chapter 6, we propose an

optimal data routing scheme that maximizes sensor network lifetime for various sensor network scenarios. Furthermore, we propose a general methodology for evaluating different sensor deployment strategies. The design of a smart document system based on the information retrieval structure is illustrated in Chapter 7. The dissertation is summarized in Chapter 8.

Chapter 2

Related Work

In this section, we review some of the work from current literature that is relevant to this dissertation. We start by introducing other peer discovery frameworks and methods, stressing the difference between these studies and our proposed methods. We then review existing data routing methods in both mobile ad hoc networks and static sensor networks. Since we implement a printing system based on the information retrieval and peer-to-peer (P2P) networking framework, we also include a review of P2P networking and the topics of resource management.

2.1 Peer Discovery

Peer discovery is the starting procedure for nodes to discover peers or target nodes that contain the desired information. Peer/target discovery can be divided into two branches. The first branch is to find at least one target from one or multiple targets. The most common use of this one-out-of-multi discovery is in routing protocol implementations. Typical examples are DSR [31] and AODV [52].

In DSR, when a source node needs to find a path to a destination node, it first asks its neighbors by broadcasting a query message. If any neighbor contains the route to the destination, it simply replies to the query. If the source node does not receive any response after a certain amount of time, it learns that no neighbor knows a path to the destination. Therefore, the source node floods its query message to the entire network, and eventually it discovers a valid path to the destination.

In AODV, a source node also starts its search by querying its neighbors. However, it doubles the searching radius instead of performing a network flooding if it fails to discover a route from its neighbors. This exponential increase of searching radius continues until either the destination is discovered or the maximum searching radius is reached.

The other target discovery branch is a more general case, which is to find more than one target from multiple members. Examples that require a mandatory multi-target discovery are NTP (Network Time Protocol) [43], ITTC (Intrusion Tolerance via Threshold Cryptography) [65], sensor localization [4], and sensor information collecting [46]. In NTP, three time references are required to calculate accurate time at the current location. Similarly in sensor networks, several location references are required to calculate the location of the current sensor. Other observations of the local area are also often calculated based on the data obtained from surrounding sensors. Thus, local sensor information collecting inevitably requires multiple target discovery.

Examples that require a multi-target discovery for robustness are NIS (Network Information System) [61], NTP and any application requiring auxiliary backups. Examples that require a multi-target discovery for load distribution are peer-to-peer systems [48] and distributed computing systems [57]. Depending on various application requirements, different portions out of the total targets are to be found. For NTP, only three servers are required. For temperature monitoring sensor networks, quite a few sensors are required. For peer-to-peer systems or distributed computation systems, as many as possible peers are usually preferred.

There are some target discovery problems that the previous general target discovery model cannot cover. Take DSR and AODV as an example. If the route reply is returned from the destination itself, the route contained in the route reply is the actual path to the destination. If the route reply is returned from intermediate nodes instead of the final destination, only a route cache is returned, and this cache may be stale. Although a destination can be exclusively specified by a node ID, route caches in the network make the problem more complex. Since caches are likely to provide false information when they are stale, treating this problem using a general one-out-of-multi discovery model becomes insufficient. Researchers generally resort to simulations to study the impact of route cache optimizations. We will separate the study and initiate a new

research topic on this problem.

In order to reduce the query overhead from the flood of query packets using traditional broadcasting, two alternative query propagation techniques, gossiping and random walk, have been proposed. In gossiping [38], a node forwards a query with a certain probability rather than automatically forwarding every new query it receives. Gossiping is able to achieve nearly full coverage with much less overhead compared to broadcasting. To query all the nodes, only a portion of nodes need to forward the query instead of having all the nodes repeat the query as in flooding. Some variations of gossiping have also been studied. It has been observed that normal gossiping often terminates at an early stage of propagation. One solution to avoid this is to use flooding during the initial propagation phase and use a smaller gossiping probability during later propagation. Another variation is to associate the forwarding probability with the number of neighbors. Obviously, this requires knowledge of neighbors, and the larger the number of neighbors, the smaller the propagation probability. The last variation is to associate self gossiping behavior with the neighbors' behavior. If enough neighbors have forwarded the same packet, a node must refrain from gossiping [38].

Using random walk query propagation, a node only forwards the query to one of its neighbors instead of broadcasting to all its neighbors. It has been shown that random walk does not reduce the searching cost for a moderate coverage ratio [1]. Therefore, random walk can only be used in some particular applications rather than general searching scenarios, and we do not consider random walk in our model.

In this dissertation, we study the peer discovery problem to reveal general trends and applicable strategies for most searching scenarios. Therefore, some particular searching scenarios with special requirements are not covered. First, our model deals with either proactive data dissemination schemes from data sources, such as SPIN [26], or reactive data query schemes, such as general ad hoc routing protocols. It does not cover certain hybrid data query schemes that combine both proactive and reactive components, such as SHARP [54], SPAN [8], ZRP [24], TTDD [66], rumor routing [3] and expansion ring in data replica [37]. These solutions usually require extra hardware such as GPS for topology setup. Also, these schemes have to find the balance point between the proactive and reactive components. This process either requires certain global knowledge about the data/query ratio, or it requires some complex adaptation

schemes. Second, our model only deals with one-shot queries. Although complex queries can be divided into multiple one-shot queries and solved individually using our model, it is more efficient to handle them together within one query process. For a complete overview of data querying and solutions for complex queries, the reader is referred to [58].

2.2 Route Discovery with Route Caches in Mobile Ad Hoc Networks

As mentioned previously, route discovery with route caches cannot be effectively covered by the general peer discovery study in the previous section. This is because the targets and returned peers are assumed to be identical and trust-worthy in the previous section, while routes returned from route caches may be invalid if the route cache is stale. Considering that route discovery is a very important part of data routing, we initiate a separate study on route discovery.

An on-demand routing protocol is composed of two phases, route discovery and route maintenance, both of which operate reactively and entirely on demand. In the route discovery phase, taking DSR for example, a source node floods a Route REQuest (RREQ) when it has a packet to send but has no route to the destination node. Upon receiving the RREQ packet, intermediate nodes without route caches for the target attach their addresses in the RREQ packet and continue to flood the packet. If an intermediate node has a cached route for the destination or the destination is reached by the RREQ packet, it unicasts a Route RESponse (RRES) following the reversed route back to the source node. After discovering a route and placing it in the cache table, the source node switches into the maintenance phase. In this phase, packets follow the cached route instead of initiating a new discovery process. If a packet fails to be forwarded through one of the links indicated in the source route, the intermediate node upstream of this broken link will unicast a Route ERRor (RERR) packet to the source node, indicating the broken link. The source node then removes all the route caches that contain the broken link and initiates a new route discovery process for an alternate route.

Caching strategies and caching designs for DSR are studied in [29]. The authors

achieved some optimal choices for timeout and route cache capacity through exhaustive searching over specific scenarios. In our work, we not only study the effects of caching, but also the effects of performing a local search before a global search. Furthermore, our caching strategies based on modelling can be applied to general scenarios, and we use simulations mainly for validation purposes.

Some optimizations have been proposed and have been shown to be effective in reducing stale caches and improving the performance of route caching [31]. The *salvaging* technique allows intermediate nodes to use an alternate route cache of its own when the attached route in the packet is broken. The *gratuitous route repair* technique helps intermediate nodes to remove stale caches by piggybacking the last route error information in the new RREQ packet. The *promiscuous listening* technique takes advantage of the broadcast property of the wireless medium and helps overhearing nodes to learn the network topology without directly participating in the routing process. Some other proactive optimizations are proposed in [41]. *Wider error notification* can be performed to further reduce the existence of stale caches. *Negative caches* can reside in nodes to prevent them from adding invalid caches. *Adaptive timeout selection* can avoid removing valid caches by observing route stability and dynamically choosing timeout values. These schemes, although not taken into account in our analysis and simulations, can cooperate with our routing strategies directly. Their existence only changes the caching availability conditions in the network, while our routing strategy is able to adjust itself adaptively based on the caching conditions and achieve the optimal performance.

Compared to the extensive studies on route caching, study in the searching localization area is relatively lacking. Although LAR [36] is able to localize its querying area, it requires geographical information, which we do not assume in our study. In DSR, the *non-propagating route request technique* is performed by the source node to search one-hop neighbors first before resorting to a network-wide flood. In AODV [52], an expansion ring searching scheme is proposed. A source node starts a route discovery process with an initial searching radius and increases the searching radius linearly upon each failure. A network-wide search is performed when the searching radius exceeds a predefined threshold. However, these two techniques are proposed without solid theoretical support. In [12], it is shown that when the existence of caching availability in the

network is weak (i.e., in networks with infrequent traffic), using one-hop local searching has an only insignificant savings in overhead, while the expansion ring scheme has more overhead than a simple network-wide flooding. When the existence of caching is moderate, using one-hop local searching is likely to be too conservative and a larger searching radius can reduce the routing overhead even more, as shown later in this dissertation. When route caches are abundant, one-hop local searching becomes a good scheme because there is no need to search farther for route caches in this case. Another searching localization technique is also proposed in [6]. It utilizes prior routing histories but does not take route caches into account. Our scheme also utilizes prior route histories, but in a different manner, and our work concentrates on the joint effects of the route caching and the local searching techniques rather than only one of these. Also, in contrast to the experimental study on cache validation and optimization schemes in [47], our study exposes the underlying relationship between routing overhead and caching optimization methods through quantitative analysis.

The authors in [60] studied the effects of DSR with both caching and local searching, and they mentioned the possible ineffectiveness of the expansion ring technique under weak caching existence. They compared the performance of one specific expansion ring scheme with the one-hop local searching scheme and analyzed when a node should switch from one scheme to the other. Instead, we analytically determine the optimal local searching radius among all the possible choices in different scenarios and propose a protocol to realize it. To the best of our knowledge, this is the first study on finding the optimal performance of on-demand routing protocols for general scenarios with both techniques applied. Once peers are found, the information/service must be obtained through data routing, as discussed next.

2.3 Data Routing in Mobile Ad Hoc Networks

Shortest-path routing is the most common algorithm in existing ad hoc routing protocols [31, 52]. However, as pointed out by [15], using shortest path routing is not good enough for finding stable links, even in static multi-hop wireless networks. In mobile ad hoc networks, links are even more fragile due to node mobility. A good metric to enable adaptive routing protocols, as suggested by [2], is link duration, or in other words,

link lifetime.

Several methods have been proposed for link lifetime estimation in different scenarios. In cellular systems, signal strength provides hints for node mobility patterns and probable connection loss. If nodes are equipped with GPS, link lifetime can be calculated from node distance and node speed. A theoretical link lifetime prediction method is proposed in [23]. This prediction method uses the current link age to predict the residual lifetime. The lifetime distribution for various mobility patterns, which can be used for link lifetime prediction, is achieved through experiments [42]. In our study, we do not intend to invent new methods for link lifetime estimation since extensive research already exists. Instead, we focus on designing a route discovery scheme that can easily work with other lifetime estimation methods.

The idea of finding a "good route" rather than a random route is not new. Several routing protocols based on link stability have been proposed, such as ABR [62] and SSA [20]. They both determine a link to be good if it has existed longer than a certain period. The common idea behind these approaches is to prefer stable links or strongly connected links rather than transient links during route setup. However, these protocols place too much stress on link qualities and neglect the fact that a route quality is restricted by the quality of its weakest link. Discovering routes with good quality is more difficult than discovering good quality links due to the lack of global information for discovery.

2.4 Data Routing in Sensor Networks

Data routing in sensor networks is different from that in mobile ad hoc networks. In sensor networks, sensors are deployed to achieve the same application goal rather than operating on their own. Unlike the distributed peer based traffic in mobile ad hoc networks, the traffic model of a sensor network is likely to be converge-cast to the same base station, especially when the application is for monitoring purposes. This characteristic determines that the efficiency study of a sensor network should be performed from the network perspective rather than from the node perspective. Since energy imbalance due to the many-to-one traffic pattern causes the nodes close to the base station to drain their energy much faster than other nodes, the major issue is to find a data rout-

ing scheme to balance the energy for the entire network and prevent premature energy depletion around the base station.

Early work assumed that forwarding data packets towards a data sink over many short hops is more energy-efficient than forwarding over a few long hops, due to the nature of wireless communication. The problem of setting transmission power to a minimal level that will allow a network to remain connected has been considered in several studies [53, 56]. Later, others noted that because of the electronics overhead involved in transmitting packets, there exists an optimal non-zero transmission range, at which power efficiency is maximized [9, 18]. The goal of these studies is to find a fixed network-wide transmission range for data routing. However, using such schemes may result in extremely unbalanced energy consumption among the nodes in sensor networks characterized by many-to-one traffic patterns. Therefore, in [28], the authors attempt to optimize data routing for each sensor by adjusting an individual sensor's transmission range in a many-to-one wireless sensor network. In [10], the authors attempt to optimize data routing by assigning different amount of energy to nodes at different locations.

Other schemes such as mobile data sinks and clustering have also been proposed to circumvent the design of complex wireless transceivers. The use of mobile data sinks for wireless sensor networks has previously been proposed in [66], [59], [35], [33]. The basic idea of these schemes is to achieve better energy balance by allowing a base station to move within the network. Another approach to solve the hot spot problem is clustering, an approach that is used in [25, 39, 68]. With a clustering architecture, data aggregation can be performed at the cluster heads to reduce the energy consumed from radio transmissions.

However, these strategies mostly focus on specifically chosen application requirements. There lacks certain cross-comparisons among these strategies for situations when multiple options are available. In our study, we fill this void by proposing general models and terms, not only to investigate the performance of each individual strategy, but also to provide a practical sensor deployment evaluation method from both an energy-efficient and a cost-efficient perspective.

2.5 P2P and Resource Management

To obtain experience and insight into a full ad hoc information discovery and retrieval system, we have designed a smart document system, which is comprised of a large number of document machines equipped with networking capabilities. Based on the design requirements of such a system, we select a peer-to-peer (P2P) structure where machines form clusters and allocate resources in an autonomous manner. A P2P system has many features that match those of the smart document system. These features include resource aggregation, reliability, ad hoc collaboration, transparency, increased autonomy and the ability to exploit heterogeneity. To simplify the integration of the resource management module with the rest of the system, however, we utilize some centralized concepts to facilitate the decision-making procedure, and thus we design a hybrid P2P system. The idea of incorporating centralized concepts in a P2P system is not new. A complete review on the design of a P2P structure can be found in [44].

Once the available resources are determined by the elected cluster heads, how to allocate these resources properly to complete the task becomes the next research topic. We categorize the problem into a job shop problem [14] and solve it correspondingly. In this work, we focus on setting up the framework for future expansions. Therefore, we choose linear programming as the general algorithm for the resource allocation module. Through the implementation of such a system, we obtain a solid understanding of the procedures involved in information retrieval and dissemination.

Chapter 3

Peer/Target Discovery

Retrieving information from other devices and disseminating information are the ultimate goals of wireless networks. Before information can be propagated into the network, the target peers, from which information is retrieved or to which information is disseminated, first need to be discovered. This target discovery problem has extensive applications in wireless ad hoc networks and sensor networks, such as route discovery in several routing protocols [31, 52], sensor discovery in wireless sensor networks [30], and service discovery in wireless ad hoc networks [64]. Usually, query packets are propagated inside the network to search for the targets. The target nodes will respond upon receiving the query packets. Unlike most unicasting traffic, the query process usually involves a costly flooding process. Therefore, choosing a proper searching strategy is crucial in reducing the searching overhead.

The most straightforward searching strategy is to search the entire interested area only once. This strategy is usually considered over-simplified and some other solutions are proposed to reduce searching overhead. In DSR [31], one-hop neighbors are first queried and the entire area is searched if the target is not among the one-hop neighbors. In AODV [52], an exponential expansion ring scheme is applied, which is to start searching from one hop and increase the searching radius exponentially upon each failure. However, a comprehensive study on these searching strategies is lacking. Specifically, under what network conditions is one scheme preferred over the other schemes? Is there any other scheme that outperforms the one-hop and exponential expansion ring schemes?

The answers to these questions vary for different searching requirements. When there is only one available target, we call the problem a single target problem. When there are multiple available targets, we call the problem either a one-out-of-multi target problem or a multi-out-of-multi target problem, depending on the required number of targets.

All these types of target discovery exist pervasively in ad hoc networks. The single-target discovery process is oriented for unique information such as the node ID in routing protocols [31, 52] or a unique service provided by a specific service provider [30, 64]. The single-target discovery problem can easily turn into a one-out-of-multi target discovery problem, e.g., when intermediate nodes have route caches for the required node ID, or when there are several service providers offering the same services. Multi-out-of-multi target discovery is also necessary for many applications to function. For example, in NTP (Network Time Protocol) [43], the three closest servers are needed to synchronize a node's clock. In sensor networks, a node may need to find out the hop-distance to the nearest k anchors in order to perform location estimation [4]. Also in sensor networks, a mobile host may need to collect, say 20, temperature samples from the nearby sensors to have an accurate overview of the local temperature situation. There may be some other applications that perform multi-target discovery in order to distribute the load evenly throughout the network. For example, in a peer-to-peer file sharing network [48], a peer may locate a number of nearby peers and distribute the load among them. Another example is to discover an ensemble of special nodes nearby to distribute the computation among them. Distributing data to multiple sinks is another example for sensor networks. Also, multi-target discovery may be intentionally performed for robustness. A simple example is to locate more service providers than necessary. When the primary service provider cannot function well, there will be some backup to take the place to avoid interruption without initializing another search. For security sensitive applications such as NTP [43] and NIS (Network Information System) [61], multiple-target discovery is almost a necessity, both for security and robustness concerns.

Despite the extensive existence and importance of the target discovery problem in wireless networks, the study of this field is almost non-existent. The schemes being used are merely from intuition without analytical support. This chapter fills this gap by

generalizing the problem and solving it both analytically and experimentally. Practical conclusions and suggestions on searching strategies are revealed.

3.1 Multi-target Discovery in Infinite Networks: Modeling and Algorithms

3.1.1 Problem modeling, assumptions and terminology

We assume a large number of nodes are placed randomly and independently in a two-dimensional space $\mathbb{R}^2$. A source node wants to find at least one target within a unit area of interest. Suppose that m targets are distributed uniformly within this unit area. Our question is: what is the optimal scheme to search this unit area with minimum cost? In other words, how many searching attempts n should be performed and what should be the searching area set $\mathcal{A}^{(n)} = \{A_1, A_2, \cdots, A_n\}$ for these n searching attempts?

Using this model, the searching strategies mentioned earlier can be exclusively expressed by $\mathcal{A}^{(n)}$. For example, the simplest searching strategy, which is to search the entire interested area only once, can be expressed as $\mathcal{A}^{(1)} = \{1\}$. The DSR searching strategy, which is to query the one-hop neighbors first and then search the entire area, can be expressed as $\mathcal{A}^{(2)} = \{\frac{1}{M^2}, 1\}$ if we denote M as the maximum hop limit allowed. For the exponential expansion ring scheme applied in AODV, the parameter set becomes $\mathcal{A}^{(\lceil \log_2(M) \rceil + 1)} = \{\frac{1}{M^2}, \frac{2^2}{M^2}, \frac{4^2}{M^2}, \cdots, \frac{(2^{\lceil \log_2(M) \rceil - 1})^2}{M^2}, 1\}$ if we assume that the searching area is on the order of the searching hop squared.

Here, we define the cost as the total area that has been searched. This general assumption does not contradict the traditional cost definition as the number of transmissions. In ad hoc wireless networks, a node needs to forward packets for other nodes, and in order to search a certain area, the nodes within this area have to forward the queries. Thus, the number of query transmissions to search an area of A is proportional to A by a constant coefficient determined by the forwarding mechanism such as flooding and gossiping. Also, by defining the cost directly as the searching area, we minimize the number of variables and simplify our analysis without loss of generality. The conclusions drawn from this definition can be specified for different applications simply by mapping the area to realistic application parameters.

Also, we ignore the potential increase of the packet length and the cost it brings during packet propagation. For simplicity, we also ignore potential packet collisions, which can be effectively decreased by inserting a random delay time before forwarding. We also ignore packet loss from unreliable wireless links since a node fails to receive a packet only when all its neighbors' query forwarding fails. This is a very low probability event in well-connected networks. For example, with a packet loss probability of 30%, if three neighbors forward the same query, the probability for a node to fail to receive it is only $(0.3)^3 = 0.027$.

During our analysis, we assume we are studying a snapshot of the network and nodes are static during the analysis. However, even if nodes are mobile, there are several reasons that our analysis is still valid. First, the flooding search time is short and nodes will not move too far away. Second, since nodes are moving randomly and independently, the number of nodes in a certain region is stable and will not have adverse effects on our analysis.

The model we are going to use in this section is based on the assumption that the source node is at the center of the searching area and the searching areas are concentric circles within the unit area as shown in Fig. 3.1. This simplified model expedites our current analysis and is easy to extend for realistic small-scale networks, as we will illustrate in Section 3.2. Another assumption, that targets are uniformly distributed within the area, may be invalid for certain scenarios as well. We will discuss other possible target distributions in Section 3.3.

For quick reference, we use the term *n-ring* as a strategy that nodes attempt at most n times to discover the targets. Other notations are listed in Table 3.1.

3.1.2 Finding 1 out of m targets

Let us first look at the simplest case of multi-target discovery, finding only one target out of a total of m targets. The single-target problem can be seen as $m = 1$, and we will discuss it as a special case of the 1-out-of-m problem. Let us restate this 1-out-of-m problem briefly. Now, there are m targets distributed randomly and uniformly in the unit area. The source node located at the center wants to find at least one target from these m targets with the least cost by using the optimal n searching attempts.

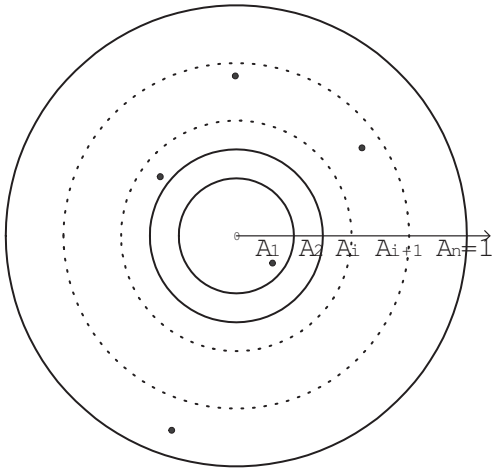

Figure 3.1: The simplified model of target discovery. The searching areas are concentric circles. Both the target nodes (black dots) and non-target nodes (not shown) are uniformly distributed in the searching area.

3.1.2.1 A two-ring approach

Suppose a two-ring approach is applied, and for the first searching attempt, the searching area is A_1. For the second searching attempt, the searching area A_2 is, of course, the entire area hence equals 1. As long as not all the m targets are located outside the A_1 area, the target will be found within the first attempt. Therefore, the probability P_1 to discover at least one target in the first attempt and the cost for the first searching attempt are

$$P_1 = 1 - (1 - A_1)^m, \qquad C_1 = A_1 \tag{3.1}$$

However, if the first attempt fails, another search has to be performed, and the total searching cost for these two searches C_2 is

$$C_2 = A_1 + A_2 = A_1 + 1 \tag{3.2}$$

Note that if a second search needs to be performed, the total cost is not only just the second searching area, but includes the cost from the previous failed searching attempt.

If a second search is required, it means that all the m targets are located in the second ring outside the A_1 area, and the probability P_2 for this case to happen is

$$P_2 = (1 - A_1)^m \tag{3.3}$$

Table 3.1: Notations used throughout this chapter.

m	the total number of targets
k	the number of targets to be found
n	the number of attempts performed
C^n	cost of an n-ring scheme
D	cost difference between two schemes
A_i	searching area of the ith attempt
$\mathcal{A}^{(n)}$	optimal searching set for n-ring search

Thus, the expected cost C^2 for a two-ring scheme to complete the 1-out-of-m target discovery is

$$\begin{aligned} C^2 &= P_1C_1 + P_2C_2 = (1-(1-A_1)^m)A_1 + (1-A_1)^m(A_1+1) \\ &= A_1 + (1-A_1)^m \end{aligned} \tag{3.4}$$

It is easy to determine the minimum C^2 for $A_1 \in [0,1]$ by solving $\frac{\partial C^2}{\partial A_1} = 0$, which results in

$$A_1 = 1 - m^{-\frac{1}{m-1}} \tag{3.5}$$

In Fig. 3.2, we show the optimal A_1 calculated from equation 3.5 for different selections of m. Also, the minimum cost calculated from equation 3.4 for the corresponding m and A_1 is shown in the bottom figure.

From this figure, we can see that when the number of existing targets m increases, the first searching area should decrease and the expected cost decreases as well. This is obvious since when more targets are available, it is more likely to find a target by searching a smaller area, resulting in a smaller cost.

3.1.2.2 An n-ring approach

To aid the expression, let us define a virtual 0th attempt search for the area of $A_0 = 0$. If the ith search attempt succeeds, the total cost C_i is simply the cost summation of the first i attempts

$$C_i = \sum_{j=1}^{i} A_j \tag{3.6}$$

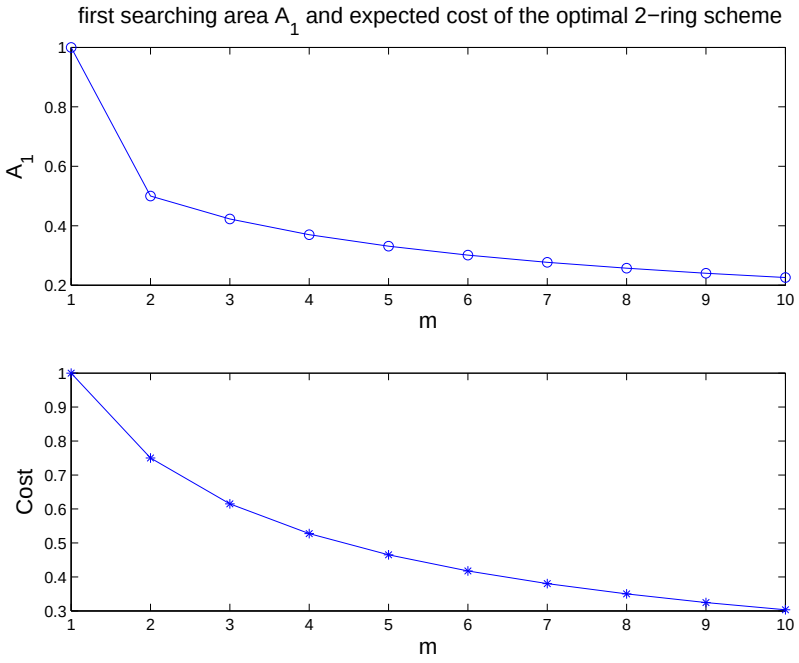

Figure 3.2: 1-out-of-m, the optimal two-ring scheme. The optimal first searching area A_1 (top graph) and the corresponding cost C (bottom graph). The values vary according to the number m of existing targets. The more targets available, the smaller the first searching area and the less expected cost are.

Similarly, in order to perform an ith search attempt and complete the task, there must be no targets in the area A_{i-1} and there must be at least one target in the area A_i. Thus, the probability P_i for the task to be completed in the ith attempt is

$$P_i = (1 - A_{i-1})^m - (1 - A_i)^m \tag{3.7}$$

Therefore, the expected cost C^n for a general n-ring searching approach is

$$\begin{aligned} C^n &= \sum_{i=1}^{n} P_i C_i = \sum_{i=1}^{n} ((1 - A_{i-1})^m - (1 - A_i)^m))(\sum_{j=1}^{i} A_j) \\ &= \sum_{i=0}^{n-1} A_{i+1}(1 - A_i)^m \end{aligned} \tag{3.8}$$

The final equality above can be easily proven through mathematical induction. Due to space constraints, we skip the intermediate steps.

3.1.2.3 Single-target discovery

The single-target discovery, as a specific case with $m = 1$, is briefly discussed here. When $m = 1$, Equation 3.4 becomes

$$C^2 = A_1 + (1 - A_1)^1 = 1 \tag{3.9}$$

The optimal cost equals 1 no matter the choice of A_1. For a general n-ring approach, it is easy to prove through mathematical induction that Eq. 3.8 is larger than 1 when $n > 2$. This means that if there is only one target, the cost of any two searching scheme is exactly the same as the cost of searching the entire area only once, and all the other searching schemes can only perform worse. Although specific ad hoc network cases may bring some cost saving as pointed out in [12], the cost saving is so negligible that the above conclusion drawn from the model still holds true.

3.1.3 Finding k out of m targets

Now, we can extend the study to a general case of finding at least k targets out of a total of m targets. Again, let us start from a two-ring approach.

3.1.3.1 A two-ring approach

Given the first searching area A_1, the probability p_i for exactly i nodes to be located within the A_1 area is a binomial distribution

$$p_i = C_m^i A_1^i (1 - A_1)^{m-i} \tag{3.10}$$

In order to find at least k nodes within the first attempt, there must be greater than or equal to k nodes within the first area A_1. The probability P_1 for this case to happen is the summation of the probabilities p_i for $i \geq k$.

$$P_1 = \sum_{i=k}^{m} p_i = \sum_{i=k}^{m} C_m^i A_1^i (1 - A_1)^{m-i} \tag{3.11}$$

The probability P_2 for the first attempt to fail is when there are less than k nodes within A_1.

$$P_2 = \sum_{i=0}^{k-1} C_m^i A_1^i (1 - A_1)^{m-i} \tag{3.12}$$

To simplify the expression, we define

$$I(p; m, k) = \sum_{i=k}^{m} C_m^i p^i (1 - p)^{m-i} \tag{3.13}$$

For a given (m, k) pair, we further simplify $I(p; m, k)$ as $I(p)$.

Eventually, we can write the cost for a two-ring searching scheme in a simpler form

$$\begin{aligned} C^2 &= P_1 C_1 + P_2 C_2 = I(A_1) A_1 + (1 - I(A_1))(A_1 + 1) \\ &= 1 + A_1 - I(A_1) \end{aligned} \tag{3.14}$$

3.1.3.2 An n-ring approach

In order to find k targets in the ith searching attempt, there must be more than k targets within the area A_i. Also, there must be fewer than k targets within the area A_{i-1}, or else the search would end in the $(i-1)$th attempt. The probability P_i for the ith search to complete the searching task is

$$P_i = I(A_i) - I(A_{i-1}) \tag{3.15}$$

The cost of the ith search, similar to the 1-out-of-m case, is

$$C_i = \sum_{j=1}^{i} A_j \tag{3.16}$$

Thus, we have the expected cost for a general n-ring search

$$\begin{aligned} C^n &= \sum_{i=0}^{n} P_i C_i = \sum_{i=0}^{n} ((I(A_i) - I(A_{i-1})(\sum_{j=1}^{i} A_j)) \\ &= \sum_{i=0}^{n-1} A_{i+1}(1 - I(A_i)) \end{aligned} \tag{3.17}$$

Now, we have the searching cost in equations 3.8 and 3.17. In the next section, we will determine how to determine the optimal searching area set $\mathcal{A}^{(n)}$ to minimize the cost.

3.1.4 Algorithms

We will examine two types of algorithms to minimize the cost depending on when the parameters are determined: pre-planned algorithms and online algorithms. For pre-planned algorithms, the entire searching set $\mathcal{A}^{(n)}$ is calculated before the first searching attempt. The source node will refer to these precalculated values during the searching process. For online algorithms, the source node only calculates the next searching area right before the search. Online algorithms need less computation than pre-planned algorithms since they only calculate when necessary. However, they may perform less than optimal due to the lack of global knowledge.

3.1.4.1 Brute force (BF)

Given n, there are $n-1$ searching area variables from A_1 to A_{n-1} (A_n is set to one). BF tries every possible combination of $A_i \in [0,1]$ and calculates the cost based on equation 3.8 or 3.17. It picks the smallest cost as the optimal cost and the corresponding area set as the optimal solution. During implementation, the interval of [0,1] for each A_i is discretized. With a granularity of δ for each dimension A_i, the computational complexity is on the order of $(\frac{1}{\delta})^{n-1}$ for an n-ring scheme.

This scheme, although simple to implement, requires excessive computation time and becomes infeasible when n increases. Also, due to discreization, the results will only be quasi-optimal. We perform BF offline just to provide a benchmark on achievable minimal cost for the other algorithms.

3.1.4.2 Ring-splitting (RS)

Since BF cannot find the optimal solution within tolerable time, especially when n increases and the granuality δ reduces, we attempt to find an alternative "good" algorithm with fewer computations. One solution is to insert a new searching ring between existing searching rings to reduce the cost as much as possible until there is no more cost reduction by inserting new rings. We implement this idea in the Ring-splitting scheme described as follows.

1. Start with the ring $[0,1]$.

2. For the nth calculation, the area set of $\{[0, a_1], [a_1, a_2], \cdots, [a_{n-1}, 1]\}$ already exists. Check all these n rings and find out the ring candidates that can be split to further reduce the cost. (We will describe how to find out the candidates right after the procedure description.)

3. Terminate if there are no more candidates. Else, go to Step 4.

4. Pick the candidate that will reduce cost the most and split it. Go back to Step 2.

Now, we discuss how we determine a ring candidate. Suppose we already have an n-ring scheme. An $(n+1)$-ring scheme can be derived from this n-ring scheme by inserting another searching attempt with searching area A_j between the ith attempt and the $(i+1)$th attempt. From equation 3.17, the cost difference D between the old n-ring scheme and the new $(n+1)$-ring scheme is

$$\begin{aligned} D &= C^n - C^{n+1} \\ &= A_{i+1}(1 - I(A_i)) - A_j(1 - I(A_i)) - A_{i+1}(1 - I(A_j)) \end{aligned} \tag{3.18}$$

Whether the ring between $[A_i, A_{i+1}]$ should be split and become a candidate is a maximization problem of D and is determined as follows.

1. By solving $\frac{\partial D}{\partial A_j} = 0$, we achieve the possible splitting point A_j. Numerical methods are required to find A_j.

2. In order to reduce cost by inserting A_j, the splitting point has to be within the ring and the cost difference should be larger than 0. Therefore, check if A_j is within $[A_k, A_{k+1}]$ first. Then, check if $D(A_j) > 0$. Only when both requirements are satisfied, should A_j be a ring splitting candidate for $[A_k, A_{k+1}]$.

Since each splitting only adds two rings for calculations in the next step, the total computation will be $2n_0 - 3$ if the algorithm stops at the n_0 ring. The number of comparisons is $i - 1$ for the i-ring scheme, and the total number of comparisons is $\sum_{i=1}^{n_0}(i-1) = \frac{n_0(n_0-1)}{2}$, which is much fewer than that of the BF scheme $(\frac{1}{\delta})^{n_0-1}$.

Although RS does not guarantee the solution to be optimal, it reduces the computation time dramatically compared with the BF scheme. Also, while BF has to calculate for each n-ring solution separately, RS is scalable to n by providing the solution for all n-rings within one sequence of calculation. The only question remaining about RS is

how its performance is compared to that of BF. We will come to this issue right after we introduce the ORS algorithm in the next section.

3.1.4.3 Online ring-splitting (ORS)

Both BF and RS are pre-planned algorithms. The optimal number of searching attempts and the entire searching area set are determined before the first search begins. ORS, instead, calculates the searching area only for the next search right before the search starts. In this algorithm, the source node always plans to finish the search within two attempts by splitting the remaining area. Upon its failure, ORS performs another splitting on the remaining unsearched area to find how far the next search should reach. This process continues until either the target is found, or there will be no more cost saving in splitting the remaining area.

ORS is very similar to RS. The only difference is that ORS can only split the remaining unsearched area, while RS can split any of the existing rings. This is because ORS is performed online, while RS is performed before the search starts and thus is able to do the global splitting.

Here is how ORS splits the remaining searching area. Suppose the source node has already searched the area of S_0 and k_0 targets have been found. The new goal is to find $k - k_0$ targets from the remaining $m - k_0$ targets in the remaining $1 - S_0$ area. If the source node plans to finish the searching within two attempts by using A as the first searching area, the new cost would be

$$\begin{aligned} C_e &= I(\frac{A - S_0}{1 - S_0}; m - k_0, k - k_0))A + (1 - I(\frac{A - S_0}{1 - S_0}; m - k_0, k - k_0))(A + 1) \\ &= 1 + A - I(\frac{A - S_0}{1 - S_0}; m - k_0, k - k_0) \end{aligned} \tag{3.19}$$

Again, some numerical methods are required to solve $\frac{\partial C_e}{\partial A} = 0$. Also, the root $\tilde{A}$ has to pass the following two checks to provide the maximum cost saving: $\tilde{A} \in [S_0, 1]$ and $C_e < 1$. If the check fails, just use $A = 1$ to finish the last searching attempt. Otherwise, use $\tilde{A}$ to perform the next search.

As can be expected, ORS performs even less than optimal compared to RS since it can only split the remaining ring. However, it requires even less computation. There is

only one computation for each additional searching attempt, and the computation stops as long as the searching goal is met.

Table 3.2: A comparison of different algorithms for finding k=1 out of $m = 3$ targets.

	Cost	n	$\mathcal{A}^{(n)}$	Computations
BF	0.560	4	{0.251,0.594,0.851,1.0}	165,667,502
RS	0.567	6	{0.111,0.422,0.746,0.926,0.988,1.0}	9
ORS	0.581	5	{0.422,0.746,0.926,0.988,1.0}	4

3.1.5 Numerical results

3.1.5.1 Algorithm evaluation

This section evaluates the performance of algorithms BF, RS and ORS. We will reveal how many searching attempts are generally enough and how these algorithms perform compared to each other.

In Fig. 3.3, the expected costs for the solution of the 1-out-of-m problem calculated by each algorithm are shown. The X-axis indicates the total number of available targets. Let us first examine the the performance of the algorithms. BF and RS have such close performance that their curves overlap with each other. ORS performs at most 5% worse than the other two algorithms. As mentioned earlier, this is because ORS is an online algorithm and lacks global knowledge. However, its performance is still very close to that of the pre-planned schemes. For the pre-planned schemes, although a different number of rings and different area parameters may be required to achieve their own optimal point (see column 3 in Table 3.2), these algorithms perform nearly identically in terms of cost (see column 2 in Table 3.2). The BF performance shown in Fig. 3.3 is on a limited brute force search on up to 4-ring schemes with a granularity of 0.001. BF uses over 165 million computations to achieve the cost of 0.560, while RS achieves a very close cost 0.567 using only 9 computations. From this view, RS is much more practical for realistic implementations.

Fig. 3.3 also reveals how many searching attempts are enough to achieve near-optimal performance. For 2-ring schemes, all the algorithms perform almost the same.

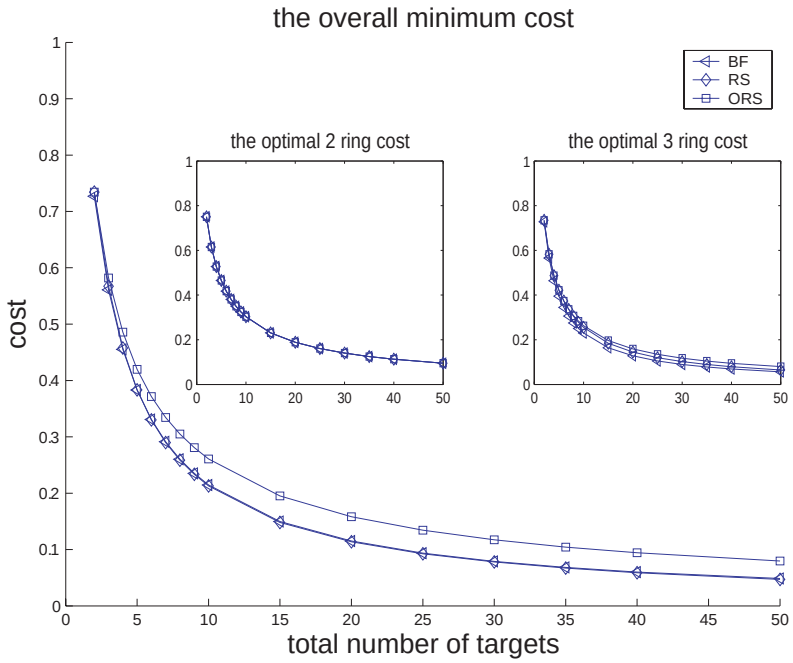

Figure 3.3: The optimal expected cost of finding one target for each algorithm and the optimal 2-ring and 3-ring cost for each algorithm. The x-axis indicates the targets available in the searching area. The y-axis indicates the expected cost. Although the number of rings n to achieve the overall optimal cost is usually larger than 3, the optimal 3-ring scheme already performs very close to the real optimal.

For 3-ring schemes, ORS performs a little worse than the pre-planned algorithms, but it is still close. Although the real optimal solution may occur at a larger value of n, the two-ring schemes have a major impact on the cost reduction compared with the 1-ring scheme whose cost is 1, and the three-ring schemes only further reduce the cost by around a trivial 2-5%. This informs us that it is very important to find a good searching area at the first attempt, and more than 3 attempts are unnecessary.

We also show the results for the k-out-of-m problem using (m,k) pairs of (6,2), (6,3), (6,4), (20,2), (20,10), (20, 18), (60,2), (60,30), (60,58). By investigating the results of these discovery requests, we can have an idea of the trend of the searching cost and the searching radius for different total numbers of targets and for cases of searching few/half/most out of these targets.

Only the results from BF and RS are shown. For ORS, after finding k_0 targets, the

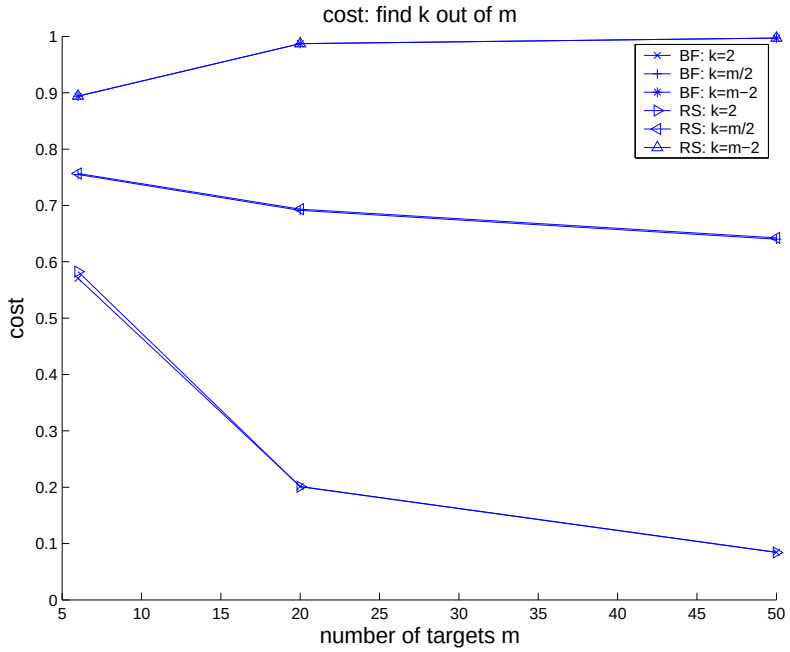

Figure 3.4: The optimal expected cost for k-out-of-m discovery by different algorithms. The x-axis indicates the targets available in the searching area. The y-axis indicates the expected cost. 2, $\frac{m}{2}$,$m-2$ targets are to be searched for each algorithm.

goal of the next search is changed to finding $k - k_0$ out of $m - k_0$. Therefore, the expected cost of ORS is dependent on each searching result; hence it is hard to determine analytically. The performance of ORS will be shown later through simulations.

As we can see from Fig. 3.4, the performance of these algorithms is still very close to each other and the curves overlap with each other. The larger the number of targets that need to be found, the less the cost can be reduced. Although the details are not shown here, the 2-ring and 3-ring schemes are still dominant in the cost reduction and more than 3-ring is unnecessary, which is the same conclusion as in the 1-out-of-m case.

In summary, the two-ring RS scheme can provide close to optimal cost performance, and the three-ring RS scheme can further reduce the cost by at most 5%. More searching attempts can only reduce the cost by a negligible amount of less than 1% and are unnecessary. When only a few number of targets are to be found, or when $k << m$, the cost saving is significant. When most of the targets are to be found, the cost is close to

the simple flooding searching scheme.

3.1.5.2 Model validation

Since our model is stochastically based, we experiment with our algorithms in a large-scale network to verify that their cost performance matches the analytical expected cost. Also, we will examine how these algorithms affect the discovery latencies compared to the one-ring searching scheme, which is not included in our analysis. Hence, we place a large number of nodes, N_T, in a disk area randomly and independently. Each node has the same transmission range of R_t and the density is large enough to form a well-connected network. The source node is located at the center of the unit area. The targets are chosen randomly from these N_T nodes, and the number of targets $m << N_T$. The source node controls its searching area by appending a searching radius limit on the query packet, and only nodes inside the distance limit will forward the query packet. Latency is defined as the round trip distance from the source node to the target. For example, for the source node to hear the response from the border, the latency is $2 \times 1 = 2$.

We experiment on 3-ring BF, 3-ring RS and 3-ring ORS using the area set obtained from analysis and record their cost and latency. The cost is compared to the expected cost of the 3-ring RS scheme from analysis. In the top row of Fig. 3.5, we show the results of the 1-out-of-m discovery, and on the bottom row, we show the results of the $\frac{m}{2}$-out-of-m discovery. In both cases, the cost of these algorithms is very close to the expected cost of 3-ring RS. This verifies our model and analysis. For latency, ORS performs a little better than the other algorithms. This is because it is more aggressive in searching a larger area and tends to take fewer attempts to complete the task. Thus, the corresponding latency is smaller.

3.2 Practical Target Discovery in Ad Hoc Networks

In the previous section, we studied the target discovery problem based on a simplified model. When applying the proposed algorithms to realistic ad hoc networks, we need to resolve several other issues. First, since hop limit is generally used to restrict query

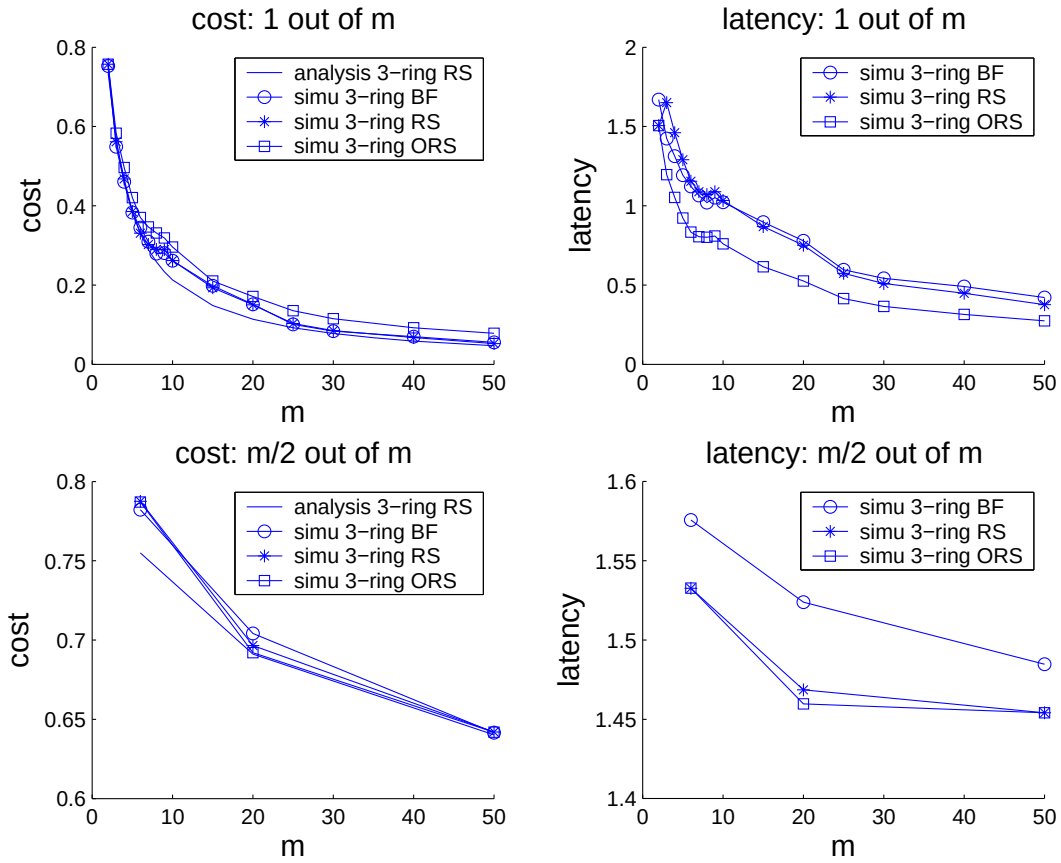

Figure 3.5: The average cost and latency performance for each algorithm for geographical scenarios. The x-axis indicates the targets available in the searching area. The top row shows the results of 1-out-of-m discovery, and the bottom row shows the results of $\frac{m}{2}$-out-of-m discovery.

flooding, we need to map the searching area set $\mathcal{A}^{(n)}$ to hop values. Second, nodes are located at different positions in the network instead of the center as assumed in our model. We want to discover how this location variation affects our model and the corresponding area-to-hop mapping. Third, since it is more likely that nodes do not know their locations, what should be the global searching hop values to save the searching cost from the network perspective? Finally, we want to determine the robustness of our algorithms under erroneously estimated network parameters.

3.2.1 Area-to-hop mapping

Note that the area A in our model indicates the portion of the total number of nodes that should be queried. The problem of mapping the area to a hop value is actually to find the hop value that can be used to cover that portion of nodes. Statistics on the number of nodes at different hop distances are required to help the mapping procedure.

In [12], we proposed an empirical estimation method to determine the number of nodes at each hop distance for connected networks. Given the total number of nodes N and the transmission range R_t, the number of nodes $\tilde{N}_{i,x_0}$ at i hops away from a source node can be estimated. Since nodes at different locations have different views at the network, x_0, which indicates the distance between the node and the border, incorporates this difference.

We show an area-to-hop mapping example for a node located at the border of the network with $x_0 = 0$ in Fig. 3.6. The network contains 1000 nodes and the transmission radius is $R_t = 0.1$. The number of nodes at each hop distance $\tilde{N}_{i,0}$ is shown in the uppermost plot, and the total number of nodes $T_{i,0}$ within hop distance i is shown in the middle plot. After we divide $T_{i,0}$ by the total number of nodes in the network, 1000 in this case, $T_{i,0}$ indicates how much portion of nodes are within hop i of this specific node. For example, about 50% of nodes are within its first 15 hops and around 75% of nodes are within its 20 hops, as shown by the dot lines in the middle plot.

Now we can consult T_i for the area-to-hop mapping. Suppose the source node needs to find 3 out of 20 targets. Using the two-ring RS scheme, we find the area set to be $\mathcal{A}^{(2)} = \{0.489045, 1.0\}$. For the first searching area 0.489045, we check T_i at each hop value i and find that $T_{15,x_0} = 0.48923$ at hop 15 is the closest to the area value 0.489045. Thus, we choose 15 as our first search hop distance. The second hop can be chosen as any integer large enough to search the entire network. By combining the node estimation method $\tilde{N}_{i,x_0}$ from [12] and the target discovery algorithm in this chapter, we can find the optimal searching hop values for each individual source node.

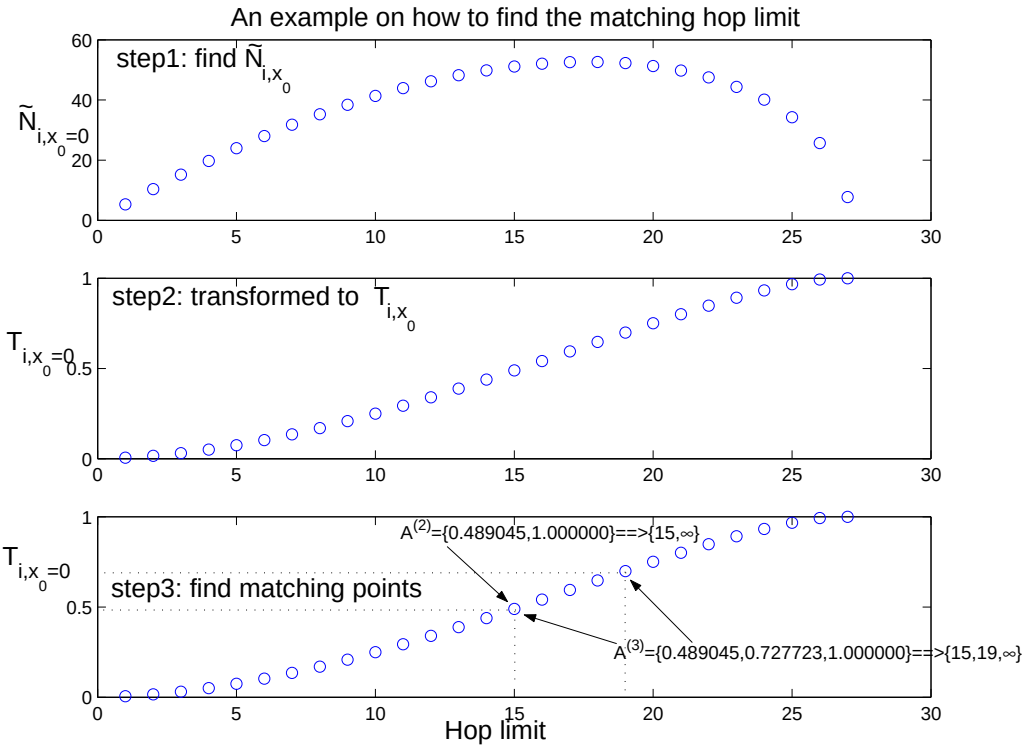

Figure 3.6: An example on how to transform calculated areas into hop limits. First, N_{i,x_0} is estimated. Second, normalized T_{i,x_0} is determined. Finally, hop limits can be found by matching the areas with T_{i,x_0}.

3.2.2 Global searching parameters

Note that given a specific target searching requirement, the area set calculated using our algorithm is always the same. The variation of node location x_0 only varies the hop choices during area-to-hop mapping by using different $\tilde{N}_{i,x_0}$ values. In other words, nodes should choose different searching hop values based on their own locations.

However, in general, nodes do not know their locations in the network and thus cannot perform $\tilde{N}_{i,x_0}$ estimation. Instead of choosing their own searching hop values, all the nodes have to apply the same searching values. The global values should minimize the expected searching cost for the entire network rather than for each individual node.

We limit our scope to the two-ring RS searching scheme since the three-ring RS scheme does not bring significant cost improvement. This conclusion is drawn from the model, and we believe that it is also valid for hop-based networks. For now, there

is only one parameter we need to determine: the first searching hop h_{sys}.

First, we can prove that for a uniformly distributed network, the Probability Distribution Function (pdf) of a random node location x away from the border is

$$f_X(x) = 2(1-x) \qquad 0 \leq x \leq 1 \tag{3.20}$$

Given a node located at x, we can reverse the earlier area-to-hop mapping to determine the searching area value $A(h_{sys}, x)$ for the searching hop h_{sys}.

$$A(h_{sys}, x) = T_{h_{sys},x} = \sum_{i=0}^{h_{sys}} \tilde{N}_{i,x} \tag{3.21}$$

Putting $A(h_{sys}, x)$ into equation 3.8 or 3.17, we can obtain the searching cost $C(h_{sys}, x)$ for the node at x using the searching hop limit h_{sys}. Considering nodes can be at any location with the pdf $f_X(x)$, the system-wide expected searching cost can be expressed as

$$C_{sys}(h_{sys}) = \int_0^1 f_X(x) C(h_{sys}, x) dx \approx \sum_{i=1}^{\frac{1}{\delta}} f_X(i\delta) C(h_{sys}, i\delta) \tag{3.22}$$

Here is how we determine the systematic searching hop using Eq. 3.22. For each possible hop value of h less than the estimated network diameter M, we sample x from [0,1] using sampling interval δ and determine the corresponding $C(h, x)$. We then use equation 3.22 to calculate the system cost $C_{sys}(h)$, and determine the optimal first searching hop h_{opt} where the minimal C_{sys} is obtained. This computation only needs to be done once, and the optimal searching hop h_{opt} will be applied by all the nodes in the network.

We tested the above global first searching hop procedure in a network with 1000 nodes. We first investigate the effect of the sampling interval δ on the accuracy of the first hop limit and the computational complexity. From table 3.3, we find that when δ decreases as the sequence $\{0.1, 0.05, 0.02, 0.01, 0.05\}$, the hop limit of the 2 out of 20 task is always 7, and the hop limit of the 10 out of 20 task is $\{14, 15, 14, 13, 13\}$. Although a small interval may lead to more accurate hop count calculation, the improvement is restricted since the hop limit must be chosen as an integer. Since computation increases linearly with $\frac{1}{\delta}$, we believe that the interval of 0.1 is good enough for use, and we apply $\delta = 0.1$ for the rest of our simulations. The ∞ for the 18-out-of-20

Table 3.3: The impact of sampling interval δ.

δ	Computations	First searching hop of $\{2, 10, 18\}$-out-of-20
0.1	10	$\{7, 14, \infty\}$
0.05	20	$\{7, 15, \infty\}$
0.02	50	$\{7, 14, \infty\}$
0.005	200	$\{7, 13, \infty\}$

task indicates that there is no better scheme to find 18 targets more efficiently than just searching the entire area once by using a large enough hop limit.

In Fig. 3.7, we compare the system-wide cost and latency performance of our scheme with that of the DSR and the EXPansion ring schemes. In RS, the first hop limit of $\{7, 14, \infty\}$ are used for finding $\{2, 10, 18\}$ out of 20 targets. Again, RS performs consistently well for all the searching tasks. When the number of targets to be found is small as for the 2-out-of-20 task, EXP performs close to RS in terms of cost with a much higher latency. The estimated network diameter is 29 in the tested scenario. Therefore, using ∞ as the first hop means to choose any number larger than 29 and search the entire network just once.

Table 3.4: The impact of erroneous N and m on cost.

e_N	-50%	-40%	-30%	-20%	-10%	10%	20%	30%	40%	50%
1st hop	9	9	8	8	8	8	7	7	7	7
e_m	-100%	-80%	-60%	-40%	-20%	20%	40%	60%	80%	100%
1st hop	10	9	9	8	8	8	7	7	7	7

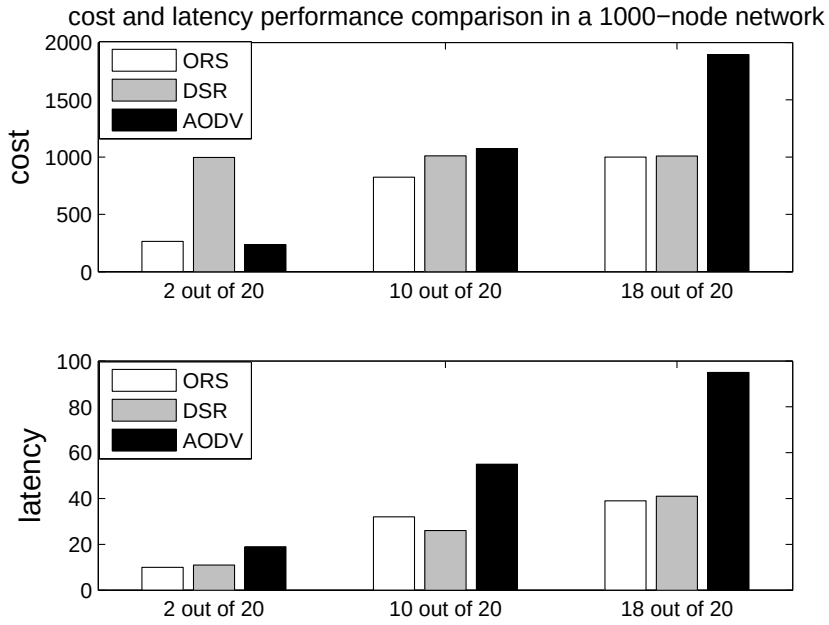

Figure 3.7: Searching cost and latency comparison in small-scale networks for self-location unaware nodes. RS performs consistently close to optimal in terms of both cost and latency for all searching tasks.

3.2.3 Robustness validation

Our algorithm RS outperforms DSR and EXP because it utilizes knowledge of the network parameters N and m to choose the optimal searching hop limits. The EXP scheme, on the other hand, also requires this network information to a certain degree. First, EXP needs m to determine if the task of k-out-of-m is feasible by checking $k < m$. Then, it requires N to estimate the network diameter M so that it knows when it should stop the expansion search. Failure to estimate M may lead to redundant attempts to flood the entire network, especially when the task cannot be completed. During the network design phase, the scale of the network is usually determined and the value of N may be roughly estimated. The information of the server numbers m can

be achieved by letting each server announce its existence through broadcasting when a service is available. Due to the dynamic nature of ad hoc networks, knowing the existence of a service does not mean that nodes will know where the server is and how to reach it.

Although both RS and EXP require knowledge of N and m, RS needs it to be more accurate. Erroneous N and m may lead to erroneous calculation of the first hop limit and thus affect the final searching cost. In this section, we will study the impact of erroneous parameters and test the robustness of our algorithms.

First, for a network of N nodes, let us define the error of N as $e_N = \frac{\tilde{N}-N}{N}$. $\tilde{N}$ is the estimated total number of nodes in the network. Similarly, we can define the error for the number of targets m as $e_m = \frac{\tilde{m}-m}{m}$, where $\tilde{m}$ is the estimated total number of targets in the network.

Although e_N and e_m are two different types of errors, when applying RS using these erroneous values, they both end up in an erroneous value of the first hop limit. For example, for the 2-out-of-20 task, the hop limits calculated based on erroneous e_N or e_m are shown in table 3.4.

An example of how these erroneous first hop limits affect the cost can be found in Fig. 3.8. Only when the error is very large, e.g., as large as $e_m = 100\%$, does the cost increase from the optimal 265 transmissions per search to 364 transmissions per search. Even so, the cost saving is still substantial. For not so large errors, the cost will be 279 or 315 transmission, which is not so far away from the cost of the optimal 2-ring searching scheme, 265 transmissions.

3.2.4 Choosing the right strategy

Depending on the amount of information about the network parameters and the searching task, different searching strategies should be chosen. When N and m can be accurately estimated, RS can be applied to save cost while reducing the latency by about 50% compared to EXP. When N and m cannot be accurately estimated but the number of required targets k satisfies $k << m$, EXP can be performed to reduce cost while doubling the latency. When k is close to m or when no information is known about the network topology, a simple flooding searching scheme is best since its latency is the smallest and it may perform even better than an arbitrary n-ring scheme. The DSR

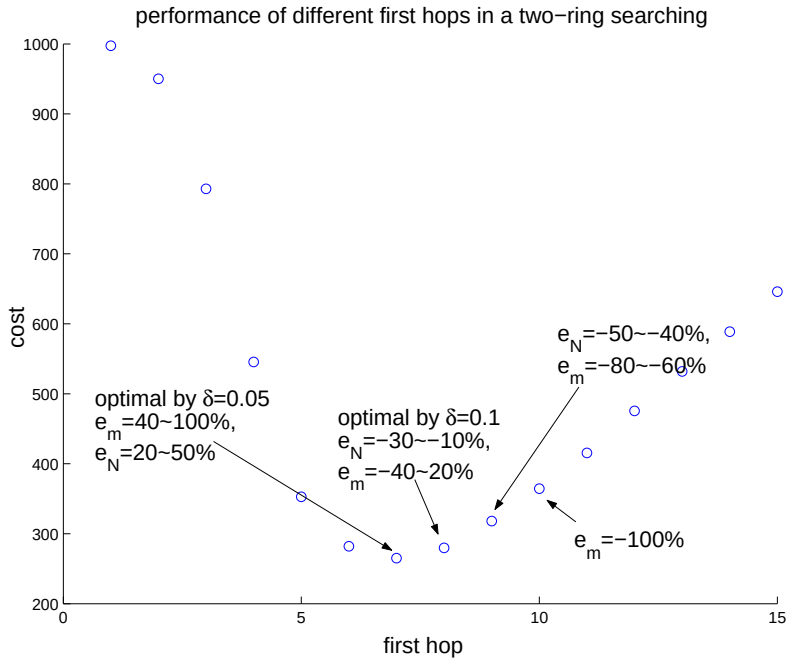

Figure 3.8: The cost for all the possible first hops using a two-ring searching in small-scale networks. The x-axis indicates the first hop limit. Erroneous network parameter estimations result in erroneous hop limit choice. Substantial cost saving can still be achieved using erroneous parameters.

scheme shows a trivial cost improvement and a trivial latency degradation compared to the 1-ring scheme, and hence is of little practical value.

3.3 Conclusions

In this chapter, we studied the target discovery problem in wireless networks. We modelled the problem and proposed RS to discover the optimal searching parameters. We illustrated how to apply the model to realistic network scenarios. General searching strategies are concluded after we investigated the performance of our scheme and compared it with that of other schemes. The amount of information about the network parameters and the desired task determines the best searching scheme.

Chapter 4

Adaptive Local Searching and Route Caching Schemes in Mobile Ad Hoc Networks

The study of peer discovery in the previous chapter is based on two assumptions. First, the normal peers and the target peers are uniformly distributed within the network. Second, the information returned from target peers is always correct. In this chapter, we will study a special but common peer discovery case: route discovery with route caches. This case violates both the assumptions if we consider nodes containing route caches as multiple targets. First, target nodes are no longer uniformly distributed since route caches are more likely to be closer to the destination node. Second, the routing information returned from target nodes is no longer guaranteed to be accurate since stale route caches may be returned from those intermediate nodes. Considering that most of the routing overhead is from route discovery and route caches are widely used in this discovery process, it is worth a separate look at the peer discovery problem for this case.

In mobile ad hoc networks, on-demand routing protocols attract more interest than table-driven protocols because they only initiate a route discovery process when a packet is ready to be transmitted. Without the necessity of persistent maintenance of a routing table as in the proactive table-driven protocols, on-demand protocols typically have lower routing overhead, especially when node mobility is involved. However, the

routing overhead is still excessive from the query flooding during route discovery. Two techniques have been introduced to further reduce routing overhead: route caching and searching localization.

Route caching plays an important role in reducing both the overhead and the discovery latency. With a route caching scheme, once a route is discovered, a node stores it in a route caching table. This route is cached for two purposes. First, when the node has another packet for the same destination, it refers to this route cache instead of initiating a new route query process. Second, when the node hears a route query from another node for the same destination, it may return the cached route to the querying node instead of continuing to forward the query. Route caching reduces the routing overhead originating from the source node as well as reducing the discovery latency for other nodes.

However, applying caching alone is not as effective as expected. This is because that route queries are flooded and route caches may become invalid due to frequent network topology changes. First, when an intermediate node returns a cached route and stops forwarding the query, it cannot quench the flooding. The flooded query will get around this node through other directions and continue to flood the whole network, just like water flowing down from a mountaintop will reach the ground even though some boulders may block the way. Therefore, routing overhead is not reduced by allowing intermediate nodes to return a route cache. Second, although the discovery latency of the first packet that triggers the route discovery process is decreased by faster responses from intermediate nodes, this gain may be counteracted by an even larger propagation latency for the following packets. This is because if the route cache is non-optimal (i.e., not the shortest path), the remaining packets that utilize this non-optimal route will have a larger latency than necessary. Overall, fast responses from route caching only benefit the first several packets while impairing the rest of the packets if the applied cached route is non-optimal. Third, the worst case happens when the returned route cache is invalid. Using this route cache, the source node has to restart the route discovery process after receiving a route error message returned from the broken link. The unwanted consequence is that both the routing overhead and latency are increased, plus several data packets are lost due to the broken route.

Caching optimizations have been extensively researched. However, we will point

out that in order to obtain full benefits from route caches, a local searching scheme must be performed in conjunction with route caches. Also, a more accurate method for quantifying the quality of caches is required in order to avoid the adverse effects from stale caches. We will propose a route caching quality measurement scheme that associates cache quality with node mobility. An adaptive local searching scheme is proposed to adjust to the caching conditions in the network. We quantify the overall routing overhead using these two techniques and determine the strategy for the local searching scheme. We apply the strategy using DSR and demonstrate its efficiency through extensive simulations.

4.1 Model and Analysis

4.1.1 Nomenclature and assumptions

Let us consider *N* homogenous nodes with unit transmission range that are randomly distributed in a disk area. Nodes move with the maximum speed of S_m in the random waypoint pattern. For the traffic model, we assume that each node has a total traffic event rate of λ_T. *Event* indicates a one-way traffic flow towards a destination that is randomly selected from all the rest nodes. The arrival of the events is a random process that follows a poisson distribution, and each event lasts for a fixed event lifetime T_l. During this lifetime, the traffic is not necessary to be bursty and continuous. For example, for an event lifetime of 10 seconds, a node may have only 10 packets with one packet per second.

During our analysis, we assume that the basic route caching options of *gratuitous route repair* and *non-propagation route request* are turned on. Without *gratuitous route repair*, after a RERR is received, the source node will keep receiving invalid caches from the same intermediate nodes each time it attempts route discover again. The loop of RREQ-invalid cache-RERR will continue until the cache in the intermediate nodes expires. The performance without this option is very poor to be studied. *Non-propagation route request* is the same as our local searching technique and will be fully studied. Furthermore, we assume that each node has at most one route cache entry for each destination. To avoid reply storms, we also assume that the destination only replies to the first route query packet.

Table 4.1: Key Notations used throughout this chapter.

Src	the Source node
I	the Intermediate node
D	the Destination node
S	node speed
M	Maximum hops
N	total Number of nodes
P_v	route caching validation probability
λ	event rate
T_l	event LifeTime
T_v	route cache Valid Time
ORR	Overhead Reduction Ratio

Table 4.1 lists the symbols, definitions and variables that are used throughout the chapter. We will extend our discussion on these assumptions and other possible assumptions in Section 4.4.

4.1.2 Route caching validation probability

The first term we are going to introduce is *route caching validation probability* P_v, which indicates the probability for a route cache to be valid. By valid route, we mean that a packet can follow this cached route to reach the destination. An invalid cache will cause routing failure and lead to a new route discovery process, which adds to the routing overhead.

P_v is related to three factors: the maximum node speed S_m, the number of links L contained in the route and the elapsed time T since its last use. It is obvious that the larger the value of S_m, T and L, the less probability P_v for the route to remain valid. In other words, the function $P_v(S_m, L, T)$ decreases monotonically and continuously as its variables increase. $P_v(S_m, L, T)$ can be decomposed as

$$P_v(S_m, L, T) = P_{lv}^L(S_m T) \tag{4.1}$$

Here, $P_{lv}(t)$ is the probability for an originally connected link to remain valid after the

two nodes of the link move for a time period t with a random speed from $[0, 1]$. This transformation simplifies the route validation problem into a unit speed link validation problem. This transformation is valid since S_m can be seen as a scale for time, and also, for a route cache to be valid, each of its independent L links must remain connected after time T. Although the lifetime of different routes may be correlated by certain common links, the lifetimes of different links within one specific route are independent with each other. This is validated through simulations.

The derivation of the closed form for $P_{lv}(t)$ in a two-dimensional space is non-trivial and we will discuss this later in section 4.1.6. In order to not deviate from the main theme, for now, we just take $P_{lv}(t)$ and the corresponding $P_v(S_m, L, T)$ as known functions.

P_v, as a route cache quality measurement, can be used in route discovery from several aspects. First, a source node can specify the quality of cache it desires before sending the RREQ packet. The source node just needs to determine the validation threshold p_t and appends this value in the RREQ packets. Intermediate nodes with route caches respond only if they have a qualified route cache with its calculated P_v larger than p_t. By adjusting p_t, the source node is able to adapt to the current caching situation and reduce unnecessary RREQ packets. Second, P_v allows the source node to determine which cache to choose from multiple RREP packets. The route cache with P_v close to 1 and a shorter route length will be preferred. Third, P_v also helps with route caching management. Nodes can remove a route cache proactively if its P_v is lower than a certain value. Also, when a new route cache arrives and the route caching table is already full, this new route cache can simply replace the route cache with the lowest P_v.

The introduction of P_v enables us to handle route caches more properly. How to set up a proper value of threshold p_t to avoid receiving "bad" caches will be resolved later. Next, we will introduce another parameter *local searching radius*, which is used to control the propagation of the RREQ packets.

4.1.3 Local searching radius

The local searching scheme and the caching scheme should work together. In the last chapter, we show that if there is only one target peer and there are no caches in the

network, local searching will not reduce any searching cost. On the other hand, without local searching scheme, flooded route query packets will propagate throughout the network, and the caching scheme cannot prevent it from happening.

Combining both schemes is a double-edge sword. If a local search finds the target itself or it finds valid route caches to the target node in intermediate nodes, the network-wide flood can be avoided, thus reducing the searching overhead. However, if this search fails to return any result, a network-wide search is still required and the overall searching cost is even more than a simple network-wide flood. Thus, the local searching radius k should be carefully chosen (Note that route caching conditions are objective and non-alterable.) If the radius, denoted by k, is chosen too large, the probability of discovering a route returned from the destination itself is large and little benefit can be obtained from the route caches. If the radius k is chosen too small, the chance of discovering the destination or a cached route to the destination in this local search is also small and little benefit can be gained from this local search because the first round local search will be part of the total searching overhead. As we will show later, the radius k is the key parameter in determining the RREQ overhead and is closely related to the caching conditions in the network. How to determine k, plus the caching threshold p_t from the earlier discussion, is the major objective of the rest of our analysis.

4.1.4 Life periods of a cache

To understand how a node responds to other nodes' route requests, let us first clarify the life periods of a route cache. The time interval between two traffic events from a certain **Src** to a certain **D** can be divided into three caching periods, as shown in Fig. 4.1, which are the guaranteed valid period I, the probable valid period II and the invalid period III. In different periods, a route cache is of different qualities and has different effects on the routing overhead and the routing performance.

Starting from the leftmost of Fig. 4.1, when a new event, say event i, just arrives, **Src** initiates a route discovery process and caches the discovered route for future use. During period I, all of the traffic packets of this event will follow this cached route. Meanwhile, if the route is broken due to node mobility, the route maintenance will detect it and initiate a new route discovery. Thus, during the event lifetime T_l, this node maintains a valid route for **D**. If the node receives a RREQ for **D** from another

node, it can guarantee to return a valid route. Ideally, during this period, the route cache validation probability P_v for **D** equals 1. However, in practice, the traffic is in the form of discrete packets instead of continuous data flow and there may be time intervals between two consecutive packets. Within these time intervals, P_v is actually less than 1. Also, a RREQ packet may arrive when the route happens to be broken and **Src** is still in the route maintenance phase, although the probability for this case to happen is quite small. Thus, strictly speaking, during period I, a node can respond with a route cache whose P_v is very close to 1.

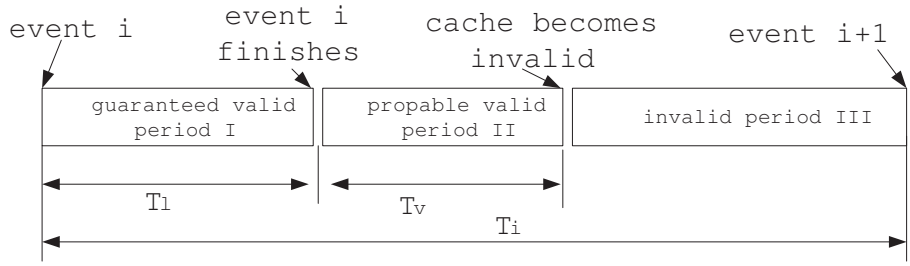

Figure 4.1: Time periods for a node's route caching between two events towards the same destination. During the guaranteed valid period I, the node has traffic and maintains the cache. In the probable valid period II, the node has stopped sending traffic to the destination and therefore has only a probable valid route cache. In the invalid period III, the route caching valid probability is so low that this route cache may be discarded.

During life period II when there is no more traffic between **Src** and **D**, there are no more proactive methods such as route maintenance to refresh the cached route. As time elapses, this cached route becomes invalid gradually, i.e., P_v decreases. If a RREQ arrives with the validation threshold p_t, a route cache will be returned in response only if its validation probability P_v satisfies $p_t < P_v < 1$. In other words, when p_t is given, the time length T_v of life period II can be explicitly found by solving $P_v(S_m, L, T_v) = p_t$, or $P_{lv}^L(S_m T_v) = p_t$.

During Period III, P_v is below the threshold p_t and we consider the cache no longer valid. When a RREQ with the caching validation p_t arrives, this node will not return the route cache. This period lasts until the next event arrives and a new round starts. Also, a node may remove a cache proactively when its P_v is very small. However, since each RREQ may arrive with different values of p_t, correspondingly, T_v may vary

for different RREQs as well. A route cache not satisfying one RREQ packet does not necessarily indicate that it does not satisfy other RREQ packets. Therefore, a cache can only be removed when it is guaranteed to be of no future use, i.e., P_v is very small.

When a route request arrives at the intermediate node **I** randomly at any time, it may fall into any of the above three periods. The probability that it falls into Period I, in other words, the probability P_I for this intermediate node to return a guaranteed valid route is

$$P_I = \frac{T_l}{T_i} \tag{4.2}$$

From a node's view, its total event rate of λ_T is evenly distributed towards the other $N-1$ nodes. Thus the event rate towards a certain node is $\lambda = \frac{\lambda_T}{N-1} \approx \frac{\lambda_T}{N}$. The average time interval T_i between two events for the same source destination pairs is $T_i = \frac{1}{\lambda} = \frac{N}{\lambda_T}$. Thus equation 4.2 becomes

$$P_I = \frac{T_l \lambda_T}{N} \tag{4.3}$$

The probability P_{II} for the route request to fall in Period II, that is, for the intermediate node to return a probable valid route is

$$P_{II} = \frac{T_v}{T_i} \tag{4.4}$$

As mentioned earlier, T_v in equation 4.4 can be solved from $P_v(S_m, L, T_v) = p_t$, or $P_{lv}^L(S_m T_v) = p_t$.

Note that except p_t, all the other variables in equations 4.3 and 4.4 are parameters related to the network pattern or the traffic pattern, and they are not controllable by the protocols. It is the source node's responsibility to determine a good p_t so that there will be route caches returned by intermediate nodes and these route caches are of good qualities.

4.1.5 Overhead reduction ratio (ORR)

The primary goal of this chapter is to achieve the minimum routing overhead, i.e., to reduce the number of RREQ and RREP packets as much as possible. We define *overhead reduction ratio* to measure the effectiveness of the combined caching and local searching schemes. Since RREQ packets are required to be flooded while RREP

packets are unicasted, RREQ packets are dominant in the overall routing overhead and we only consider RREQ packets in the measurement of ORR. Suppose that with the caching and local searching schemes, the overhead is O, and without these schemes, the overhead is O_n. Then ORR is defined as

$$ORR = \frac{O_n - O}{O_n} \tag{4.5}$$

Next, we will derive the overhead O as a function of the variables of p_t and k and other network and traffic parameters. The analysis of O can be briefly described as follows. If the destination is non-local, when a RREQ is flooded locally with radius k, it may fall into different life periods of the required route caches. Different caches with different validation probabilities P_v may be returned, and different routing overhead occurs correspondingly. Therefore, the expected overhead can be calculated based on the validation probability of these returned caches.

The propagation of a RREQ packet in a local searching scheme can be illustrated as in Fig. 4.2. The source node **Src** floods a RREQ packet looking for node **D** locally with radius (hops) k and route cache validation probability threshold p_t. Each node inside the k hops range may return a guaranteed valid route cache with probability P_I. Suppose there are N_i nodes at exactly i hops away from node **Src**. The probability P_g for node **Src** to receive at least one guaranteed cache is

$$P_g = 1 - \prod_{i=1}^{k} (1 - P_I)^{N_i} \tag{4.6}$$

This equation can be explained as the probability of obtaining no cache at all occurs only when not a single local node has the cache, and by subtracting this value from 1 we find the probability of obtaining at least one cache.

Similarly, each node inside the k-hop range may return a probable valid route cache if the RREQ falls into the cache life period II. Thus, the probability of receiving at least one probable valid route cache P_p is

$$P_p = 1 - \prod_{i=1}^{k} (1 - P_{II})^{N_i} \tag{4.7}$$

The difference between equation 4.6 and 4.7 is that P_I is a constant value for all the nodes, while P_{II} is distinct for each RREQ. In equation 4.7 we categorize nodes at the

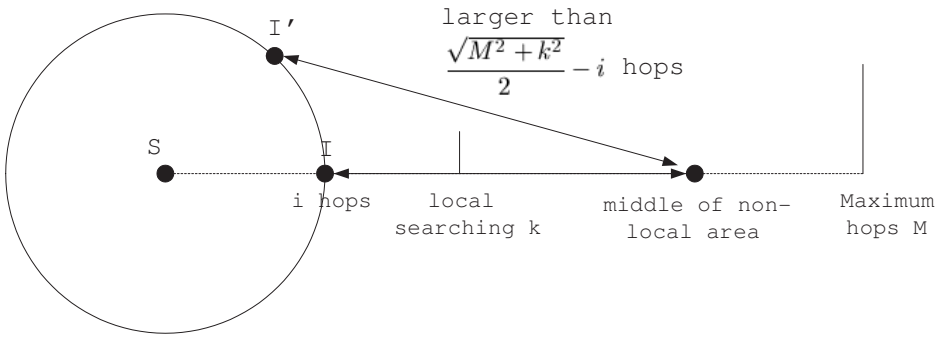

Figure 4.2: Local searching example. The shortest distance from the intermediate node that is i hops away from the source node to the middle of the non-local area is $\frac{\sqrt{M^2+k^2}}{2} - i$. k is the local searching radius and M is the maximum hop of the network.

same hop-distance from node **Src** for a simpler expression, e.g., nodes **I** and **I'** from Fig. 4.2 are considered to have the same value for P_{II}. Although this categorization simplifies the expression and the analysis, it may bring error in the final results. We will discuss the error and how to compensate it in section 4.2.3.

Suppose there are $N_l(k)$ nodes in the k-hop local area, and the total number of nodes in the network is N. We can categorize the expected RREQ overhead O into the following cases:

1. When node **D** is local: cost is local $N_l(k)$.
2. When node **D** is non-local and a guaranteed valid route cache is found locally: cost is local $N_l(k)$.
3. When node **D** is non-local and a guaranteed valid route is not found but a probable valid route is found, and it is valid: cost is local $N_l(k)$.
4. When node **D** is non-local and a guaranteed valid route is not found but a probable valid route is found, and it is invalid: cost is local plus network-wide $N_l(k) + N$.
5. When node **D** is non-local and no cache is found: cost is local plus network-wide $N_l(k) + N$.

Considering the probability for a node to be local as $\frac{N_l(k)}{N}$ and non-local as $1 - \frac{N_l(k)}{N}$, we can transform the above descriptions into the equation of the expected routing

overhead O:

$$\begin{aligned} O =& \frac{N_l(k)}{N} N_l(k) + (1 - \frac{N_l(k)}{N}) P_g N_l(k) + (1 - \frac{N_l(k)}{N})(1 - P_g) P_p P_v N_l(k) \\ &+ (1 - \frac{N_l(k)}{N})(1 - P_g) P_p (1 - P_v)(N_l(k) + N) \\ &+ (1 - \frac{N_l(k)}{N})(1 - P_g)(1 - P_p)(N_l(k) + N) \\ =& N_l(k) + (N - N_l(k))[(1 - P_g)(1 - P_p P_v)] \end{aligned} \tag{4.8}$$

Compared to a simple network-wide search without caching whose overhead O_n is the total number of nodes N, the Overhead Reduction Ratio ORR is

$$\begin{aligned} ORR &= \frac{O_n - O}{O_n} = \frac{N - N_l(k) - (N - N_l(k))[(1 - P_g)(1 - P_p P_v)]}{N} \\ &= (1 - \frac{N_l(k)}{N})(P_g + P_p P_v - P_g P_p P_v) \end{aligned} \tag{4.9}$$

Equation 4.9 informs us that the caching scheme saves only when the target is in the non-local area and when a valid route cache is found locally, either from a guaranteed valid route cache or from a probable valid route cache which *is* valid. The last item $P_g P_p P_v$ needs to be deducted since there is a chance that both guaranteed valid caches and probable valid caches are received while only one of them can be chosen.

Let us first look at a scenario where P_g and P_p are much less than 1. This is a typical scenario in which the route cache availability is weak or moderate. Now ORR can be further expressed as

$$\begin{aligned} ORR &= (1 - \frac{N_l(k)}{N})(P_g + P_p P_v - P_g P_p P_v) \approx (1 - \frac{N_l(k)}{N})(P_g + P_p P_v) \\ &\approx (1 - \frac{N_l(k)}{N})(P_g + P_p p_t) \end{aligned} \tag{4.10}$$

The expressions of P_g and P_p from equations 4.6 and 4.7 are too complex to be directly input to equation 4.10 for analysis. However, with the assumption of P_g and P_p being much less than 1, P_g and P_p can be transformed to

$$P_g \approx 1 - (1 - \sum_{i=1}^{k} N_i P_I) = \sum_{i=1}^{k} N_i P_I, \qquad P_p \approx 1 - (1 - \sum_{i=1}^{k} N_i P_{II}) = \sum_{i=1}^{k} N_i P_{II} \tag{4.11}$$

Now, we have the ORR as

$$\begin{aligned} ORR &\approx (1 - \frac{N_l(k)}{N})(P_g + P_p p_t) = (1 - \frac{N_l(k)}{N})(\sum_{i=1}^{k} N_i P_I + \sum_{i=1}^{k} N_i P_{II} p_t) \\ &= ORR_1(k) + ORR_2(k, p_t) \end{aligned} \tag{4.12}$$

$ORR_1(k)$ is the overhead reduction from local guaranteed valid caches, and it is only related to the local searching hop number k. $ORR_2(k, p_t)$ is the overhead reduction from local probable valid caches. Besides k, ORR_2 is also related to p_t since an appropriate value of p_t can not only prevent the source node from receiving bad caches by an overly large p_t, but also avoid receiving no cache and achieving no benefits from caching schemes by an overly small p_t. With equation 4.12, we are able to determine the optimal parameters k and p_t to maximize the overall ORR.

4.1.6 Optimize the parameters: k and p_t

The ORR in equation 4.12 is composed of two items. The first item represents the overhead reduction achieved from the caches in life period I. The second item represents the overhead reduction achieved from the caches in life period II. In this section, we will optimize the parameters k and p_t to maximize $ORR1$ and $ORR2$, and correspondingly, to maximize ORR.

First, from statistical results, in a network of limited size, the number of nodes at certain hops N_i and the corresponding number of nodes $N_l(k)$ in the local searching area of radius k can be estimated as

$$N_i \approx \frac{6(Mi - i^2)}{M^3} N, \qquad N_l(k) = \sum_{i=1}^{k} N_i = N \frac{6}{M^3} (\frac{Mk^2}{2} - \frac{k^3}{3}) \tag{4.13}$$

where M is the maximum number of hops in the network. Figure 4.3 shows how close the above two estimations are to the numerical results. In this experiment, 150 nodes are randomly distributed in a unit circle area. Each node has a transmission range of 0.35. The results are based on 30 simulations. As can be seen, the estimated number of nodes in the left plot is close to that of the numerical results. Their major difference is located at the trailer part for hop 7 and 8. At those hops, there may still be some nodes in rare cases. However, these values are so small (less than 5) that we believe our estimation is accurate enough. Also, the estimation for the number of nodes within hop i is accurate enough for our analysis purpose.

Plugging these values into ORR_1, we have

$$ORR_1(k) = (1 - \frac{N_l(k)}{N}) \sum_{i=1}^{k} N_i P_I = (1 - \frac{6(\frac{Mk^2}{2} - \frac{k^3}{3})}{M^3}) \frac{6}{M^3} (\frac{Mk^2}{2} - \frac{k^3}{3}) T_l \lambda_T \tag{4.14}$$

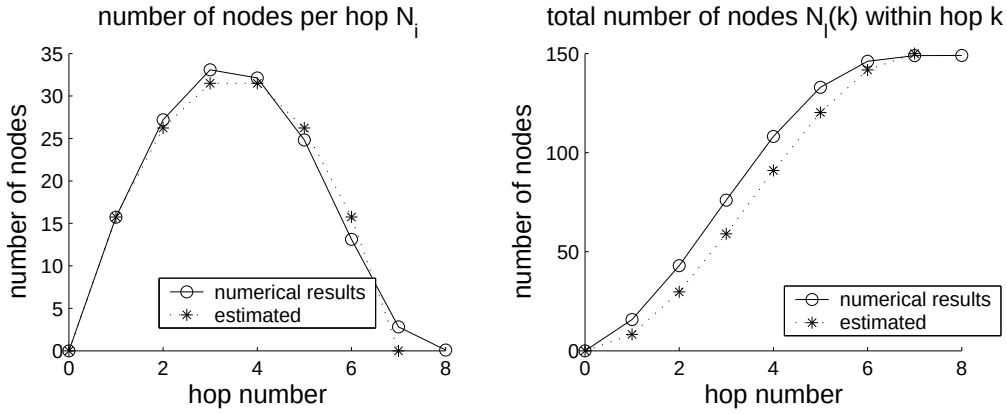

Figure 4.3: The number of nodes at each hop N_i (left) and the number of nodes within k hops (right). Both the experimental results and their estimates are shown.

It is easy to determine that ORR_1 reaches its maximum at $k = \frac{M}{2}$. In other words, setting $k = \frac{M}{2}$ will lead to the largest overhead reduction achieved from the guaranteed valid life period I.

The second item $ORR_2(k, p_t)$ becomes

$$ORR_2(k, p_t) = (1 - \frac{N_l(k)}{N}) \sum_{i=1}^{k} N_i P_{II} p_t = (1 - \frac{N_l(k)}{N}) \sum_{i=1}^{k} \frac{6(Mi - i^2)}{M^3} N \frac{T_v(S_m, L, p_t)}{T_i} p_t \tag{4.15}$$

In the above equation, L and $T_v(S_m, L, p_t)$ are still needed to be clarified. From Fig. 4.2, L is the hop distance from the intermediate node to the destination node. Since the destination node is randomly distributed within the non-local area, we just simply take the middle of the non-local area from hop k to hop M. Since the number of nodes is proportional to the area and thus proportional to the square of the hops, thus we have the middle hop k' that separates the non-local area equally as

$$(k')^2 - k^2 = M^2 - (k')^2 \Rightarrow k' = \frac{\sqrt{M^2 + k^2}}{2} \Rightarrow L = k' - i = \frac{\sqrt{M^2 + k^2}}{2} - i \tag{4.16}$$

L is the average shortest hop distance from the intermediate node located at hop i to this middle point.

Now that S_m, L and p_t are known, T_v, the time length of the cache period II, can be determined through the function $P_v(S_m, L, T_v) = p_t$. Next, we will determine the expression for P_v to solve for T_v.

In the earlier section when we introduced P_v, we mentioned that $P_v = P_{lv}^L(S_m T)$. Let us restate the definition of $P_{lv}(t)$. A link is formed by two nodes of random distance D uniformly distributed from $[0, 1]$. Suppose that they both have a velocity with a random speed from $[0, 1]$ and a random direction from $[0, 2\pi]$. If we say that the link is broken when the distance becomes larger than 1, then $P_{lv}(t)$ indicates the probability for the link to remain connected at time t. Since P_{lv} is not linearly related with the caching lifetime t, as seen from Fig. 4.4, our measurement for cache qualities is more accurate than traditional cache timeout schemes.

We solved the P_{lv} for a one-dimensional case in which the nodes of the link move either towards each other or away from each other. The closed form of this one-dimensional case is

$$P_{lv}(t) = \begin{cases} 1 - \frac{t}{3} & t < 1 \\ \frac{1}{t} - \frac{1}{3t^2} & t \geq 1 \end{cases} \tag{4.17}$$

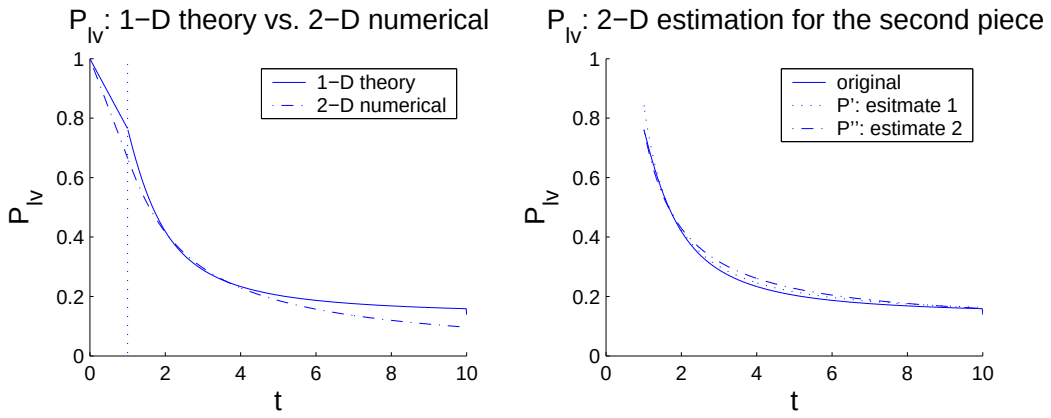

Figure 4.4: The $P_{lv}(t)$ in 1-D and 2-D cases. On the left figure, the theoretical results of $P_{lv}(t)$ in a 1-D case and the numerical results in a 2-D case are shown. They both have a two piecewise tendency. On the right figure, two estimations for the 2-D numerical results are shown.

The deduction of the closed form for the two-dimensional case is very difficult since there are two more angle random variables involved. Numerical results show that the 2-D case has a two piecewise tendency, similar to the 1-D results, shown in the left part of Fig. 4.4. When t starts from zero, P_{lv} decreases linearly. After $t = 1$, the decreasing tendency starts to flatten. From this, we conjecture that the closed form of the 2-D case is of the same form as the 1-D case. For the first linear part where $t < 1$, it is easy to

derive the estimated form from the numerical results

$$P_{lv}(t) = 1 - 0.2338t \qquad 0 < t < 1 \tag{4.18}$$

As for the second part where $t > 1$, we estimate its form as $\frac{a}{t^m} + \frac{b}{t^n}$, similar to the form in the 1-D case in equation 4.17. Two estimated functions P' and P'' are shown in the right part of Fig. 4.4.

$$P'_{lv}(t) = 0.0998t^{0.08} + \frac{0.7518}{t^{1.2400}}, \qquad P''_{lv}(t) = 0.0919 + \frac{0.6762}{t} \tag{4.19}$$

The first curve $P'_{lv}(t)$ has a smaller mean square error than $P''_{lv}(t)$. However, we take the second curve $P''_{lv}(t)$ for our later analysis because it has a much simpler expression for analysis and the error is not too large from the numerical results. Note that we do not consider the curve after $t > 10$ because P_{lv} is already lower than 0.2, which indicates a cache of too low quality P_v to be utilized.

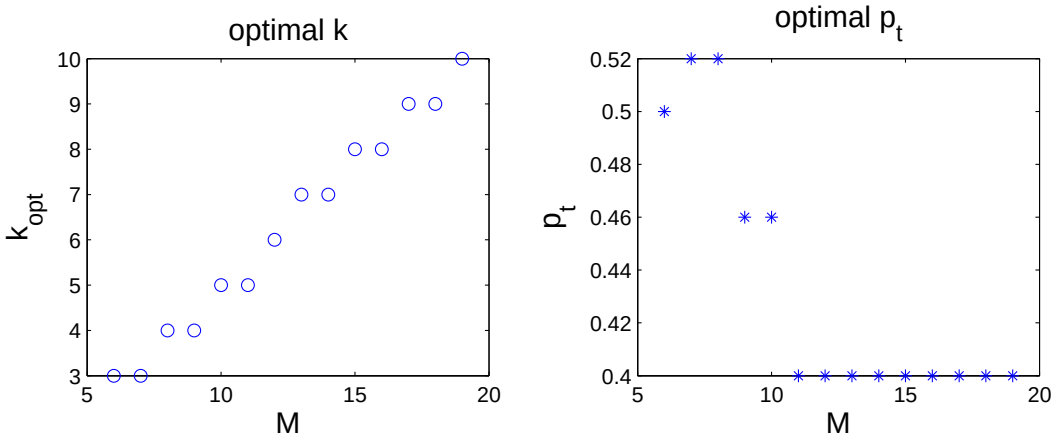

Figure 4.5: The optimal parameters k and p_t. The optimal k value is consistently around $\frac{M}{2}$. The optimal p_t is around 0.4 to 0.5, depending on the the maximum hop radius M.

Since L and the function of $P_v(S_m, L, T_v)$ have been determined, ORR_2 can be rewritten as a deterministic form

$$\begin{aligned} ORR_2(k, p_t) &= (1 - \frac{N_l(k)}{N}) \sum_{i=1}^{k} \frac{6(Mi - i^2)}{M^3} N \frac{T_v(S_m, L, p_t) p_t}{T_i} \\ &= (1 - \frac{N_l(k)}{N}) \sum_{i=1}^{k} \frac{6(Mi - i^2)}{M^3} N \frac{(P'')^{-1}_{lv}(p_t^{\frac{1}{\sqrt{M^2+k^2}/2} - i})}{T_i S_m} p_t \end{aligned} \tag{4.20}$$

In the above equation, $P''_{lv}(t)$ is

$$P''_{lv}(t) = \begin{cases} 1 - 0.2338t & 0 < t < 1 \\ 0.0919 + \frac{0.6762}{t} & t \geq 1 \end{cases} \tag{4.21}$$

Although ORR_2 is expressed as a deterministic function with only two variables k and p_t, it is still difficult to find the maximum value of ORR_2 and the corresponding maximum point k and p_t theoretically. Again, we use numerical methods to find the optimal points. The results are shown in Fig. 4.5. As we can see, when the maximum hops $M < 12$, k should be set to $\lfloor \frac{M}{2} \rfloor$, and when $M \geq 12$, k should be set to $\lfloor \frac{M+1}{2} \rfloor$. For p_t, the optimal value is around 0.4 to 0.5 depending on the maximum number of hops. Thus, to achieve a maximum overhead reduction from route caching life period I and period II consistently, we choose $k = \lfloor \frac{M}{2} \rfloor$ and $p_t = 0.4$.

4.1.7 Numerical examples

So far, we have found the optimal parameters for p_t and k, which are $p_t = 0.4$ and $k = \lfloor \frac{M}{2} \rfloor$, to maximize ORR. These results are based on the scenarios in which P_g and P_p are much less than 1. Typically, this kind of scenario represents low caching availability at low event rate, fast node speed and low traffic lifetime. In other scenarios where P_g and P_p are comparable to 1, the calculation of ORR should be performed by resorting to equations 4.6 and 4.7 instead of the approximations in equation 4.11. In order to have a complete understanding of the relationship between ORR and the scenario parameters, we demonstrate some numerical examples in Figs. 4.6 and 4.7.

In both figures, p_t is fixed at 0.4. The total event rate for each node is 0.05. The total number of nodes is 150 and the estimated maximum hop M for the network is 7. P_g and P_p are shown in the left part of each figure, together with their approximations P_{ge} and P_{pe} calculated from equation 4.11. ORR is shown in the right part of each figure. Fig. 4.6 shows the results for the event lifetime as low as 2 seconds and the maximum node speed of 0.004m/s (the upper part) and 0.04m/s (the lower part). Fig. 4.7 shows the results for the event lifetime as high as 10 seconds and the maximum node speed of 0.004m/s (upper part) and 0.04m/s (lower part). If we map the maximum link distance 1m into the transmission range of 250m, the relative speeds of 0.004m/s and 0.04m/s

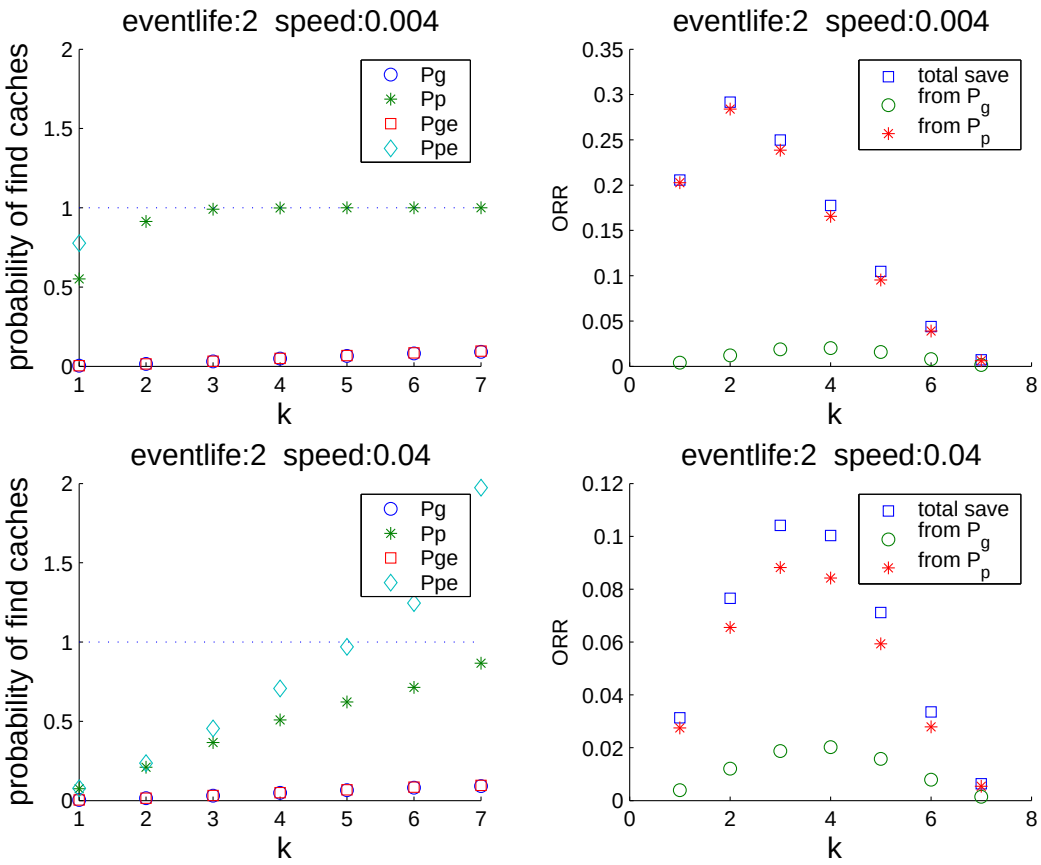

Figure 4.6: The scenarios for event lifetime as low as 2 seconds. The maximum node speed is either 0.004 (upper part) or 0.04 (lower part). The X-axis indicates the local searching radius k and the Y-axis indicates P_g and P_p and their estimations P_{ge} and P_{pe} in left figures and ORR in right figures. The estimation of P_p go beyond 1 at hop 2 and cannot be seen in the upper left plot.

can be mapped to the low speed of 1m/s and the high speed of 10m/s. Thus, the simulation scenarios to be used in the simulation part correspond to the scenarios here.

As can be seen from the lower part of Figs. 4.6 and 4.7, when the number of P_g and P_p is less than 1, the approximation equations are accurate and ORR reaches its maximum when k is near $\lfloor \frac{M}{2} \rfloor$, the same as the conclusions from the last sections. However, when the node speed is as low as in the upper part of both figures, the approximation from equation 4.11 is no longer valid for P_p and ORR reaches its maximum at $k = 2$. P_p becomes close to 1 and the approximate P_p value using equation 4.11 becomes an

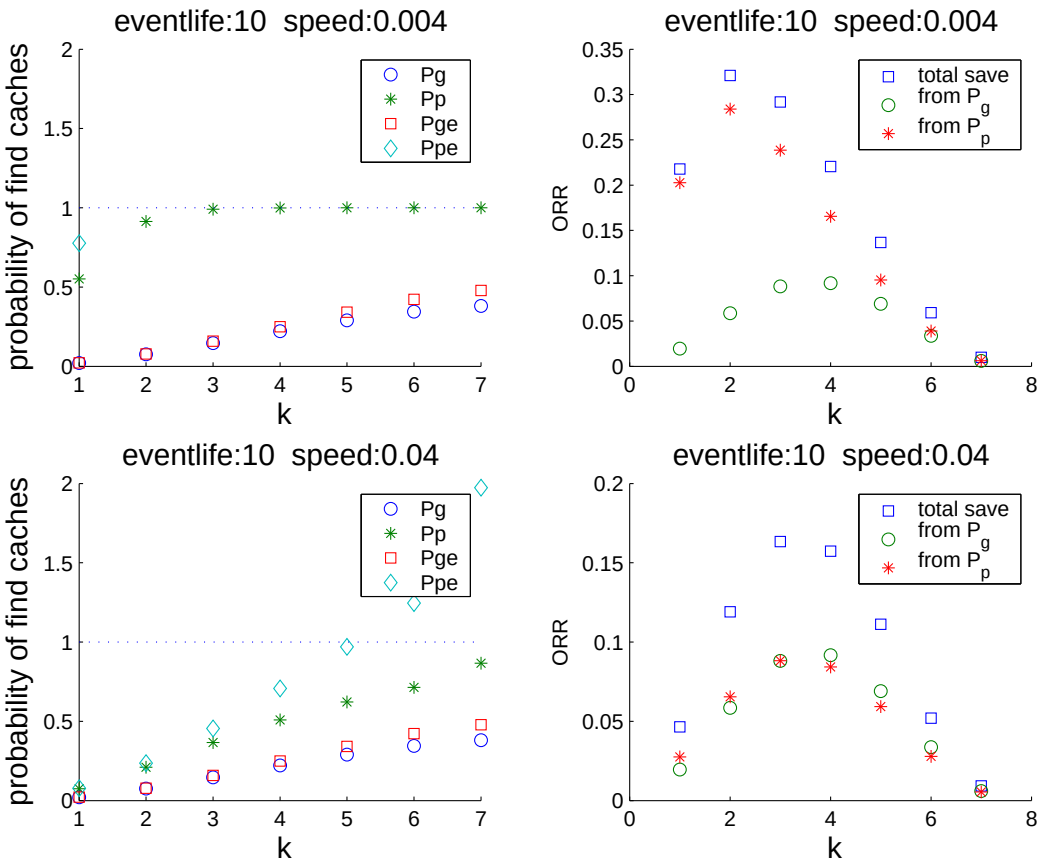

Figure 4.7: The scenarios for event lifetime as high as 10 seconds. The maximum node speed is either 0.004m/s (upper part) or 0.04m/s (lower part). The X-axis indicates the local searching radius k and the Y-axis indicates P_g and P_p and their estimations P_{ge} and P_{pe} in left figures and ORR in right figures. The estimation of P_p go beyond 1 at hop 2 and cannot be seen in the upper left plot.

unrealistic value larger than 1. We do not show these unrealistic approximate values of P_p in the upper left part of Figs. 4.6 and 4.7 once they become larger than 1. There are abundant caches available in this low speed network and it only adds to unnecessary searching cost to apply a large radius. The number of caches from probable valid period II dominates the number of caches from guaranteed valid period I, and the local radius should be adjusted based on the dominating factor. Although we could not determine the optimal k and p_t in a closed form in this abundant caching scenario, we know that k should be adjusted smaller than $\lfloor \frac{M}{2} \rfloor$ and p_t should be adjusted larger than 0.4 to reduce

the overhead. This criteria will be realized in the protocol design in the next section.

Now we have a complete understanding of the relationship between ORR and the network parameters. In summary, when caching availability is moderate, the optimal parameters from analysis should be applied to achieve the maximum overhead reduction. When caching is abundant, the local searching radius should be set where a cache is very likely to be found. With these analytical results, in the next section we will design a routing scheme that is able to dynamically adjust itself to approach the maximum performance under all possible scenarios.

4.2 Protocol Description

Although the analysis is involved, the final conclusions are quite simple. Only two primary parameters are needed, the caching validation probability threshold p_t and the local searching radius k. When the network is just set up, or a node just joins a network, these values should be set to $p_t = 0.4$ and $k = \frac{M}{2}$, assuming weak or moderate caching conditions. When more abundant caching conditions are detected based on history, k should be set to a smaller value according to the dominant factor, such as P_p, the probable valid caching period, shown in the upper part of Fig. 4.6 and 4.7. Also, p_t can be adjusted larger to reduce unnecessary caches of low qualities.

Therefore, only minor changes are needed for existing on-demand routing protocols to fully attain the benefits of caches. In this section, we propose an add-on protocol for existing routing protocols. Its three components, new data structures, protocol procedures and parameter adjustment rules, will be described in detail next.

4.2.1 New data structures

A new field for caching validation probability is required for both the RREQ and the RREP packets. For RREQ packets, the value of this field is calculated through the parameter adjustment rules described below and appended in the RREQ packets to serve as the caching validation threshold. For RREP packets, the value of this field is calculated by the node that initiates the RREP packet to indicate the cache's quality using equations 4.1 and 4.21.

Also, each node maintains a statistic such as the number of recent RREQ attempts,

the values of k and p_t applied, the number of guaranteed valid caches and the number of probable valid caches received. This information is used to estimate the current caching condition to serve for the parameter adjustment rules. The counting of these numbers does not differentiate destinations and only needs a little extra storage. This non-differential counting is valid for uniform traffic scenarios. For biased traffic scenarios such as the client-server traffic model, a maintenance of the history of different destinations may provide more accurate parameter adjustment. The tradeoff is much larger extra storage for each destination node. In our current work, we utilize the general statistical method without destination differentiation.

4.2.2 Protocol procedure

When a source node needs to send a RREQ, it calculates the parameters of k and p_t according to the parameter adjustment rules and attaches the values in the RREQ packet. Intermediate nodes calculate P_v for their cached route to the destination from equation 4.1 and return a RREP packet with P_v attached if P_v satisfies $P_v > p_t$. The source node picks the cached route with the largest P_v. When two cached routes have close P_v values, the one with a shorter route length is preferred. Each node refreshes the statistics each time it sends out a RREQ packet and receives RREP packets from intermediate nodes.

4.2.3 Parameter adjustment rules

The parameter adjustment rules determine the value of p_t according to the current caching situation. A node first calculates the average number of guaranteed valid route caches N_g and the average number of probable valid route caches N_p received from its history, say the last 100 RREQ attempts. Also, from the history it calculates the averages $\tilde{k}$ and $\tilde{p_t}$. These values indicate the current caching conditions. If k already equals $\frac{M}{2}$ and p_t is already 0.4 and N_p and N_g are still less than 1, there is no need to further increase k and p_t since this is a weak or moderate caching condition and the protocol is already running using optimal parameters. A running average over all the past RREQ attempts instead of the last 100 attempts requires less storage. However, it cannot represent the most recent caching conditions and is less accurate for the parame-

ter adjustment. Therefore, there is a tradeoff between storage and parameter adjustment accuracy.

When N_g is larger than 1, guaranteed valid caches become the dominating factor. k should be primarily adjusted according to N_g towards the goal of receiving only one guaranteed cache by using $k = \frac{\tilde{k}}{\tilde{N}_g}$. For example, if the source node uses $\tilde{k} = 4$ to achieve $\tilde{N}_g = 2$ guaranteed valid caches in the history, it should use $k = \frac{4}{2} = 2$ to expect to receive only one guaranteed cache this time. p_t is set to a value at 0.9, which indicates only guaranteed valid caches (or almost guaranteed, more strictly speaking) are needed.

In a more general case, the average number of guaranteed valid caches is much lower than 1 and the probable valid caches are the dominating factor such as in the examples shown in Figs. 4.6 and 4.7. Thus, k should be adjusted towards the goal of $N_p = 1$ by using $k = \frac{\tilde{k}}{\tilde{N}_p}$. If k is already as low as 1 and there are still more than necessary caches returned, p_t should be adjusted larger to accept only more qualified caches by using $p_t = \frac{\tilde{p_t}}{\tilde{N}_p}$. This indicates a very abundant caching condition such as when the node speed is very low and traffic between nodes happens very often.

In summary, k and p_t are adjusted towards receiving one guaranteed valid cache, or one probable valid cache when the chance of receiving guaranteed valid caches is small. However, the adjustment of k should not exceed $\frac{M}{2}$, and the adjustment of p_t should not be lower than 0.4. Exceeding these values only brings about more routing overhead although it may bring about more returned caches.

However, during our simulations, we notice that the adjustment towards the goal of receiving only one probable valid cache is a little conservative in finding qualified caches in some cases. There may be several reasons for this. First, in our analysis part, we use the shortest hop distance from the intermediate node **I** to the destination **D** for all the intermediate nodes with the same hop distance i from the source node. However, most of these intermediate nodes, such as node **I**' in Fig. 4.2, have longer distance and less qualified caches than the shortest hop distance from node **I**. Thus, the results drawn for probable valid caches are overly optimistic in finding route caches. Second, each destination has a different hop distance towards the source node. In our analysis, we simply take the middle point of the non-local area to represent all the non-local destinations. This makes it unfair for some farther destinations and makes the

local searching radius not large enough for those destinations. One solution, as mentioned earlier, is to maintain a more detailed statistic for all the possible destinations and take the difference among destinations into account during the parameter adjustments. However, in our implementation, we use a simpler but more aggressive parameter adjustment method by setting $k = \frac{2\tilde{k}}{\tilde{N}_p}$ towards the goal of receiving two probable valid caches. This simple modification can provide enough good performance and avoid the excessive storage required by the per destination based statistic maintenance.

4.3 Performance Evaluation

4.3.1 Assumptions, metrics and methodology

We simulate our routing scheme as an add-on to the DSR protocol in a mobile ad hoc network scenario. The simulations are performed using ns-2 [19]. In order to focus our study in the routing level, we programmed an ideal lower layer below the routing layer, which is able to send packets without collision and detect link failures automatically with no time and no cost. Working with this virtual bottom layer, the routing protocol can be approximately seen as working at low traffic. As pointed out by [6], although it is a limitation of the simulation model, it is able to make a very good qualitative evaluation at the packet level without being confined by the specific design of the MAC layer. We believe that realistic MAC will have negligible effect due to the broadcast nature of RREQ packets. And for unicasting packets such as RREP, the MAC layer should have the same impact on our scheme as on traditional schemes since there is little modification on the packet structures.

We test all the scenarios in a square region of size $1400m \times 1400m$. Although the region is not of the disk shape as in the analysis part, this square scenario is roughly close to the circular scenario, and it is easier for the setup of the node mobility model. There are 150 nodes inside this area, each moving in a random waypoint mobility pattern. The number of nodes is chosen large enough for good connectivity as well as to make it easy to investigate the performance difference between our scheme and the original DSR scheme. Each node has a transmission range of 250m, and the estimated maximum hop value is 7. The maximum node speed of 1m/s can be mapped to 0.004m/s for unit transmission range (10m/s maps to 0.04m/s), which enables us to compare the

simulation results with the analytical results found in Figs. 4.6 and 4.7 in Section 4.1.

The total simulation time is either 4500 seconds or 2000 seconds, depending on the event rate. The simulation time is chosen longer than 1000 seconds to remove the potential undesirable effects of starting the random waypoint mobility model [5] simultaneously for all the nodes. Also the simulation time is long enough for each pair of nodes to have an event rate larger than 1. For example, if a node has a total event rate of 0.05 events/sec, it needs 4500 seconds to have an average of $\frac{0.05}{150} \times 4500 = 1.5$ events toward each other node.

In our simulations, basic metrics commonly used by former studies [31] are investigated, which are routing overhead, successful delivery ratio, discovery latency and route optimality. However, in order to concentrate on our topic of reducing routing overhead, we only show in the figures the metrics that have significantly changed. Other metrics with little changes will just be briefly mentioned.

We study the performance of three routing schemes: the original DSR with No Caching (DSR-NC), DSR with Caching (DSR-C) and DSR with our scheme of Local searching and Caching added on (DSR-LC). We first validate our results of the selection of k and p_t through exhaustive simulations on DSR-LC. Then we simulate DSR-C and show that the selection of the timeout value has a similar impact on the performance as the selection of p_t in DSR-LC. Finally, we compare the performance of DSR-LC, DSR-C and DSR-NC under different scenarios. We show that our scheme DSR-LC is able to adjust to all the circumstances and shows an overhead reduction ratio up to about 80% compared to DSR-NC and up to about 40% compared to DSR-C, depending on the scenarios studied.

4.3.2 DSR-LC, effects of k and p_t

In this part, we will validate the claim that $k = \lfloor \frac{M}{2} \rfloor$ and $p_t = 0.4$ are optimal values by testing all the possible k and p_t values in a scenario with moderate caching availability. First, we fix p_t at 0.4 and change k from 1 to 4. From the results shown in Fig. 4.8, we can see that there is an optimal point for the selection of k. In the tested scenario, it is $k = 3$, which matches with our analytical result $k = \lfloor \frac{M}{2} \rfloor = \lfloor \frac{7}{2} \rfloor = 3$. Although the number of RREP also increases as k increases, this increase is not of the same order as the number of RREQ. In addition, the decrease of the packet delivery ratio at k =

3 is small (less than 1%). Another trend which is not shown but worth pointing out is the increasing path optimality with increasing k (the difference is also very small, however). With a larger local searching radius, it is more possible to find the target itself. Therefore, the path optimality increases correspondingly.

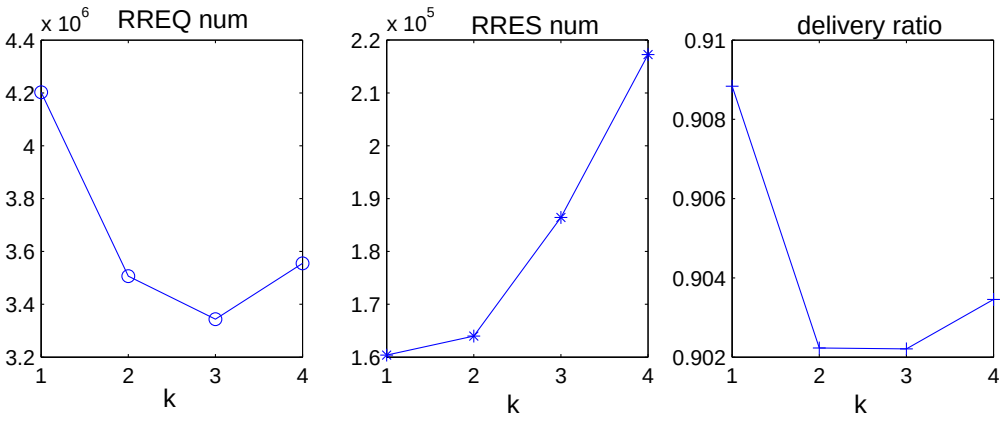

Figure 4.8: Performance of DSR-LC with p_t fixed at 0.4. The X-axis indicates the local searching radius k, ranging from 1 to 4. The optimal point is at $k = 3$ for the number of RREQ with almost no effect on other metrics.

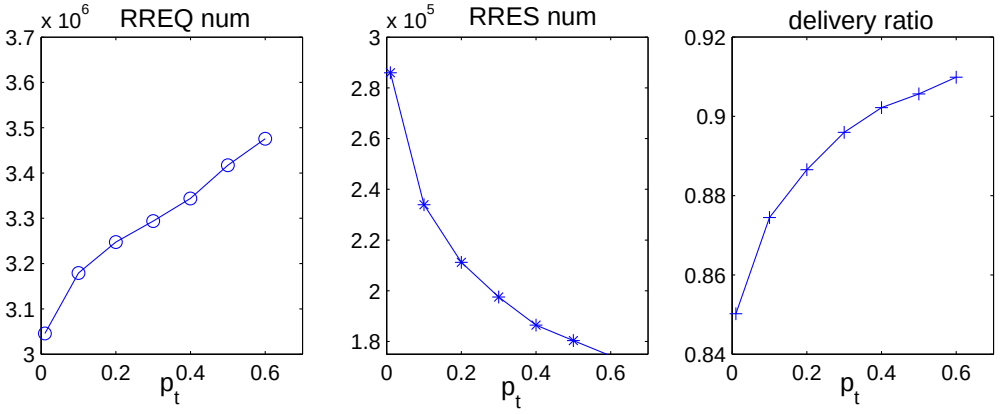

Figure 4.9: Performance of DSR-LC with k fixed at 3. The X-axis indicates the route caching validation probability threshold p_t, ranging from 0 to 0.6. The tradeoff is between the number of RREQ and the the number of RREP plus the delivery ratio. A good balance point is at $p_t = 0.4$.

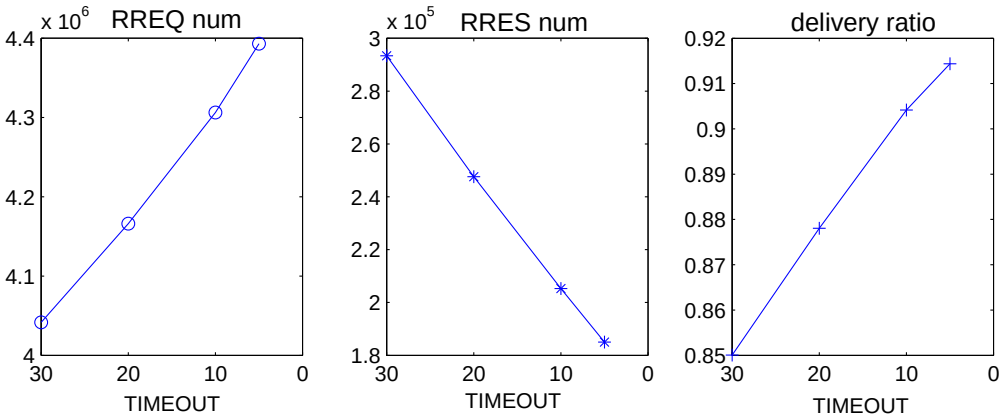

Figure 4.10: Performance of DSR-C with one-hop neighbor searching. The X-axis indicates the timeout value, ranging from 30 seconds to 5 seconds. The tradeoff is also between the RREQ number and the delivery ratio. Just like Fig. 4.9, the increase of TIMEOUT causes both metrics to decrease. A good balance point is at TIMEOUT=10 seconds.

Next we fix k at 3 and change p_t from 0 to 0.6. The simulation results are shown in Fig. 4.9. When the threshold p_t is set low, intermediate nodes tend to return a route cache and stop forwarding the query packet. Thus the number of RREQ packets is lower while the number of RREP packets becomes higher. However, the reduction of RREQ packets by lowering p_t may not be desirable because the packet delivery ratio also decreases. With a low p_t, the quality of the routes returned from intermediate nodes tend to be low. It is more possible to fail to deliver packets by using these less valid routes.

From Fig. 4.9, some point between 0.3 and 0.4 is a good balance point for p_t. Before this point, the increase of p_t leads to an approximately linear increase of RREQ while leading to a *faster* decrease of RREP and a *faster* increase of the packet delivery ratio. The knee of the curves of the RREP number and the packet deliver ratio is at around 0.4. Also, considering the delivery ratio to be larger than 90%, we choose p_t equal to 0.4 for the rest of the simulations. The study of p_t also provides us a method to trade the routing overhead for the packet delivery ratio by adjusting p_t.

So far, we have validated our analytic results by simulation. In the rest of this section, we will apply DSR-LC with an initial value of $k = 3$ and $p_t = 0.4$.

4.3.3 DSR-C, effects of timeout

We test DSR-C with the route cache timeout value of 5s, 10s, 20s and 30s, and the results are shown in Fig. 4.10. In order to compare the results with the selection of p_t in DSR-LC shown in Fig. 4.9, we reverse the order of the cache timeout values on purpose.

As shown in Fig. 4.10, there is a similar trend and tradeoff for the timeout value as there was for p_t shown in Fig. 4.9. The relationship between the timeout and p_t can be partially explained by equation 4.1. The larger the time for a route cache to be stale, the easier it is for a route request to be satisfied with certain cached routes. However, the sacrifice is a higher number of RREP packets and a lower packet delivery ratio due to stale routes. We pick the balance point TIMEOUT equal to 10s as the representative for DSR-C to ensure that the delivery ratio is larger than 90%.

4.3.4 DSR-LC, DSR-C and DSR-NC

In this part, we will compare these three routing schemes under different traffic rates, different node speeds, different event lifetimes and different traffic patterns.

First, we experiment with a low event rate of 0.05 events per second. We test the scenarios with the duplex [event lifetime T_l, maximum node speed S_m] valued at [2s, 10m/s], [10s, 10m/s] and [2s, 1m/s]. These scenarios can all be categorized as moderate caching availability. From the results shown in Fig. 4.11, DSR-LC achieves a significantly lower routing overhead for this low event rate, despite the fact that the savings may vary depending on the other parameters.

Next, we test an extreme scenario with abundant caching availability. The event rate is as high as 0.5 events per second and the maximum node speed is as low as 1m/s. As shown in Fig. 4.12, the overhead reduction ratio of DSR-LC in this abundant caching scenario is as high as about 80% compared to DSR-NC, thanks to the caches. DSR-LC eventually adjusts its local searching radius to around 1 and p_t to around 0.65. DSR-C may achieve a closer number of RREQ packets if it adjusts its timeout value to 40 seconds (shown in the fourth column) instead of 10 seconds. However, the number of RREP packets increases correspondingly since more route caches are available for returning. DSR-LC, with the adjustment of both k and p_t, restrains the number of both

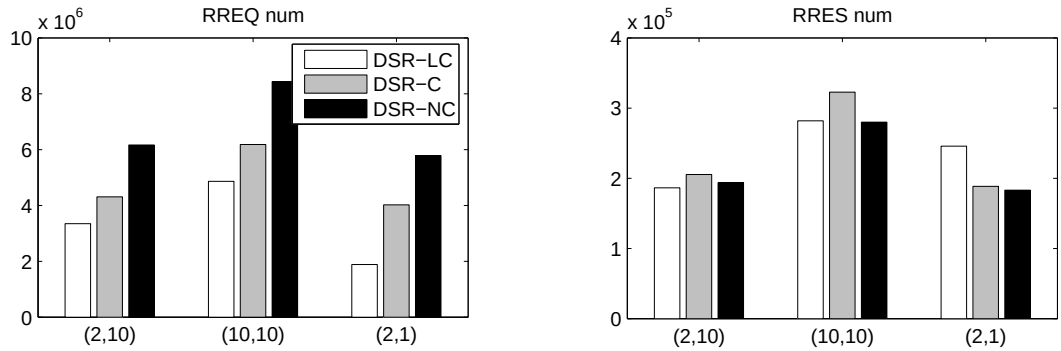

Figure 4.11: Routing overhead comparisons under a low event rate of 0.05 events/sec. The X-axis indicates different scenarios tested, from left to right stands for (event life, node speed) pairs of (2s, 10m/s), (10s,10m/s), (2s,1m/s). This figure shows the effects of event lifetime and the node speed on routing overhead in a moderate caching condition.

RREQ and RREP in a satisfactory range.

In the above experiments, each node has the same traffic pattern as other nodes and has events toward other nodes with equal probability. In contrast with this peer-to-peer traffic model, we experiment with a client-server traffic model. In this model, we choose 20% of the nodes as servers and 80% of the total traffic is towards these servers. The results shown in Fig. 4.13 are based on a maximum node speed of 1m/s and a total event rate of 0.05. As can be seen from this figure, the shift from a peer-to-peer model to a client-server model reduces the overhead reduction ratio of DSR-LC compared to DSR-C but increases the overhead reduction ratio of DSR-LC compared to DSR-NC. This is reasonable since the client-server model implies a more abundant cache availability. For this reason, DSR-LC eventually adjusts k to an average of around 1.3 in the client-server model, while it adjusts k to an average of around 2.5 in the peer-to-peer model. Thus, in the client-server model, DSR-C with local searching radius fixed at 1 is already close to the optimal value, and therefore, the number of RREQ packets in DSR-C is close to that in DSR-LC. However, for the same reason illustrated in the last paragraph, DSR-C has a large number of RREP packets when route caches are abundant.

Another metric that is not shown in the figures but worth mentioning is the number

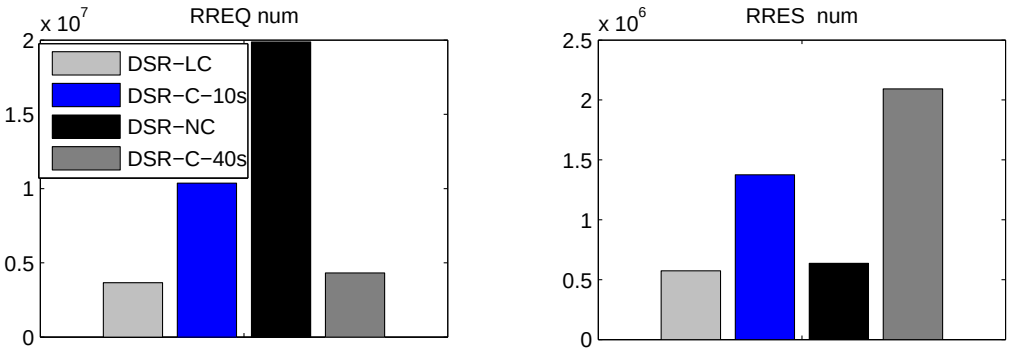

Figure 4.12: Routing overhead comparisons under a high event rate of 0.5 events/sec and a low maximum node speed of 1m/s. This figure shows the performance of different schemes in an abundant caching condition.

of forwarded data packets. DSR-LC shows up to 10% less data forwarding than DSR-C. This implies that the path chosen in DSR-LC is closer to the optimal path than DSR-C. That is because in DSR-LC, nodes can differentiate the quality of a route cache from the value of P_v and determine whether they should replace the old route cache or not. In DSR-C, only the route cache lifetime is checked, which cannot accurately reflect the real quality of a cached route.

If any of the caching optimizations, they just increase the caching availability and the quality of the caches, similar to how the client-server model increases the quality of the caches. Our routing scheme adjusts its parameters in response to the caching availability conditions.

Overall, DSR-LC can achieve an overhead reduction up to 80% compared to DSR-NC and up to 40% compared to DSR-C, depending on the caching level in the network. When the route cache availability is moderate, DSR-LC has a larger ORR compared to DSR-C. When route caches are abundant, DSR-LC has less overhead reduction in RREQ packets compared to DSR-C while it has much larger reduction compared to DSR-NC. Besides, DSR-LC can restrain the number of RREP packets by adjusting p_t without degrading cache qualities, while DSR-C does not have an effective method to control the number of RREPs.

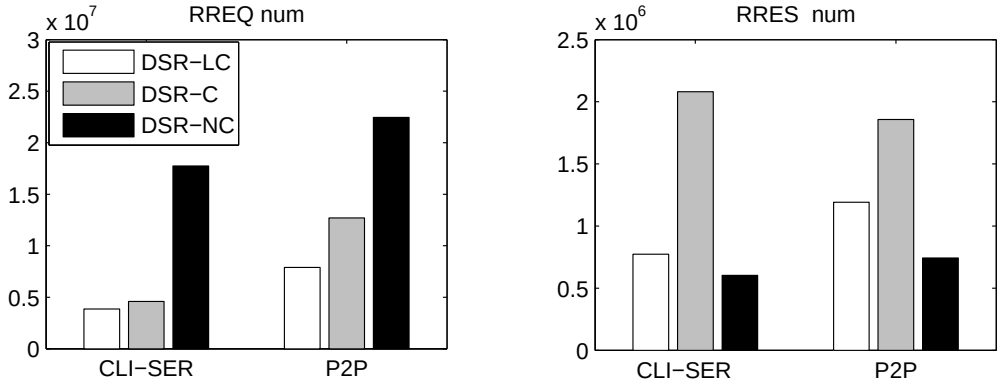

Figure 4.13: Routing overhead comparisons for peer-to-peer and client-server traffic models.

4.4 Conclusions

In this chapter, we studied peer discovery with route caches for mobile ad hoc networks. We proposed to adaptively adjust the local searching radius to the current caching condition. We also proposed a route cache quality measurement method to quantify the caching condition. We showed that routing overhead from route query flooding is greatly reduced and other routing performance metrics are also improved.

In the next chapter, we will shift our research on information retrieval from peer discovery to the area of data routing. Although the study in this chapter also deals with certain aspects of data routing, the goal is to reduce the cost/overhead of peer discovery. This is different from the goal of the data routing study in the next chapter, which is to discover “good routes” to improve the routing performance.

Chapter 5

Data Routing in Mobile Ad Hoc Networks

In this chapter, we switch our topic from information discovery to information retrieval, i.e., data routing. In mobile ad hoc networks, node mobility is the major factor that affects the performance of data routing. Since a link break from node mobility invalidates all the routes containing this link, alternate routes have to be discovered once the link is detected as broken. This new discovery phase incurs network-wide flooding of routing requests and extended delay for packet delivery. Furthermore, the upper transport layer may mistake this temporary route break as long term congestion and execute unnecessary backoffs. Since ad hoc routing protocols usually have their own retransmission scheme for route discovery, failure of synchronization between the routing and transport layers often occurs, resulting in poor overall performance.

Discovering long lifetime routes (LLRs) can reduce the impact of node mobility and improve the overall performance compared to using randomly chosen shortest-path routes. When a route with a longer lifetime is chosen, less frequent route discovery, which usually involves expensive network-wide flooding, is required, thus less routing overhead is incurred. The impact of long lifetime routes on upper layer protocols is also obvious. First, an LLR can reduce the chance of a route break, thus reducing the chance for abnormal TCP transmission behaviors observed in [27]. If two LLRs can be provided at a time, the routing protocol can save the longer LLR as a backup and use the shorter LLR, which usually has a shorter route length and is thus more energy-

efficient, as the primary route for transmissions. The routing protocol can switch to the longer LLR to maintain the flow of traffic when the shorter LLR breaks. Meanwhile, a new route discovery procedure can be initiated, and when the newly discovered LLRs are returned, the old LLRs can be replaced.

In this chapter, we first present a global LLR algorithm (g-LLR) that discovers the longest lifetime route at each different route length for a given pair of nodes. This algorithm requires global knowledge and provides the optimal LLRs for analysis. The study using g-LLR suggests to us that only LLRs with short route lengths are desirable since they can reduce route break frequency and there is no sacrifice on packet delivery efficiency from using more hops. Based on this principle, we propose a distributed Long Lifetime Route (d-LLR) discovery approach that finds two LLRs, termed as the primary LLR and the auxiliary LLR, in one best-effort discovery procedure. The primary LLR is the LLR at the shortest route length, and the auxiliary LLR is the LLR that contains one more hop than the shortest route. Simulations show that these two LLRs are very similar with the LLRs discovered using g-LLR and greatly improve the overall routing performance. Using these two LLRs, we also propose an intelligent fast-switch scheme that maintains continuous traffic flow for upper transport layers. This is crucial for reliable transport layer protocols and stream-based applications where the interruption of traffic may cause abnormal behaviors and deteriorate the overall performance.

5.1 A Review of Link Lifetime Estimation

From the literature, there are two general methods to quantify the quality of a link using link lifetime. The first method expresses link lifetime in a stochastic manner. A link break probability $P(t)$ indicates the probability that a link is broken at time t. Fig. 5.1 shows an example of $P(t)$ in a Gauss-Markov scenario from [23]. $P(t)$ is a non-decreasing function starting from 0. Obviously, as time elapses, the probability that a link will break increases.

The second method expresses link lifetime in a deterministic manner. Link lifetime can be estimated through the link break probability given an estimation rule, such as from now to when the link break probability is higher than a certain threshold. The quality of a link can be thus quantified using this estimated link lifetime. Link life-

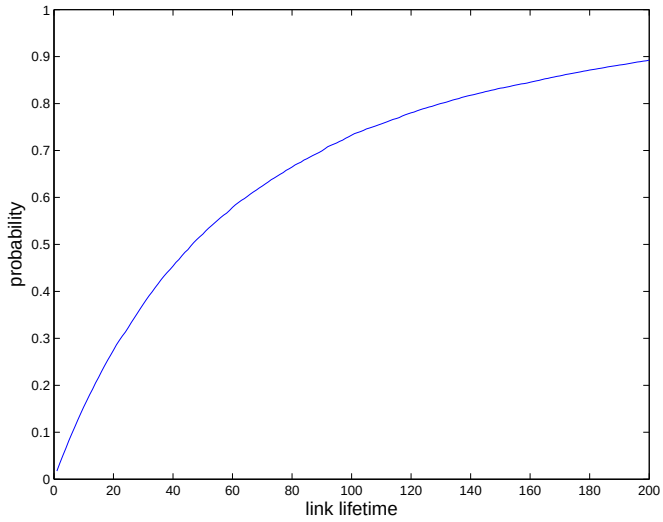

Figure 5.1: Link lifetime distribution in the Gauss-Markov scenario.

time can also be calculated using node location and movement estimated from signal strength or GPS. For practical protocol designs, link lifetime quantifications are necessary since it is much easier to append a value into a route message than to append an entire probability function.

Correspondingly, route lifetime can also be expressed in both manners. Suppose a route is composed of n links. Using link lifetime probability, the route lifetime distribution $P_r(t)$ can be calculated as

$$P_r(t) = 1 - \prod_{i=1}^{n}(1 - P_i(t)) \tag{5.1}$$

$P_i(t)$ indicates the probability for link i to be broken at time t. On the contrary, $1-P_i(t)$ indicates the likelihood for link i to be valid at t. The probability for all the n links to be valid at time t is $\prod_{i=1}^{n}(1 - P_i(t))$, and the probability for the route to be broken at t is one minus this value. On the other hand, using quantified link lifetime estimations, the route lifetime l_r is simply the minimum lifetime of the n links.

$$l_r = \min\{l_1, l_2, ..., l_n\} \tag{5.2}$$

LLR determines the route query forwarding delay based on the quantification of route lifetimes. To focus on the study of LLR, we assume that lifetime has been esti-

mated in a quantified manner, and it can be directly appended to routing messages as an additional field.

5.2 G-LLR: Global LLR Algorithm

The global LLR algorithm discovers the LLR at different route lengths for a given pair of nodes in a given network. The basic idea of g-LLR is to add the link with the longest link lifetime, then adjust the route length between each pair of nodes. Once the route length between two nodes changes, the new route is the LLR at the new route length, and the last added link lifetime is the route lifetime of this LLR. This step continues until all the links have been added to the network. Eventually, we have the LLRs at all the different route lengths.

Suppose we are interested in investigating the LLRs between the source node **S** and the destination node **D**. The arc set A is sorted in descending order by the lifetime $c[i, j]$ of the link composed of nodes i and j. We denote an edge as e or a link between node i and j as $e[i, j]$ if node i and j are connected. $d[i, j]$ is the hop distance between nodes i and j. d_{prev} is the last route length recorded between the pair. The g-LLR algorithm is shown in Algorithm 1.

We experiment with g-LLR in different scenarios with various network parameters such as node speed, node distance and network sizes. A general trend is discovered from the simulations: the lifetime of LLRs increases linearly with the route length of LLRs for non-stop random moving patterns [13], as shown in Fig. 5.2. Therefore, there is a certain tradeoff on whether to choose an LLR with short route lengths or to choose an LLR with long lifetime but longer route lengths. On one hand, an LLR with a short route length can deliver packets using fewer hops, thus reducing the packet delivery overhead. On the other hand, an LLR with a short route length also has a shorter route lifetime and breaks faster than longer LLRs, thus increasing the routing overhead from route discovery. Depending on traffic density and node mobility, LLRs with different route lengths should be chosen correspondingly. When traffic is heavy and node mobility is low, packet delivery overhead becomes the dominating factor and LLRs with short route lengths should be used. When traffic is light and node mobility is high, route discovery overhead becomes dominant and LLRs with long lifetimes should

Data: A, initial $c[i,j]$ for each link
Result: Record of the longest lifetime achievable for routes with different hop distances $\{d$[**S,D**], c[**S,D**]$\}$
begin
  $S := \emptyset; \overline{S} := A$; $d_{prev} = \infty$;
  for *all node pairs* $[i,j] \in N \times N$ **do**
    $d[i,j] := \infty$; $pred[i,j] := 0$;
  end
  for all nodes $i \in N$ **do** $d[i,i] := 0$;
  while $|S| \neq A$ **do**
    let $e[i,j] \in \overline{S}$ for which $c[i,j] = \max\{c(e), e \in \overline{S}\}$;
    $S := S \bigcup \{[i,j]\}$; $\overline{S} := \overline{S} - \{[i,j]\}$; $d[i,j] = d[j,i] = 1$;
    for *each* $[m,n] \in N \times N$ **do**
      if $d[m,n] > d[m,i] + d[i,j] + d[j,n]$ **then**
        $d[m,n] := d[m,i] + d[i,j] + d[j,n]$ and $pred[m,n] := i$;
      end
      if $d[m,n] > d[m,j] + d[j,i] + d[i,n]$ **then**
        $d[m,n] := d[m,j] + d[j,i] + d[j,n]$ and $pred[m,n] := j$;
      end
    end
    if d[***S,D***] $< d_{prev}$ **then**
      $d_{prev} = d$[**S,D**] and **record** $\{d$[**S,D**],c[**S,D**]$\}$
    end
  end
end

Algorithm 1: G-LLR algorithm.

be chosen.

For our distributed LLR design, we only attempt to discover LLRs with short route lengths. This is for several reasons. First, it is difficult to obtain LLRs with long route lengths in a distributed manner due to lack of global knowledge. Second, LLRs with long route lengths may outperform LLRs with short route lengths only in network scenarios with very light traffic and very high node mobility. Therefore, LLRs with short route lengths are more suitable for the majority of practical applications. Finally, errors during link lifetime estimation will eventually be reflected by route lifetime errors. The

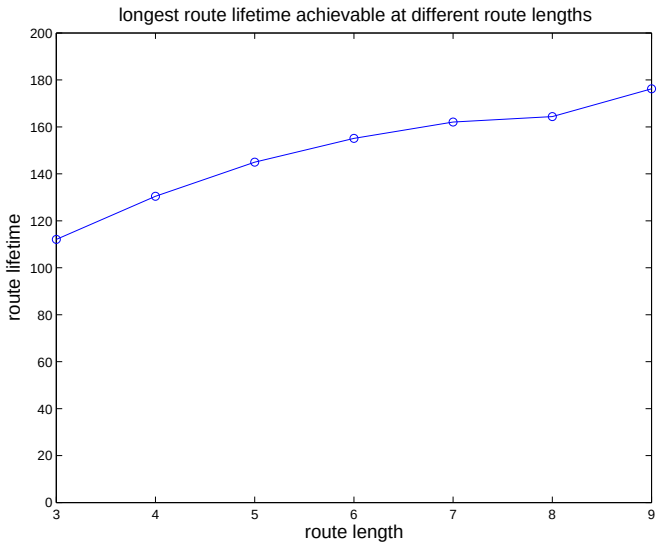

Figure 5.2: The longest lifetime achievable for routes of different route lengths. A linear tendency is shown.

longer a route is, the more likely the route lifetime is shorter than expected. Therefore, for our distributed LLR approach, we are only interested in LLRs with short route lengths.

5.3 D-LLR: Distributed LLR Algorithm

D-LLR can be used as an extension to most ad hoc routing protocols with minor modifications. D-LLR achieves two LLRs in a best-effort query procedure by intelligently associating the request forwarding delay with the currently observed LLR lifetime. The main procedure of d-LLR is similar to a typical on-demand routing protocol such as DSR: broadcast a route request from the source node and unicast a route reply message back from the destination node. The difference lies in the implementation details such as routing packet format, routing update rules and LLR-REQ forwarding delay rules.

5.3.1 General procedures

In d-LLR, LLR-REQ contains a primary route that expands while the LLR-REQ propagates throughout the network, similar to that in DSR. In addition, it contains an aux-

iliary route, which does not have any impact on LLR-REQ forwarding decisions. Also included in the LLR-REQ are the primary and auxiliary route lifetimes. These route lifetimes are calculated at each intermediate node by choosing the minimum from the composed estimated link lifetimes. Finally, an extra field that specifies propagation duration is included in the LLR-REQ. This propagation duration indicates the time since the LLR-REQ packet was transmitted by the source node, and it is used by intermediate nodes to calculate the local delay time for LLR-REQ forwarding.

The procedures of d-LLR are illustrated as follows. We focus on the differences with the procedures of DSR.

1. The source node broadcasts an LLR-REQ packet, which contains two routes: the primary LLR and the auxiliary LLR, with their respective lifetimes. The primary LLR is the LLR with the shortest route length, while the auxiliary LLR is the LLR that is one hop longer than the primary LLR. Initially, these routes only contain the source node and their lifetimes are set as 0.

2. When an intermediate node receives an LLR-REQ for the first time, it appends itself into the prim/aux routes in the packet and records the request locally. Then it adjusts the lifetimes by choosing the minimum of the previous route lifetime and its link lifetime with the previous node. Next, it schedules a local delay time for forwarding this modified LLR-REQ based on certain delay rules (see below). When the delay time is up, it forwards this LLR-REQ packet.

   If the intermediate node receives a duplicate LLR-REQ, it will update the prim/aux routes in its recorded LLR-REQ based on the LLR update rule (see below). Meanwhile, it reschedules the delay time if a better route is found and a shorter delay should be applied. The update rule and the delay rule will be explained in detail later. In brief, the delay rule requires routes with longer primary lifetime to have shorter delay, and the LLR update rule requires that the auxiliary route is longer than the primary route in both route length and route lifetime, and that the route length should be one-hop longer than the primary LLR.

3. The destination uses the same LLR update rules when receiving LLR-REQs from different paths. However, it simply waits enough time and then it will unicast an

LLR-REP to the source node using the primary route with the auxiliary route attached, just as in the normal DSR route response procedure.

5.3.2 LLR update rules

A node compares the routes in its current record with those in the newly arrived LLR-REQ packets. There are four routes involved in making the decision: the primary and auxiliary LLR in the local node's record and those in the newly arrived LLR-REQ. In brief, the node picks the one with the longest lifetime from the shortest routes as the primary route, and it picks the one with the longest lifetime from the second shortest routes as the auxiliary route.

In more detail, there are several cases. If an LLR-REQ arrives with a shorter primary route than the recorded primary route, the new primary route will be recorded as the primary route. The auxiliary route will be chosen from the recorded primary route and the newly arrived auxiliary route. If an LLR-REQ arrives with a primary route of the same route length as the recorded primary route, the primary route will be chosen from these two routes, and the auxiliary route will be chosen between the recorded and newly arrived auxiliary routes. If an LLR-REQ arrives with a longer primary route, only the auxiliary route will be chosen between the recorded auxiliary route and the newly arrived primary route. More details can be found in Algorithm 2.

5.3.3 Delay setup rules

To find both the primary and auxiliary routes in one discovery procedure, two key rules have to be observed when setting up the LLR-REQ forwarding delay. First, intermediate nodes with a longer primary route lifetime should forward earlier so that neighboring nodes will have a better chance of incorporating this route in their auxiliary route before they do their forwarding. Second, nodes at the same hop distance to the source node should forward at the same pace to reduce the chance of missing a primary LLR by forwarding the LLR-REQ too early. We illustrate how both rules help set up the primary and the auxiliary routes using Fig. 5.3 as an example.

In Fig.5.3, the number on each link indicates the link lifetime. After node **a** sends out the LLR-REQ, the ideal scenario is that when node **c** forwards the LLR-REQ, the

Data: pT_r, recorded primary lifetime;
aT_r, recorded auxiliary lifetime;
pT_n, newly arrived primary lifetime;
aT_n, newly arrived auxiliary lifetime;
pL_r, recorded primary route length;
aL_r, recorded auxiliary route length;
pL_n, newly arrived primary route length;
aL_n, newly arrived auxiliary route length;

begin

  if $pL_n < pL_r$ **then**

    $aL_r = \min\{pL_r, aL_n\}$ and adjust aT_r accordingly;

    $pL_r = pL_n$ and $pT_r = pT_n$;

  else if $pL_n = pL_r$ **then**

    $aL_r = \min\{aL_r, aL_n\}$ and adjust aT_r accordingly;

    $pT_r = \max\{pT_n, pT_r\}$;

  else

    $aL_r = \min\{aL_r, pL_n\}$ and adjust aT_r accordingly;

  end

end

Algorithm 2: LLR update rules at intermediate nodes.

LLR-REQ packet should contain the quadplex
<prim route, prim lifetime, aux route, aux lifetime>
as <[**a,b,c**], 3, [**a,d,e,c**], 4>. If the first rule is violated by forwarding earlier for shorter lifetime routes, then node **c** will broadcast the LLR-REQ with [**a,b,c**] before receiving the LLR-REQ from node **e**, thus missing the auxiliary route. If the second rule is violated by forwarding longer lifetime routes too fast, then node **c** may forward [**a,d,e,c**] before receiving the LLR-REQ from node **b**, thus missing the primary route.

The delay function also needs to avoid potential packet collisions from the MAC layer. Neighboring nodes are likely to determine their rebroadcast time based on the same primary lifetime value, especially when the primary route lifetime is determined by the earlier shortest link lifetime. In Fig. 5.4, nodes **b** and **c** receive the same primary

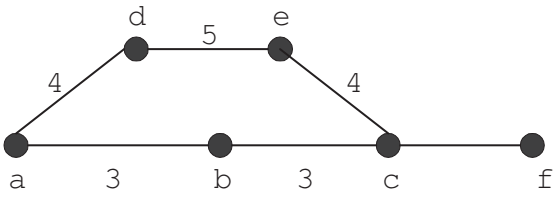

Figure 5.3: An example showing how the delay rules can help set up both the primary and the auxiliary routes.

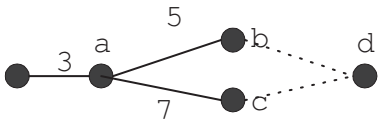

Figure 5.4: An example showing a potential collision using the same primary lifetime. Nodes b and c may collide when using primary lifetime 3 to calculate their forwarding delay.

route lifetime 3 from node **a** simultaneously. To avoid node **b** and **c** choosing the same delay time and colliding in their transmissions to node **d**, jittering should be introduced in the delay function.

In our design, a node chooses its overall delay t_d based on the following function.

$$t_d = f(l; D_1) + D_2 \times (h - 1) + U(D_3) \tag{5.3}$$

The first item follows the first delay rule, where l is the primary route lifetime and D_1 is the delay parameter for rule 1. Function f is a monotonic non-increasing function of link lifetime that determines the delay from 0 to D_1 based on the primary route lifetime. The second item indicates that nodes at the same hop distance h to the source node should broadcast approximately at the same time. D_2 is the second delay parameter, and it should be larger than D_1 so that nodes will not forward out of pace by the delay from D_1. The last item is the jittering to avoid collisions. D_3 is the third delay parameter, and $U(D_3)$ is a uniform distribution function from $[0, D_3]$. D_3 should be much smaller than D_1 so that it is unlikely to alter the rule that longer lifetime routes lead to shorter delay.

Notice that instead of local delay, t_d in equation 5.3 is the overall delay from when the source node starts the LLR-REQ to when the current node forwards the packet. Therefore, when each node forwards the packet, it should attach the current propagation duration in the LLR-REQ packet. With this information, the next node is able to calculate the local delay time by subtracting the propagation duration from the overall delay t_d. We do not include the initial LLR-REQ time in the LLR-REQ because this would require global clock synchronization among all the nodes in the network, which is difficult to implement.

5.3.4 Other design considerations

There are some design specific issues remaining to be discussed. First, what should be the values for the delay parameters D_1, D_2 and D_3, and what should be the delay function f? By choosing a larger value of D_1, we are able to better differentiate routes with different lifetimes. However, since D_2 has to be larger than D_1, the overall delay for the route discovery will increase correspondingly. Although choosing a smaller D_1 may lead to smaller discovery delay, an even smaller D_3 has to be chosen, which may increase the chance of collisions and missing some important routes. Thus, proper parameters have to be chosen.

For the delay function f, a simple choice is to associate the delay linearly decreasing with the link lifetime. However, since we are able to determine the route lifetime distribution through either statistical results or analytical results [13], we could make f biased on the most possible occurring lifetime spans instead of spreading evenly.

We tested different sets of parameters by ranging D_1 from 0.01 to 0.1 seconds, D_2 from 0.1 to 1 second, and $D_3 = 0.01D_1$. We also tested two delay functions: a simple linear function as shown in the left plot of Fig. 5.5, and a biased two-piece linear function as shown in the right plot of Fig. 5.5. We noticed that lifetime performance is not much affected by various parameter choices. The lifetime difference is within 2 seconds. Considering the fact that link lifetime estimation is erroneous in nature, this minor lifetime performance difference can be easily dominated by the error. Therefore, by default, we use D_1=0.05s, D_2=0.1s, D_3=0.0005s, and a linear function with a cutoff time L_t at 1000 seconds throughout the entire chapter.

A priority queue, similar to the one used in DSR, is utilized in our design. The

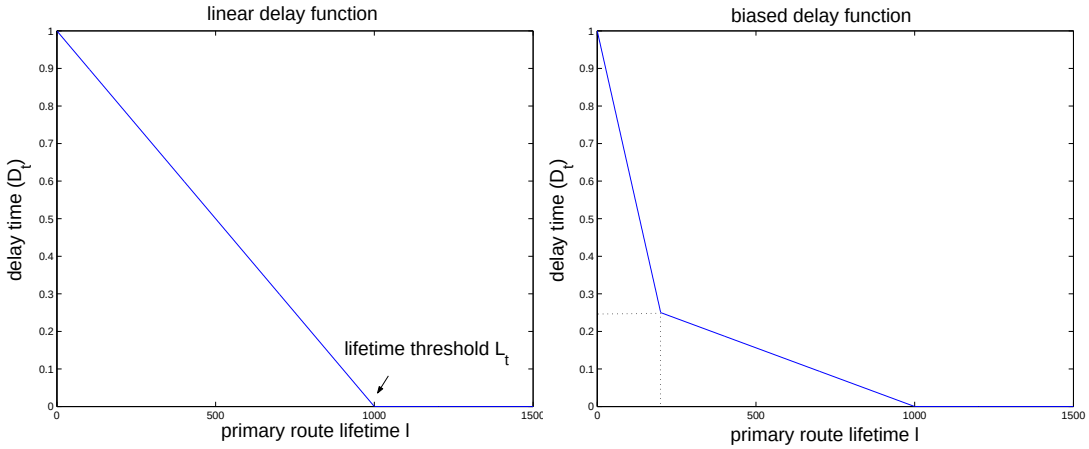

Figure 5.5: The delay function $f(l; D_1)$. On the left is a simple linear function. On the right is a biased linear function to concentrate on the first 200 seconds.

priority queue is a necessity for LLR, especially when there is data traffic. Without the priority queue, delay from the data in the queue will add up to the local delay of the LLR-REQ, which invalidates the delay rule that a longer lifetime route leads to shorter delay.

5.4 Performance Evaluation

In this section, we evaluate the performance of LLR through several groups of experiments. The first group of experiments focuses on evaluating the route lifetimes discovered using d-LLR with those discovered using g-LLR and those discovered using DSR. This informs us how close to optimal our distributed LLR scheme performs compared to a global algorithm, and how much lifetime improvement we can obtain over random shortest-path routing. The second group of experiments compares the general routing performance of d-LLR with that of DSR using UDP as the transport layer protocol. Since UDP is a semi-transparent transport protocol, we are able to demonstrate the direct advantage of LLR over random routes. The third group of experiments uses a more widely used transport layer protocol, TCP, and reinvestigates the performance of LLR. The last group of experiments tests the robustness and effectiveness of d-LLR by introducing errors to link lifetime estimations.

5.4.1 Lifetime performance

First, we compare the route lifetimes found using g-LLR, d-LLR and DSR. Our simulations are done using NS-2 [19]. We look at a fully connected network by randomly placing 100 nodes inside a network with a radius of 500m. The transmission range of each node is 250m. Nodes move with a speed uniformly distributed from $[0, 10m/s]$, according to the random waypoint model with no pause time. 802.11b is used as the MAC layer protocol and the wireless channel bandwidth is 2Mbps. We group the 100 nodes into 50 pairs and observe their initial route lifetime at time 0. We experiment using 10 different scenarios and average the results. Link lifetimes are calculated offline and input into each node based on node mobility pattern before the simulation.

We measure the route discovery success ratio and route lifetime for g-LLR, d-LLR and DSR. Route discovery success ratio (RDSR) indicates the likelihood that a route can be discovered when there exists one. Both d-LLR and DSR may fail to discover a route even if there exist routes between the observed pair of nodes. This is due to the potential collisions among flooded route request messages. Route lifetime is another metric we observe. For DSR, we examine the route lifetime of the returned random route. For LLR, the lifetimes to be examined are the primary lifetime (PL) and the auxiliary lifetime (AL).

The results are shown in Table 5.1. We found that for a total of 500 cases tested, DSR can successfully discover an initial route in one discovery attempt for only about 67% of the time. On the other hand, d-LLR discovers a route in one discovery attempt for 498 out of 500 cases, leading to a discovery ratio close to 100%. The offline lifetime results obtained using g-LLR show that routes exist between the observed pairs for all these cases. When we look into the two cases where d-LLR fails to find a route, however, we notice that in one case, the LLR ends at 0.44s, and in the other case, the LLR ends at 0.3s. When the LLR route is being returned by the destination, this route is already broken. That is why d-LLR fails to discover them in these two cases.

D-LLR is more capable of discovering routes than DSR because d-LLR uses an intelligent query forwarding delay scheme. By using a relatively longer delay time and a jittering scheme, d-LLR is able to greatly reduce collisions. Also, d-LLR ensures that nodes at the same hop distance rebroadcast at the same phase, thus propagating the query message in a more organized ring-out pattern. This further reduces the col-

lisions, especially when the query has been broadcasted a few hops away from the center. Although the initial route discovery delay may become longer, fewer collisions are incurred, and thus this one-time route set-up delay is still within the scale of a second. Once a good route is discovered, this delay will be effectively compensated by the extended traffic flow using the route.

As for the auxiliary LLR, d-LLR can find a route for 351 out of 500 cases, resulting in a discovery ratio of about 75%. Meanwhile, g-LLR shows that for 437 out of 500 cases, there exists an auxiliary LLR that is one hop longer than the primary LLR, and for 33 cases, g-LLR discovers LLRs that are more than one hop longer than the primary LLR. For the remaining 30 cases, only the primary LLR exits.

Table 5.1: Lifetime performance of DSR, g-LLR and d-LLR.

	RDSR	**PL(s)**	**AL(s)**
DSR	67%	28.3	none
d-LLR	100%	40.26	55.4
g-LLR	100%	40.24	60.9

The average route lifetime discovered by DSR is about 28.3 seconds. The average lifetime of the primary LLR discovered by d-LLR is 40.26s, while the average lifetime from g-LLR is 40.24s. The primary lifetime of d-LLR is slightly larger than that of g-LLR simply because d-LLR fails to discover the two LLRs with very small lifetime of 0.44 and 0.3 seconds. These two small lifetimes are not included in calculating the average lifetime, thus resulting in seemingly abnormal lifetimes for d-LLR. Therefore, d-LLR performs almost the same as g-LLR in terms of primary route lifetime. As for the auxiliary route, the average lifetime from g-LLR is 60.9s, while the average auxiliary lifetime from d-LLR is 55.4s, only 5 seconds less than that of g-LLR. Compared to DSR, d-LLR is able to discover a primary route that lasts 40% longer, and an auxiliary route that lasts 100% longer. The lifetime performance improvement, which occurs without sacrificing route length, directly leads to the routing performance improvement to be shown in the next section.

5.4.2 LLR routing performance with UDP

In this section, we use UDP as the transport layer protocol and investigate the routing performance of d-LLR and DSR. UDP is a semi-transparent best-effort transport protocol, and it feeds packets to the underlying routing protocol with little modification. A study using UDP provides direct insight into the advantage of LLR over DSR in terms of packet delivery ratio, packet delay and energy consumption.

We will experiment using d-LLR and DSR with various options to test the effect of each option. For d-LLR, we will investigate d-LLR with only the primary route used, denoted as LLR. We will also investigate a fast-switch LLR scheme with the auxiliary route reserved as a backup route. In this scheme, when the primary route is detected as broken, the auxiliary route will be used and a new route discovery procedure will be initiated. This auxiliary route continues to be used until a new route response is received. A new round starts where the new primary route will be applied, and the new auxiliary route will serve as the new backup route. We term this fast-switch scheme LLR-FS.

LLR attempts to discover the best route towards the destination. Therefore, intermediate nodes do not return route caches upon receiving a route request. This is because with high node mobility, route caches are likely to be stale and invalid. If a stale route cache is returned, a longer route discovery delay may occur. Similarly, only one route response needs to be returned by the destination node. This route response already contains the best-effort primary route and auxiliary route from the destination's point of view. As for DSR, we also remove the options of caching and multiple route replies. The removal of these options is in favor of DSR because these options incur long route discovery delay and make the connection between the source and destination unstable. We denote this DSR with no caching as DSR-NC.

Using the same scenarios as in the previous section, we feed the 50 pairs of nodes with traffic at a rate of one packet per second, and run each simulation for 300 seconds. The average performance of Packet Delivery Ratio (PDR), Packet Delivery Delay (PDD), and Energy Consumption Per Packet (ECPP) are shown in Table 5.2.

LLR achieves a packet delivery ratio of about 98%, while DSR-NC achieves about 95%. DSR-NC drops more packets due to more frequent and longer route recovery. LLR-FS has a packet delivery ratio of 98%, which is slightly lower than that of LLR.

Table 5.2: UDP performance using DSR and LLR.

	PDR	PDD(s)	ECPP(J)
UDP+DSR-NC	95.3%	1.226	0.15
UDP+LLR	98.3%	0.027	0.10
UDP+LLR-FS	98.0%	0.015	0.10

This is because LLR-FS may attempt to deliver a packet using an invalid auxiliary route due to erroneous link lifetime estimation. In the random waypoint model, a node may change its direction and thus either extend or decrease its link lifetime with its neighbors. However, earlier link lifetime estimation cannot predict this and have to estimate link lifetime based on the previous linear mobility pattern. Thus, both the primary and auxiliary route lifetime estimates may be erroneous. For LLR, when a primary route breaks, only one UDP packet is lost. Upon the next UDP packet, new routes will be discovered and the packet will be delivered successfully. For LLR-FS, however, the first lost packet only invalidates the primary route, while the auxiliary route still exists. The second packet will be lost if the auxiliary route is also invalid. Since a new route discovery is initiated simultaneously with the attempt of delivering the second packet, the third packet will definitely be delivered using new routes. Therefore, the packet delivery ratio of LLR-FS is slightly lower than LLR, at most one packet per route break.

However, if the auxiliary route of LLR-FS is valid, there will be no packet delivery delay from route discovery for the second packet and the rest of the buffered packets. From Table 5.2, the average packet delay is about 0.027s for LLR, and it is an even lower 0.015s for LLR-FS. With the fast-switch scheme, the delay is greatly reduced since there is almost no delay from route discovery, and the only delay is from packet forwarding. LLR-FS sacrifices very little packet delivery ratio for a significant delay improvement and thus achieves potential fast recovery for upper transport layer protocols.

DSR, on the other hand, has an average delay as large as 1.23 seconds. This long delay is mostly due to the delay from route discovery. When a RREQ message arrives at the destination, it is very likely that the local area has high contention from flooded RREQ messages. An ARP process may also be required if a node does not recognize

its neighbors. Therefore, the RREP message may not be returned to the source node in a timely manner, and the source node will back off its next route discovery attempt. Packets during the backoff will be buffered at the source node, and the delay for these packets eventually adds up if several route discovery attempts fail. For a similar reason, the energy consumption of DSR is about 50% more than that of LLR due to more frequent route breaks and more route discovery attempts.

5.4.3 LLR routing performance with TCP

The advantages of LLR can be further revealed when LLR is functioning with a more widely used transport layer protocol, TCP. Previous studies have shown that it is difficult to cooperate TCP with DSR without modifications to these protocols [27]. Packet flow may completely stop and become very hard to recover once a route breaks. The reason lies in both ends of TCP and DSR. For TCP, route breaks may be mistaken for network congestion, and thus TCP may perform an unnecessary slow start once it does not receive a TCP acknowledgement. For DSR, the excessive usage of cache may bring about very long route discovery delays. The combination of both drawbacks causes the TCP performance to be unacceptable for highly mobile scenarios.

Most solutions attempt to solve this problem through a cross-layer design by explicitly differentiating network congestion from link failure [27, 45, 69]. However, the problem can only be partially solved. If a TCP acknowledgement is lost en-route, the source node cannot learn of the link failure, and thus still has to initiate a slow start. Considering the fact that TCP/IP has existed for decades and is a solid and mature protocol, we believe that a more practical approach should start from the routing protocols themselves.

We do not intend to test all combinations of TCP and DSR approaches in this section. Instead, we test two fundamental options. The first option is the same no caching option as in the previous section, which is to completely remove caches from DSR to ensure instant route discovery. The second option is to forbid TCP to backoff when an acknowledgement is not received due to temporary route break, termed as TCP-NB (No Backoff). This is essentially the same idea as explicitly differentiating link breaks with network congestion. For d-LLR, we test the same basic LLR scheme and the advanced LLR scheme with fast switch LLR-FS.

The performance of these schemes at a packet rate of one packet per second is shown in Table 5.3. As we mentioned earlier, if the default TCP and DSR is used, TCP barely recovers from the unsynchronized backoff schemes with DSR. The performance under these circumstances is very poor as shown in the first row. The packet delivery ratio is only around 56%. Many packets are still buffered, waiting for a route to be discovered when the simulation ends.

In DSR-NC, we remove the caching technique and allow source nodes to discover a route that is guaranteed to be valid. With caching removed, TCP is better able to deliver packets, providing a 92% packet delivery ratio. If we further remove the slow start scheme from TCP, the packet delivery ratio is further improved to 99% in the scheme TCP-NB+DSR-NC. TCP probes periodically when a route breaks, and once a route is discovered, TCP can start its transmission with minimal waiting time. Therefore, almost every packet can be delivered at the end of the simulation. This better packet delivery performance results in a higher energy consumption since more efforts are made to discover routes and push the traffic flow forward.

Table 5.3: TCP performance using DSR and LLR with a traffic rate of one packet per second.

	PDR	**PDD(s)**	**ECPP(J)**
TCP+DSR	56.7%	0.035	0.307
TCP+DSR-NC	91.7%	0.118	0.467
TCP-NB+DSR-NC	99.4%	0.130	0.554
TCP+LLR	99.9%	0.031	0.413
TCP+LLR-FS	99.9%	0.020	0.426

On the other hand, d-LLR achieves high packet delivery ratio without any need to modify TCP. Since new good routes can be discovered immediately and routes do not break very often, TCP barely needs to back off. As a result, the packet delay is as low as 0.031 seconds, mostly just from packet forwarding. If we apply the fast-switch technique to LLR, the traffic becomes more continuous, and the packet delivery ratio becomes even higher. Packet delivery delay is further reduced since TCP no longer needs to wait for a route to be discovered. LLR-FS switches to an auxiliary LLR when

the primary LLR breaks, and when new packets arrive from TCP, new routes have already been discovered and the new primary LLR will be used for the rest of the transmissions until it breaks again. Similarly, energy consumption is less than DSR due to fewer route discovery attempts.

We also tested a higher packet rate of 10 packets per second. The results are shown in Table 5.4. When a larger packet rate is applied, the advantage of LLR becomes less obvious since more packets can be delivered once a route is discovered. These packets will smooth out and reduce the effect of high delay and energy consumption from route discovery. Nevertheless, we can still obtain better packet delivery, lower delay and better energy consumption using LLR. The advantage of continuous packet flows from LLR-FS over the basic LLR becomes more obvious since more packets have to be buffered in LLR when no fast switching is available.

Table 5.4: TCP performance using DSR and LLR with a traffic rate of ten packets per second.

	PDR	PDD(s)	ECPP(J)
TCP+DSR	58.8%	0.023	0.275
TCP+DSR-NC	88.7%	0.070	0.375
TCP-NB+DSR-NC	98.8%	0.065	0.414
TCP+LLR	97.5%	0.019	0.382
TCP+LLR-FS	99.7%	0.017	0.388

5.4.4 LLR performance with inaccurate link lifetime estimation

In the previous studies, even though a link lifetime can be altered by nodes changing directions, we assume that link lifetime is accurately estimated if there are no such unpredictable changes. However, link lifetime estimation based on signal strength inevitably introduces errors from wireless channels, even if nodes do not change their directions. In order to discover how LLR is affected by these errors, we introduce link lifetime errors into our link lifetime estimates and reinvestigate the performance of LLR.

We assume that the path loss of the signal strength follows the shadowing model from [16].

$$PL(d)[dB] = P_0(d_0) + 10n\log(\frac{d}{d_0}) + X_\sigma \tag{5.4}$$

In this equation, the path loss PL is related to three components. The first component, P_0, is the received power at a reference distance d_0. The second component is related to the distance d between the transceivers and the path-loss exponent n. This component indicates the general trend of attenuation with distance. Typically, n is assumed to be 2 for the free space model and 4 for the two-ray ground model. The last component, X_σ, is a zero-mean Gaussian distributed random variable (in dB) with standard deviation σ. X_σ describes the randomness from shadowing.

Fig. 5.6 shows an example of link lifetime estimation distribution using 7 signal samples. We choose $n = 2.7$ and $\sigma = 11.8$ from [55]. The X-axis is the estimated link lifetime referring to the normalized link lifetime. The Y-axis is the probability distribution of the estimated link lifetime. This figure can be explained as the link lifetime estimation error pattern, and it can be used to calculated the estimated link lifetime. For example, when the real link lifetime is 3, the link lifetime input to LLR could be 0.9 times the actual link lifetime, resulting in an estimated link lifetime of $0.9\times 30 = 2.7$s. Or, it could be 1.4 times the actual link lifetime, resulting in an estimated link lifetime of 4.2 seconds. However, according to the estimation distribution pattern, the likelihood of estimating the lifetime as 2.7 seconds should be about 7 times that of estimating the link lifetime as 4.2 seconds.

The performance of LLR with TCP using erroneous link lifetime estimation is shown in Table 5.5. As expected, the performance of LLR is degraded by the erroneous link lifetime estimation. The packet delivery ratio drops even lower than the DSR-NR scheme. The LLR-FS, however, is much more robust in maintaining a good performance. Despite the fact that the real lifetime of the primary LLR is likely to be lower than expected, the auxiliary route lifetime is still very likely to be larger than the primary route lifetime. Therefore, when a route breaks, the traffic flow still can be continuous.

LLR-FS actually provides an automatic tradeoff method between energy and link lifetime estimation accuracy. When lifetime estimation is accurate, LLR-FS is both energy-efficient and ensures continuous traffic flows. When lifetime estimation be-

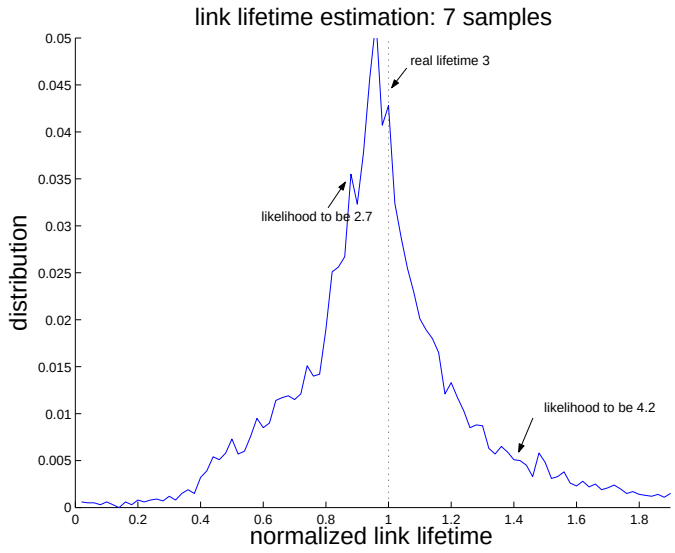

Figure 5.6: Link lifetime estimation error pattern. The X-axis indicates the normalized link lifetime estimation. The Y-axis indicates the probability distribution of lifetime estimation. For example, when the real lifetime is 3, the likelihood of estimation as 2.7 is 7 times that of estimation as 4.2.

comes inaccurate, LLR-FS still ensures continuous traffic flow, but it requires more frequent route searches. In other words, LLR-FS is self-adaptive to the accuracy of the current link lifetime estimation method. Therefore, it can cooperate with current lifetime estimation schemes to improve the overall ad hoc networking performance without sacrificing the integrity of the existing TCP protocols and networking stacks.

Table 5.5: TCP performance using LLR with inaccurate link lifetime estimation for different traffic rates.

	PDR	PDD(s)	ECPP(J)
TCP+LLR-1	98.7%	0.043	0.431
TCP+LLR-FS-1	99.0%	0.028	0.445
TCP+LLR-10	87.0%	0.022	0.374
TCP+LLR-FS-10	99.2%	0.021	0.388

5.5 Conclusions

In this chapter, we propose LLR, a long lifetime route discovery approach to improve the performance of ad hoc networks without modifying existing network stacks and protocols. A global LLR algorithm is proposed to study the statistical behavior of long lifetime routes, and a distributed LLR approach is proposed to implement LLR for practical design. Simulation results show that LLR is able to improve the performance of ad hoc networks with high mobility. LLR can take advantage of existing link lifetime estimation technologies. It automatically adapts to different estimation schemes by trading off energy consumption with link lifetime estimation accuracy. D-LLR can be applied to most ad hoc routing protocols as an extension, and it does not require any modification on TCP and existing network architecture.

Chapter 6

Data Routing in Wireless Sensor Networks

Data routing in wireless sensor networks is essentially different from that in mobile ad hoc networks. Rather than maintaining energy efficiency for each individual node as in mobile ad hoc networks, it is more important to maintain a balance of power consumption in sensor networks so that certain nodes do not die much earlier than others, leading to unmonitored areas in the network.

Previous research has shown that because of the characteristics of wireless channels, multi-hop forwarding between a data source and a data sink is often more energy efficient than direct transmission. Based on the energy model of a specific sensor node platform, there exists an optimal transmission range that minimizes overall power consumption in the network. When using such a fixed transmission range in general ad hoc networks, energy consumption is fairly balanced, especially in mobile networks, since the data sources and sinks are typically assumed to be distributed throughout the area where the network is deployed. However, in sensor networks, where many applications require a many-to-one (covergecast) traffic pattern in the network, energy imbalance becomes a very important issue, as a hot spot is created around the data sink, or base station. The nodes in this hot spot are required to forward a disproportionately high amount of traffic and typically die at a very early stage. If we define the network lifetime as the time when the first subregion of the environment (or a significant portion of the environment) is left unmonitored, then the residual energy of the other sensors at

this time can be seen as wasted.

Many network deployment and relative data routing strategies have been researched to extend network lifetime by mitigating the hot spot around the data sink and balancing the energy consumption within the network. Variable transmission power control is considered to reduce the burden of nodes in the hot spot by allowing nodes at farther distances to transmit over a long distance directly to the data sink [70]. Deploying relay nodes around the data sink to help forward the traffic is also considered in [11, 22]. In LEACH [25], sensors organize clusters to take advantage of data aggregation and variable transmission range power control. The focus of network deployment must shift to the data sink when the capability of sensors is limited. Multiple data sinks can be deployed to take over a certain sub-region of the entire area [25, 39, 68], or a mobile data sink roaming within the network can be deployed [33, 35, 40, 59, 66].

Despite these various strategies, there is no general data routing framework to evaluate the maximum lifetime obtainable by different network deployment strategies and to evaluate their actual deployment cost (i.e., monetary cost). In this chapter, we formulate the data routing problem for sensor networks and analyze the limits of network lifetime for different types of sensor network scenarios and corresponding network deployment strategies. Since applying a more complex strategy may introduce extra cost, we also provide a simple yet effective cost model to explore the cost tradeoff for using advanced solutions. Proposing a general data routing analysis framework for different sensor deployment plans and including the cost factor during analysis are the two major contributions described in this chapter.

6.1 Data Routing Models

Our goal in this work is twofold. First, we provide a general data routing model for different network deployment strategies to determine the optimal lifetime possible for that strategy. Then, in order to compare across different strategies, we determine a normalized lifetime and the corresponding cost to achieve a given lifetime goal. This will enable sensor network designers to select the most cost-efficient solution to meet a particular lifetime goal for their sensor network.

We begin by discussing several different strategies for sensor network deployment

and some assumptions we make in order to accommodate these strategies in our model.

6.1.1 Deployment strategy options

Several key parameters can be used to describe sensor network data routing strategies. These parameters include the following.

1. Sensor capabilities. In some cases, sensors have a non-adjustable transmission power and thus a fixed transmission range, while in other cases, sensors equipped with more advanced transceivers may vary their transmission ranges by using different transmission powers.

2. Base station options. Some sensor networks are deployed with a fixed base station that cannot change its physical location. However, another deployment option is to utilize a mobile base station that changes its physical location over time. A third option is to deploy multiple base stations, where each base station can collect all of the data for a period of time or some of the data for the entire network duration.

3. Initial energy assignment. The initial energy assignment for each sensor reflects how much freedom a sensor network deployment strategy has. When the deployment is in a controlled manner, nodes can be assigned different levels of initial energy depending on their locations and their roles in the network. For general sensor network deployments, however, we usually assume that the initial energy of all the sensors is the same. This might be true especially when sensors are manufactured in large quantities without differentiation.

4. Sensor locations. Similarly, the location of sensors, relay nodes and data sinks depend on how much freedom a sensor network deployment has. If the deployment is under full control, more sensors can be placed where energy is needed, and relay nodes can be placed in areas likely to receive the most traffic.

5. Traffic generation pattern. The traffic generation pattern is closely related to the sensing application. For environmental monitoring applications, e.g., temperature monitoring, sensors may generate samples at the same rate. The traffic

generation pattern is uniform in this type of network. For intruder detection applications where an intruder is expected to be detected at the farthest end from the base station, more traffic is expected to be generated at far distances. The traffic generation pattern is thus non-uniform in this case.

A good data routing strategy should resolve energy imbalance while maintaining high energy efficiency for a given network deployment. We list some potential sensor network deployment strategies in Table 6.1, labeled as DS_1 through DS_6. We do not intend to list every possible deployment strategy in Table 6.1, but rather merely to highlight the characteristics of these strategies and the design directions for data routing for these strategies.

A sensor network is deployed to provide a certain quality of service for a maximum lifetime using a minimum cost. Although the more complex deployment strategies listed in Table 6.1 may provide much longer network lifetimes, the extra cost of sensor hardware, base station hardware, and incurred deployment complexity may lead to a disproportionate increase in deployment cost. While maximizing network lifetime is most often the desired research goal, the ultimate goal for a real sensor network deployment plan is to reduce network deployment cost per network lifetime without sacrificing quality of service. Therefore, cost must be considered along with network lifetime during the analysis of different deployment strategies.

Table 6.1: Sensor network deployment strategies, the corresponding scenarios and the potential difficulty.

Strategies	`Scenario:` {**Traffic, Sensors, Energy, Sink**}	*Difficulty*
DS_1: **Power control**	{uniform, uniform, uniform, single/static}	*Complex transceiver*
DS_2: **Mobile base station**	{uniform, uniform, uniform, **single/mobile**}	*BS mobility*
DS_3: **Multiple data sinks/clustering**	{uniform, uniform, uniform, **multiple/static**}	*Extra BS*
DS_4: **Non-uniform energy assignment**	{uniform, uniform, **non-uniform,** single/static}	*Individual energy assignment*
DS_5: **Non-uniform relay/sensor nodes**	{uniform, **heterogeneous,** uniform, single/static}	*Sensor/relay placement*
DS_6: **Non-uniform traffic**	{**non-uniform,** uniform, uniform, single/static}	*Case dependent*

6.1.2 Assumptions

To generalize the data routing model and obtain a true upper bound on network lifetime for different deployment strategies, we have made several simplifications. Specifically, we focus on energy dissipation through wireless communications rather than sensing, and we ignore the potential overhead from multiple network layers by assuming perfect scheduling. These assumptions enable us to evaluate these strategies at a high level.

First, we assume that the power consumption of sensor nodes is dominated by communication costs, as opposed to sensing and processing costs. This assumption is reasonable for many types of sensor nodes that require very little energy, such as pressure and temperature sensors. We also ignore the overhead that would typically be introduced by the routing layer. However, for long lasting sensor networks with little or no mobility, route updates should be performed infrequently and should not significantly affect the overall power consumption in the network. We have also ignored any potential overhead at the MAC layer. Due to the scarce energy supplies in sensor nodes, TDMA scheduling is commonly proposed for use in the MAC layer of sensor networks. Because of the low data rates expected in many sensor network applications, even localized TDMA scheduling (as opposed to globally coordinated scheduling) should not induce much communication overhead in the form of collisions and necessary retransmissions. Furthermore, TDMA scheduling can eliminate most overhead introduced by idle listening and overhearing. As with the overhead associated with routing updates, the establishment of schedules can take place very infrequently and should not contribute significantly to overall power consumption. Finally, we assume that the channels are lossless. Although lossy channels will induce retransmissions for reliable data delivery, they have the same effect on all strategies and do not affect the relative lifetime performance of these strategies.

6.2 Generalized Data Routing Model for Lifetime and Cost Analysis

We propose a generalized data routing model for both lifetime and cost analysis to provide a complete network deployment strategy evaluation. During lifetime analysis,

we determine the maximum network lifetime and normalized lifetime for a given network deployment strategy. This reveals the potential energy efficiency of a network deployment strategy. During cost analysis, we determine the overall monetary cost of a network deployment strategy. We include the extra cost of using a more complex deployment strategy and evaluate whether the improved energy efficiency is worth the extra cost incurred.

For quick reference, Table 6.2 lists the parameters we use in this chapter. These parameters include general network parameters, the key parameters of different network deployment strategies, the variables used during our lifetime and cost analysis, and the power model parameters.

6.2.1 Data routing model

In our data routing model, a set of N sensors is deployed in a region in order to monitor some physical phenomenon. We refer to the complete set of sensors that has been deployed as $S = \{s_1 \dots s_N\}$. Node i generates traffic at a rate of $r_g(i)$ bits per second. All these data must eventually reach a single data sink, labeled s_0. The power consumption model is adopted from [25] where the amount of energy to transmit a bit can be represented as $E_t = E_e + \epsilon d^{\alpha}$, and the amount of energy to receive a bit can be represented as $E_r = E_e$. E_e represents the electronics energy, ϵ is determined by the transmitter amplifier's efficiency and the channel conditions, and α represents the path loss exponent.

We adopt a common lifetime definition as the time when the first sensor dies. This lifetime definition, proposed in [7], is widely utilized in the sensor network research field. The purpose of our model is to discover the data routing pattern to achieve maximum network lifetime L given a fixed deployment strategy and network scenario parameters. We assume that the network scenario, which includes parameters such as traffic generation pattern $r_g(i)$, node distances $d(i, j)$ and the maximum transmission distance d_{max}, is known. The model is able to determine the maximum network lifetime L by discovering the amount of traffic each sensor should distribute to the other sensors in order to balance energy dissipation among the sensors. This traffic distribution is denoted by $t(i, j)$, indicating the traffic that sensor s_i transmits to sensor s_j.[1]

[1]Note that $t(i, i) = 0 \; \forall \; i$.

Table 6.2: Key notations used throughout this chapter.

GENERAL	
N	Total number of sensors
s_i	Sensor i, $i \in \{1, \cdots, N\}$
s_0	Base station
λ_a	Minimum sensor coverage density
λ_e	Energy density
A	Network area
KEY PARAMETERS	
$r_g(i)$	Traffic generation rate of sensor i
$E(i)$	Initial energy of sensor i
d_{max}	Maximum transmission distance
LIFETIME ANALYSIS	
$e(i)$	Energy consumption of sensor i
$t(i,j)$	Amount of traffic that sensor j forwards for sensor i
T	Matrix representation of $t(i,j)$
$d(i,j)$	Distance between sensors i and j
$d_t(i,j)$	Transmission distance between sensors i and j
L	Sensor network Lifetime
$\tilde{L}$	Normalized sensor network lifetime
COST ANALYSIS	
C	Overall deployment Cost
C_s	Cost of sensors
C_e	Extra cost
POWER MODEL	
E_e	Energy consumption from electronic overhead
E_t	Energy consumption for each bit transmitted
E_r	Energy consumption for each bit received
ϵ	Transmitter amplifier coefficient
α	Path loss exponent
E_T	Total energy

During network lifetime L, sensor s_i will generate a total of $r_g(i)L$ traffic. The first constraint of our problem, related to the conservation of data flow at all sensor nodes, is

$$\sum_{j=1}^{N} t(j,i) + r_g(i)L = \sum_{j=1}^{N} t(i,j) + t(i,0) \qquad \forall i \in \{1, \cdots, N\} \tag{6.1}$$

Equation 6.1 says that all the traffic received at s_i plus the traffic generated at s_i must be transmitted to other sensors s_j or to the data sink, s_0. The energy expense for sensor s_i is from both transmitting and receiving data, and it can be expressed as

$$e(i) = \sum_{j=1}^{N} E_e t(j,i) + \sum_{j=0}^{N} [E_e + \epsilon d_t^{\alpha}(i,j)] t(i,j) \tag{6.2}$$

Therefore, the second constraint, related to the initial energy at each sensor, $E(i)$, is,

$$e(i) \leq E(i) \qquad \forall i \in \{1, \cdots, N\} \tag{6.3}$$

The third constraint, related to the maximum transmission range d_{max} of each sensor, depends on whether nodes are capable of transmission power control. If so, then the transmit power required to deliver a packet from sensor s_i to s_j will be controlled in such a way that the transmission distance $d_t(i,j)$ equals the physical distance $d(i,j)$ as long as $d(i,j) \leq d_{max}$.

$$d_t(i,j) = d(i,j) \qquad \text{if } d(i,j) \leq d_{max} \qquad \forall i \in \{1, \cdots, N\}, \forall j \in \{0, \cdots, N\} \tag{6.4}$$

If nodes must use a fixed transmission range d_{max}, then the constraint simply becomes

$$d_t(i,j) = d_{max} \qquad \text{if } d(i,j) \leq d_{max} \qquad \forall i \in \{1, \cdots, N\}, \forall j \in \{0, \cdots, N\} \tag{6.5}$$

Note that if $d(i,j) > d_{max}$ then there is no link between sensors s_i and s_j (i.e., $t(i,j) = 0$).

The last constraint, related to the energy distribution at each sensor, depends on whether energy can be freely assigned to each sensor. If so, then the total energy consumption of all the sensor satisfies

$$\sum_{i=1}^{N} E(i) = E_T \tag{6.6}$$

If sensors are initially assigned the same amount of energy, then

$$E(i) = \frac{E_T}{N} \qquad \forall i \in \{1, \cdots, N\} \tag{6.7}$$

The optimal network lifetime can be obtained using a linear programming approach that sets the constraints as in Equations 6.1, 6.3, 6.4 (or 6.5) and 6.6 (or 6.7), and sets the goal of maximizing L. The linear program finds the maximum lifetime L for a given scenario, and it also discovers the traffic distribution matrix $t(i, j)$, indicating how this lifetime can be obtained through intelligent traffic distribution.

6.2.2 Lifetime analysis

While the lifetime L found in the previous section allows us to determine an absolute maximum time that the network can operate, this value is highly dependent on the network scenario parameters, including the network area, the required density of active sensors, the energy density, and the data generation rate. In order to obtain a more general understanding of the energy efficiency of different deployment strategies, we propose a normalized network lifetime $\widetilde{L}$, which measures how many total bits can be transported on the network per unit of energy. Similar sensing tasks should result in the same normalized network lifetime for a given sensor network deployment strategy.

A typical sensing task can be described as the requirement to monitor an area and to provide a certain quality of service for a certain period. For example, suppose we want to monitor the temperature of a region for one year with a temperature sample rate of once per hour. Design parameters of this task include the area of the region (A), the average traffic generation rate among active sensors ($\bar{r_g}$), and the monitoring period or network lifetime (L). These parameters affect the absolute lifetime, and they should be factored out during the calculation of the normalized network lifetime.

In sensor networks, the minimum number of sensors that are required to cover an area can be calculated. We denote this minimum sensor coverage density as λ_a. However, more sensors may be deployed, and they can rotate their sensing time while maintaining the same network coverage [21, 51, 63, 67]. Once the network is fully covered, network lifetime can be arbitrarily increased by simply putting more energy into the network. This can be realized by scaling up the deployed sensor density, or increasing the initial energy per sensor. Network lifetime can also be increased by reducing

the traffic generation rate among active sensors. Therefore, a normalized lifetime $\widetilde{L}$ that accounts for the total energy consumption by considering the above factors can be expressed as

$$\widetilde{L} = L(\frac{\bar{r_g}\lambda_a}{\lambda_e}) \tag{6.8}$$

where λ_a represents the minimum sensor coverage density, $\bar{r_g}$ represents the average bit rate among active sensors[2], λ_e represents the energy density of the network, and L is the lifetime achievable with the given scenarios parameters.

In terms of units, L is measured in seconds, $\bar{r_g}$ is measured in bits per second, λ_a is measured in the number of sensors per meter squared, and λ_e is measured in Joules per meter squared. $\widetilde{L}$ is thus measured in terms of bits per Joule, which explicitly indicates the energy efficiency of a particular deployment strategy for a given network scenario (area to be monitored). $\widetilde{L}$ excludes factors such as traffic generation rate and sensor density, and thus it enables us to compare the energy efficiency of different deployment strategies.

6.2.3 Cost analysis

Normalized lifetime only reflects the energy efficiency of different deployment plans. However, there are certain hidden costs associated with different deployment strategies in addition to the normal sensor deployment costs. Our cost analysis explores this hidden cost that is oftentimes overlooked, and it enables the evaluation of different deployment strategies from a complete monetary cost perspective.

In order to fairly evaluate and compare different deployment strategies, we fix all the network scenario parameters and determine the cost for a particular deployment strategy to meet a target lifetime goal. We denote by $\mathbf{P}$ the network parameters for a particular scenario, such that $\mathbf{P}$ is the parameter set tuple $[A, \bar{\mathbf{r_g}}, \lambda_\mathbf{a}]$, where A is the network area, $\bar{\mathbf{r_g}}$ is a vector representing the data generation rate of each active sensor, and λ_a is the required sensing density to meet the application quality of service requirements. For a particular deployment strategy DS_i, given $\mathbf{P}$ and a target network lifetime goal L, we can calculate the number of required sensors N as follows.

[2]Note that $\bar{r_g}$ is different from $r_g(i)$ in that $r_g(i)$ represents the overall traffic generation rate of sensor i, while $\bar{r_g}$ represents the average traffic generation rate when a sensor is active.

$$N(DS_i) = \begin{cases} \frac{L\bar{r_g}\lambda_a A}{\widetilde{L}E} & i = 1, 2, 3 \\ \lambda_a A & i = 4, 5 \end{cases} \tag{6.9}$$

This tells us how many sensor nodes must be deployed for a particular deployment strategy DS_i given network scenario $\mathbf{P}$ in order to meet the quality of service goal set by λ_a and the lifetime goal L. For deployment strategies DS_1, DS_2, and DS_3, where each node has a uniform data generation rate r_g and a uniform initial energy E, Equation 6.9 determines the number of sensors that are needed based on the normalized lifetime $\widetilde{L}$, while for deployment strategies DS_4 and DS_5, Equation 6.9 specifies that the minimum number of sensors that support the application quality of service (sensing density) should be deployed since unequal energy (DS_4) and deployment of relay nodes (DS_5) can be used to ensure that the lifetime goal is met[3].

More intelligent deployment strategies will have a higher normalized lifetime (i.e., they will be more energy-efficient and carry more traffic per unit of energy) and thus have a lower number of sensors $N(DS_i)$ that need to be deployed to meet the target lifetime. Thus, the deployment cost from sensors, $C_s(DS_i)$, is lowered. However, these complex strategies may have higher extra deployment cost $C_e(DS_i)$. Furthermore, for different deployment strategies, the cost for normal sensors $c_s(DS_i)$ will be different. For, example, if non-uniform initial energy is needed (deployment strategy DS_4), then the cost to manufacture the sensors will be higher and vary between sensors. The total cost for the sensors is $C_s(DS_i) = c_s(DS_i)N(DS_i)$, and the overall deployment cost $C(DS_i)$ becomes

$$C(DS_i) = C_e(DS_i) + C_s(DS_i) \tag{6.10}$$

Our cost analysis is simple yet effective, and it allows network designer to compare different deployment strategies on an equal basis.

[3]The number of sensors can always be scaled up to meet the lifetime goal when sensor energy capacity is limited.

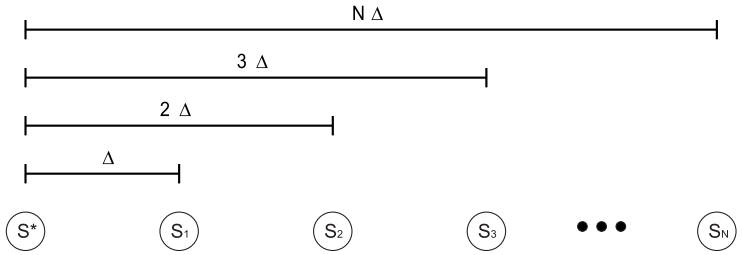

Figure 6.1: A one-dimensional regular spaced topology network. Nodes are equally spaced in this scenario.

6.3 A Case Study for the Simplest Deployment Strategy: DS_1

We begin our data routing study for the simplest, most common sensor network deployment scenario, DS_1 in Table 6.1. For this deployment strategy, the only option to reduce the effects of the hot spot problem and maximize network lifetime is to employ transmission power control. We are interested in discovering how the optimal lifetime, found through intelligent data routing, is affected by network size and the maximum transmission range.

We start our study with a one-dimensional network deployment, which may occur in such applications as highway traffic congestion monitoring or boundary monitoring. In this network, nodes are separated by a distance of δ, leading to the data sink, as depicted in Fig. 6.1. We assign nodes the same initial energy $E(i) = 1$ Joule and the same traffic generation rate $r_g(i) = 1$ bit per second. In all the simulations and analysis, we use values of $E_e = 50\ nJ/bit$ and $\epsilon = 100\ pJ/bit/m^2$.

To obtain the maximum achievable network lifetime, we first assume that nodes can adjust their transmit power large enough to cover the entire network. This is implemented in our data routing model by setting $d_{max} = \infty$. Equations 6.1, 6.3, 6.4 and 6.7 are the constraints for the linear program, and the linear program provides us the maximum achievable network lifetime for this strategy, which is shown in Fig. 6.2 using circles. If we maintain the same node density/spacing and increase the network radius from 75m to 250m by adding more nodes to the network, the lifetime decreases from about 4.3×10^6 seconds to about 9.9×10^5 seconds, as shown in Fig. 6.2.

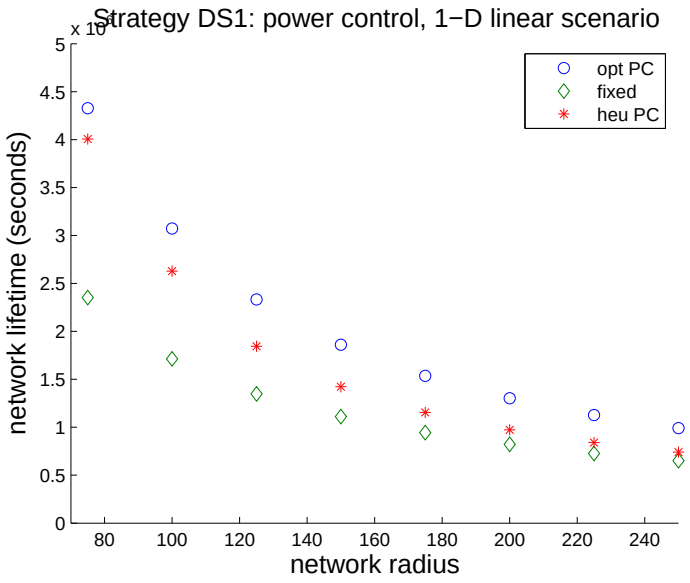

Figure 6.2: Network lifetime as a function of network radius for the optimal power control scheme, the fixed transmission power scheme and the heuristic power control scheme in a one-dimensional scenario. X-axis indicates network radius, and Y-axis indicates the maximum achievable network lifetime. “Opt PC” represents the optimal power control scheme, “fixed” represents the fixed transmission power scheme, and “heu PC” represents the heuristic power control scheme.

Although the traffic distribution matrix T is not shown here, a general trend on data routing can be observed from it. Nodes that are very close to the base station simply transmit directly to the base station. Nodes at farther distances transmit most of their packets over multiple hops and send the rest of their packets directly to the base station over long distances. The amount of packets sent over long distances decreases as nodes are farther from the base station. For more details on the analysis of the traffic distribution, readers are referred to [49].

In a real network, the optimal power control scheme would introduce large interference over long distances and make scheduling difficult. It also requires much more complex transceivers to be installed on sensors. In many scenarios, sensors are equipped with transmitters of fixed transmitting powers. To investigate the performance of this fixed transmit power scheme, we use Equations 6.1, 6.3, 6.5 and 6.7 as the con-

straints and rerun the simulations. The results are shown using diamonds in Fig. 6.2.

Power control scheme with unlimited transmission coverage is unrealistic, and thus is only of theoretical value. To avoid interference from large transmit powers and to reduce the complexity of scheduling protocols, we propose a heuristic power control scheme that allows nodes to transmit using a fixed limited transmission power for forwarding transmissions, while they use transmission power control for the last-mile transmission when they are able to send their traffic directly to the base station. Translated into our model, the third constraint is modified to: for $\forall i \in \{1, \cdots, N\}$ and $\forall j \in \{1, \cdots, N\}$

$$\begin{aligned} d_t(i,j) &= d_{max} \qquad \text{if } d(i,j) \leq d_{max} \\ d_t(i,0) &= d(i,0) \qquad \text{if } d(i,0) \leq d_{max} \end{aligned} \tag{6.11}$$

The lifetime performance of this heuristic power control scheme is shown in Fig. 6.2 using stars. The optimal lifetime for the fixed transmission range scheme and the heuristic power control scheme is obtained through brute force on all possible transmission ranges. By comparing the optimal power control scheme (legend as "opt PC"), the fixed transmission range scheme (legend as "fixed") and the heuristic power control scheme (legend as "heu PC",) we can see that the heuristic power control scheme performs somewhere between the optimal power control scheme and the fixed scheme. It always performs better than the fixed transmission range scheme due to the energy savings at the last hop to the base station. However, the improvement from the optimal power control scheme compared to the heuristic power control scheme is not significant.

The above conclusions are drawn based on one-dimensional regularly spaced networks. We have also simulated two-dimensional networks using a circular grid scenario as shown in Fig. 6.3. Such two-dimensional networks can be modeled as a one-dimensional field with nonuniform spacing. With very dense sensor deployment, we can assume that sensors will always send their packets within an infinitesimally thin angle toward the data sink, as shown in Fig. 6.4(a). Since the number of nodes N within the distance r from the data sink satisfies $N \propto r^2$ for two-dimensional networks, when mapped onto a one-dimensional space, the distance of a node to the data sink should be proportional to the square root of the node index, as shown in Fig. 6.4(b).

Energy consumption imbalance becomes even more serious in two-dimensional net-

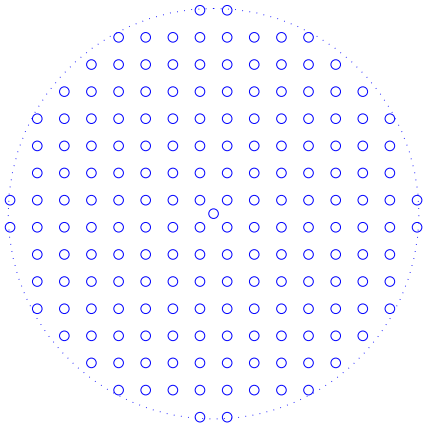

Figure 6.3: The 2-D circular grid topology. 180 nodes are placed within the circle area in this plot.

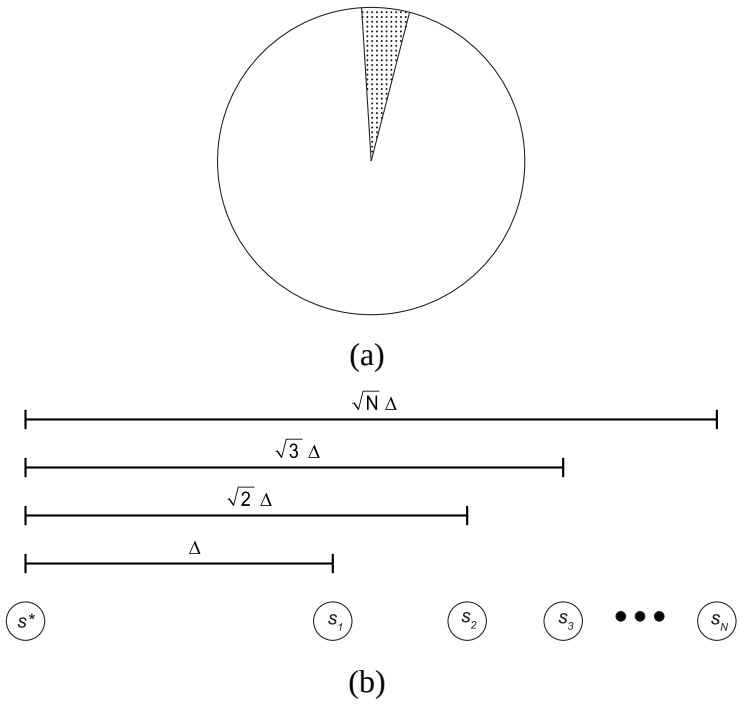

Figure 6.4: Two-dimensional sensor field (a) and its one-dimensional modeling (b).

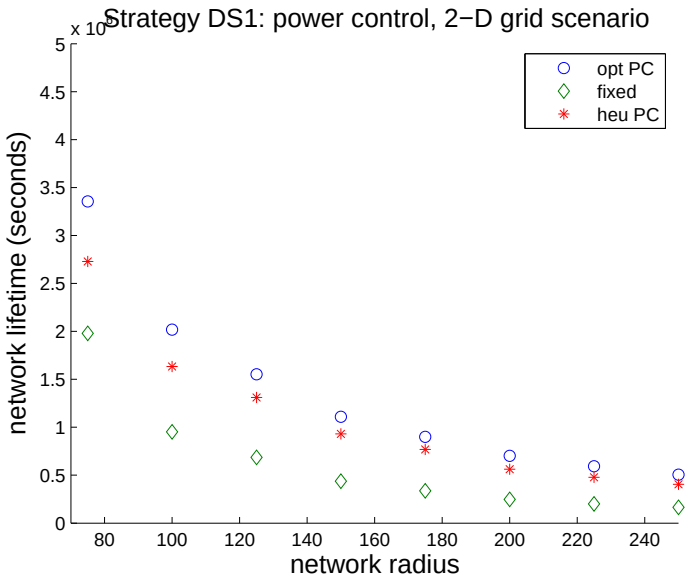

Figure 6.5: Network lifetime as a function of network radius for the optimal power control scheme, the fixed transmission power scheme and the heuristic power control scheme in a two-dimensional scenario. X-axis indicates network radius, and Y-axis indicates the maximum achievable network lifetime. “Opt PC” represents the optimal power control scheme, “fixed” represents the fixed transmission power scheme, and “heu PC” represents the heuristic power control scheme.

works since more nodes are located far from the base station, as can be seen from its 1-D modeling in Fig. 6.4 (b). In the optimal solution, more packets are transmitted over long distances to use up the residual energy of nodes far from the base station. As can be expected, the lifetime performance improvement from the optimal power control to the heuristic power control scheme is more limited in this scenario. Fig. 6.5 proves our conjecture by comparing the performance of the optimal power control scheme, the fixed transmission power scheme and the heuristic power control scheme for two-dimensional grid networks. The heuristic scheme performs very close to the optimal power control scheme, especially when the network radius is large.

In general, the network lifetime improvement from the optimal power control scheme is not significant compared to a heuristic scheme with transmission powers fixed most of the time. This is especially true when energy imbalance becomes worse such as

when the network size increases or when utilizing two dimensional networks.

Early studies [9, 18] show that the ideal energy-efficient transmission range should be set as $d^* = \sqrt[\alpha]{\frac{2E_e}{(\alpha-1)\epsilon}}$, which results in a value of $d^* = 32m$ for our energy dissipation model. This optimal transmission range is derived for general ad hoc networks and is energy-efficient for general transmissions. However, traffic flows in general ad hoc networks are assumed to be randomly distributed within the network rather than converging to a single base station as in sensor networks. Thus, this fixed optimal solution may not be suitable for sensor networks.

If we take a closer look at the fixed transmission range scheme, we notice that when the first node (the one closest to the base station) dies, there is still some energy left in the remaining nodes, especially those nodes far from the base station. The residual energy increases as the node distance to the base station increases since nodes at far distances have many fewer packets to forward. In the optimal power control scheme, we notice that most of the traffic is still forwarded around the ideal transmission distance d^*. Sensors manage to increase the network lifetime by using up their residual energy transmitting some extra packets directly towards the base station over long distance, lightening the load on the nodes closer to the base station. However, these transmissions are very energy-inefficient, and only a very few extra packets can be transmitted in this manner.

Therefore, a good strategy should strive to achieve energy efficiency and energy balance simultaneously. On one hand, it should allow nodes to transmit their packets using the optimal transmission range as much as possible. On the other hand, a good strategy should simultaneously allow nodes to use up all of their energy. Given the limited freedom on the key parameters, even the optimal data routing cannot achieve both goals for the given deployment strategy. Therefore, in the next section, we will evaluate other strategies and investigate how well they meet both goals.

6.4 Comparison of Different Deployment Strategies

In the previous section, we saw that transmission power control cannot provide much lifetime extension compared with a fixed transmission power approach. However, for deployment strategy DS_1, this is the only option for extending network lifetime. In this

section, we investigate how well each of the other strategies listed in Table 6.1 improve network lifetime. Correspondingly, we will evaluate these strategies using the general terms normalized lifetime and deployment cost, defined in Section 6.2.

In this section, we focus on the two-dimensional grid network as shown in Fig. 6.3. First, we use an arbitrary sample scenario to determine the normalized lifetime for each strategy. In the sample scenario, $N = 180$ nodes are deployed in an area $A = \pi 250^2\ m^2$. We simply assume that this node density is the minimum sensor coverage density. Therefore, $\lambda_a = \frac{N}{A} = \frac{180}{\pi 250^2}$. We also assume a total of 180 Joules are initially assigned to the sensors, which results in 1 Joule per sensor for even energy distribution. Therefore, the energy density $\lambda_e = \frac{E_T}{A} = \frac{180}{\pi 250^2}\ J/m^2$. Finally, the average bit rate among active sensors is $\bar{r_g} = 1$ bits/s.[4]

Once $\widetilde{L}$ has been determined for the sample scenario, we use this value with a target scenario (fixed $\mathbf{P}$ and lifetime L) as a case study to compare the different deployment strategies. For the target scenario, $\mathbf{P} = [A = \pi 250^2\ m^2, \bar{r_g} = 10\ bits/s, \lambda_a = \frac{400}{\pi 250^2}]$, $E = 100$ J/sensor, and the target network lifetime L is one year.

6.4.1 Deployment strategy DS_1: power control

The strategy of power control has been fully studied in Section 6.3. The lifetime for the sample scenario is 4.05×10^5 seconds using the heuristic power control scheme, denoted by the star at the network radius of 250 m in Fig. 6.5. Using Equation 6.8, the normalized lifetime is,

$$\widetilde{L} = L(\frac{\bar{r_g}\lambda_a}{\lambda_e}) = 4.05 \times 10^5 bit/J \tag{6.12}$$

Using this normalized lifetime, the required number of sensors for the target scenario $\mathbf{P}$ and lifetime L can be determined using Equation 6.9. Note that parameters used here are from the target scenario **P** rather than the sample scenario.

$$N(DS_1) = \frac{L\bar{r_g}\lambda_a A}{\widetilde{L}E} = \frac{365 \times 24 \times 3600 \times 10 \times 400/A \times A}{4.05 \times 10^5 \times 100} = 3,116 \tag{6.13}$$

Thus, for the network to operate for one year while meeting the sensing goals requires 3,116 sensor nodes using deployment strategy DS_1. Note that a sensor equipped

[4]Note that because we assume the minimum sensor coverage density in the sample scenario, the average data rate r_g of each sensor is equal to the average bit rate $\bar{r_g}$ among active sensors.

with power control hardware is more expensive than a sensor with a fixed transmission power. A case study of cost evaluation will be presented after we evaluate the other deployment strategies.

6.4.2 Deployment strategy DS_2: mobile data sink

As we mentioned in Section 6.3, variable transmission power control alone does not effectively extend network lifetime. For further lifetime improvement, more constraints must be loosened and alternative deployment strategies must be considered. Referring to the deployment strategies from Table 6.1, if we have freedom on data sink deployment, two alternative strategies can be applied: mobile data sink and multiple data sinks. Let us look at the first case, a mobile data sink (i.e., DS_2).

There are two possible scenarios in which a mobile data sink may be used. The first is a network in which multiple aggregator-capable nodes are deployed, each of which collects all of the data in the network at a given time[5]. The second scenario is a network that employs a mobile data sink (e.g., a robot). These two scenarios are similar from the network routing perspective since all data is sent to a single data sink during a given period.

Suppose that the mobile data sink stops at a given number n_l of data sink locations, and all of the active sensors report to this sink when it stops at a new location[6]. One interesting question to resolve before starting the simulations is how to choose these n_l data sink locations. For small values of n_l such as 2, 3, and 4, we assume the sink locations to be a symmetric pattern as shown in Fig. 6.6. We adjust the size of each pattern gradually and run the simulation. The optimal sink locations with the best lifetime performance can thus be determined through brute force searching. For n_l larger than four, it is difficult to determine the location pattern even if we presume that it is symmetric. Therefore, we resort to random location deployment in our simulations

[5]This scenario may occur if an aggregator node needs a complete picture of the network in order to make any decisions. These aggregator nodes could conceivably collect all data in their region and forward these data to another aggregator node for analysis; however, this would be extremely costly for the aggregator nodes and we cannot assume that they are *completely* unconstrained by energy. Furthermore, unless these aggregator nodes can communicate directly with each other on another channel, this data will need to be forwarded between aggregator nodes by the ordinary microsensors in the network.

[6]We ignore the travel time for a data sink to change its location.

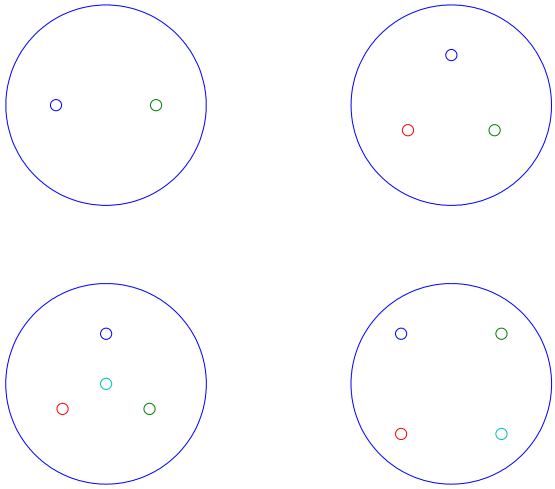

Figure 6.6: Sink locations can be presumed to be in a symmetric pattern. The pattern for less than 5 sinks can thus be determined as shown.

for these cases.

Our data routing model needs to be adjusted for this strategy as follows. First, during the period that the data sink is at each of the locations, the data flow at each sensor should still be balanced. Second, the overall energy consumption of each sensor during the entire network lifetime should still be limited by the initial energy. Therefore, the only necessary modification is on the first constraint: for each data sink ds_k at location k, $\forall k \in \{1, \cdots, n_l\}$

$$\sum_{j=1}^{N} t(j,i) + r_g(i)\frac{L}{n_l} = \sum_{j=1}^{N} t(i,j) + t(i,ds_k) \qquad \forall i \in \{1, \cdots, N\} \qquad (6.14)$$

Using the sample scenario, we can calculate the normalized lifetime $\widetilde{L}$ using our data routing model. Figure 6.7 shows a plot of the normalized lifetime $\widetilde{L}(n_l)$ as a function of the number of data sink locations n_l. Plots of $\widetilde{L}(n_l)$ using optimal data sink locations are given by the dashed lines, and plots of $\widetilde{L}(n_l)$ using randomly chosen data sink locations are given by the solid lines with standard deviation bars in these figures. The performance of sink rotation with no power control is also shown[7].

[7]The optimal lifetime for scenarios with more than four data sink locations is hard to determine

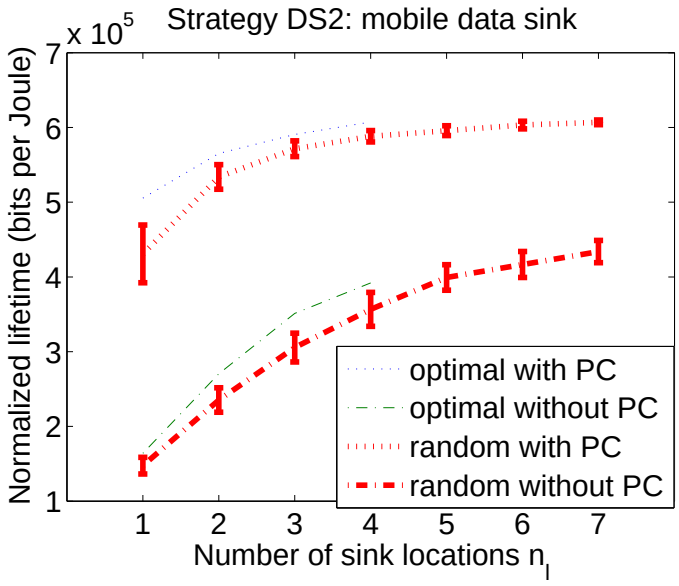

Figure 6.7: Normalized lifetime vs. number of data sinks deployed for the sample scenario. Increasing the number of sink locations improves lifetime until a certain threshold is met and the hot spot problem has been effectively solved.

With the random data sink deployment and power control, the network lifetime shows an improvement of 31% when using three data sink locations instead of just one, and this improvement increases to 38% for six locations. However, using more than six data sink locations does not provide significant lifetime improvement since the hot spot problem is already solved effectively at this point. Simulation results also show that larger gains in network lifetime can be obtained as the network size grows. This is because the hot spot problem becomes worse as the network becomes larger. This trend is essentially the same as the trend discovered in Section 6.3. Also, the gap between choosing optimal locations and random locations tends to diminish as the number of data sink locations increases. When power control is not available, the improvement from more data sink locations is larger. However, the improvement still flattens out at about 7 data sink locations.

since the optimal pattern of the data sink locations is not obvious. Nevertheless, we can assume that the optimal pattern would be able to achieve a lifetime comparable to the upper bound seen in the simulations utilizing randomly chosen locations.

For the target scenario with no power control, if we choose the mobile sink to move 7 times randomly, the normalized network lifetime is $\widetilde{L} = 4.3387 * 10^5$ bit/J, and the number of required sensors is

$$N(DS_2) = \frac{L\bar{r_g}\lambda_a A}{\widetilde{L}E} = \frac{365 \times 24 \times 3600 \times 10 \times 400/A \times A}{4.3387 \times 10^5 \times 100} = 2,907 \qquad (6.15)$$

Although the required number of sensors and the corresponding cost from sensor deployment is decreased compared to DS_1, we should note that a mobile data sink is much more expensive than a stationary data sink used in the power control scheme. This cost may affect the overall desirability when we compare and evaluate the different deployment strategies.

6.4.3 Deployment strategy DS_3: multiple data sinks/clustering

In a clustering approach, multiple aggregator-capable nodes are deployed and each sink collects data from only part of the sensor network for the entire network lifetime. Such clustering schemes have been proposed for wireless sensor networks in [25, 39, 68]. Previous work in this area deals primarily with homogeneous networks, in which any of the deployed nodes is capable of acting as cluster head. In this section, we consider heterogeneous networks, where cluster heads are actually data sinks that are more capable (e.g., those with larger batteries, more processing power and memory, and possibly a second radio to link back to a central base station) and significantly more expensive than ordinary microsensors. In our model, a sensor may send its traffic to whichever cluster head it chooses[8].

In our model, the modification is still on the first constraint. The first constraint should specify that the overall data flow for each sensor is balanced for each data sink's operation. Notice that this constraint is looser than the constraint for DS_2, which utilizes multiple data sinks and thus the data flow to each of the data sinks is balanced. Equation 6.1 should be modified as:

$$\sum_{j=1}^{N} t(j,i) + r_g(i)L = \sum_{j=1}^{N} t(i,j) + \sum_{k=1}^{n_l} t(i, ds_k) \qquad \forall i \in \{1, \cdots, N\} \qquad (6.16)$$

Using the sample scenario, we find the relationship between the normalized lifetime and the number of data sinks that are deployed, as shown in Fig. 6.8. The normalized

[8]The chosen cluster head is typically, but not necessarily, the closest cluster head.

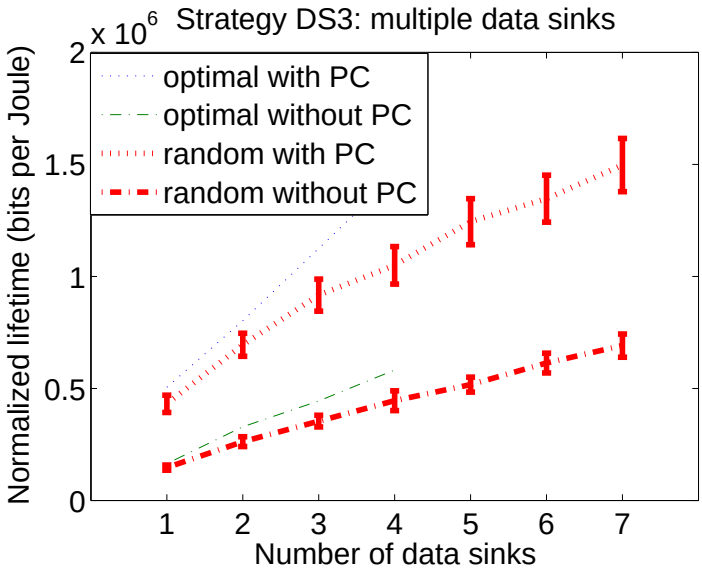

Figure 6.8: Normalized lifetime vs. number of cluster heads deployed. Large gains in network lifetime can be achieved when even a few extra cluster heads are deployed, especially when their locations are optimized. Random sink locations can provide lifetime improvement, but it is not as large as that obtained using the optimal sink locations. When power control is unavailable, the gap between random sink locations and optimal sink locations is greatly reduced.

lifetime is given for optimal cluster head placement[9] as well as random placement, and with power control as well as without power control. As expected, when more cluster heads are deployed, the hot spot problem is reduced and the network lifetime is improved. The most extreme case is when there are so many data sinks deployed that every sensor can find a data sink just one hop away. The hot spot problem is completely resolved in that case. Also note that the performance of clustering is better than that of data sink rotation due to the looser constraint on data flows, which translates into traffic being forwarded over much shorter distances to the closest data sink rather than the single global data sink.

Increasing the number of randomly deployed data sinks from 1 to 7 increases the

[9] Again, optimal cluster head location patterns are only achievable for a small number of cluster heads through brute force searching.

normalized lifetime from 2.6239×10^5 to 6.9122×10^5 when no power control is applied, and it reduces the number of normal sensors from 4,808 to 1,825 for the target scenario. However, more data sinks also increase the extra cost from data sinks. Unlike the nearly fixed extra cost for a mobile data sink, the extra cost of this strategy is more likely to be linearly associated with the number of deployed data sinks. Therefore, a proper number of data sinks must be chosen according to the cost ratio of data sinks and normal sensors. Readers are referred to [50] for more details on how to choose a proper number of data sinks based on the relative costs of network components.

6.4.4 Deployment strategy DS_4: non-uniform energy assignment

Lifetime improvement from power control alone is limited because of the energy inefficiency of the sensors farthest from the data sink sending data directly to the base station in order to evenly distribute the energy load among the nodes. According to [50], in order to achieve near-optimal network lifetimes, it is only necessary to use a fraction of the total energy available in the network.In strategy DS_4, we release the initial energy constraint and allow each sensor to start with a different amount of energy. Thus, we have another strategy that we name non-uniform energy assignment.

In our model, equations 6.1, 6.3, 6.4 and 6.6 are used as the constraints. The lifetime performance for non-uniform energy assignment is shown in Fig. 6.9. Compared to Fig. 6.5 where using optimal power control is the only option, lifetime is greatly improved from 5.05×10^5 seconds (denoted by the circle at the network radius of 250 m in Fig. 6.5) to 9.09×10^5 seconds for the sample scenario when both power control and non-uniform energy assignment are applied.

If non-uniform energy assignment is applied with a fixed transmission range, the lifetime performance is still close to that with power control. This implies that non-uniform energy assignment is more efficient in improving network lifetime compared to power control. With non-uniform energy assignment and no power control, the optimal network lifetime is found when the transmission range is fixed at 34 m, which is very close to the general optimal transmission range $d^* = 32\ m$. Because sensors can be assigned the proper amount of energy for packet forwarding, they should transmit all of their packets using the most efficient transmission range without worrying about energy imbalance. Thus, energy balance and energy efficiency are both achieved using

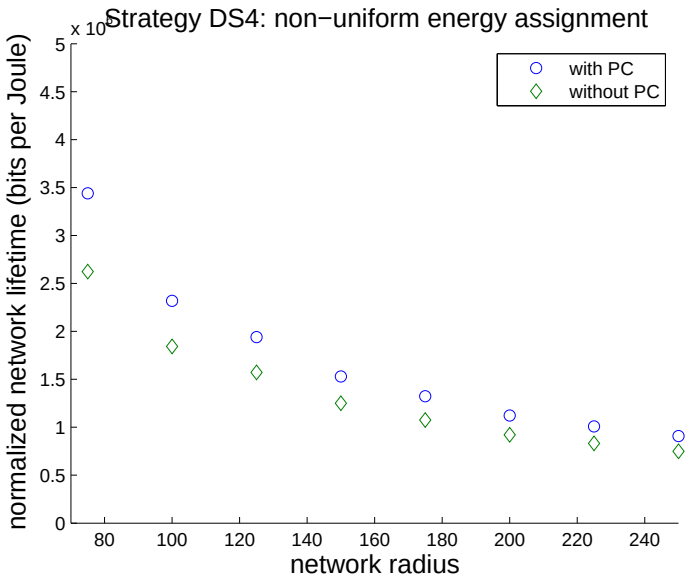

Figure 6.9: Normalized lifetime for different network radius. Power control is no longer important once energy can be freely assigned.

non-uniform energy assignment.

Therefore, intelligent energy assignment seems to be the best choice for sensor network deployment considering that it does not require much extra hardware to implement. However, this strategy is inherently difficult since energy has to be assigned differently for individual nodes at different locations. When the deployment is in a random manner, this becomes almost impossible. A more reasonable energy-based deployment plan is to divide the sensing area into rings according to their distance to the base station. Sensors can be assigned different levels of energy, and be deployed in their respective rings accordingly. Since sensors are not differentiated when deployed in the same energy ring, the complexity of such a deployment plan is reduced. During deployment, nodes closer to the base station should be assigned more energy since they have to forward traffic for farther nodes. Fig. 6.10 shows the optimal energy assignment map for nodes at different distances to the base station. The dotted line is an interpolated polynomial function using the energy map obtained from our model.

For this non-uniform energy assignment deployment strategy, DS_4, the normalized lifetime for the sample scenario is 9.09×10^5 bit/J with power control and 7.49×10^5

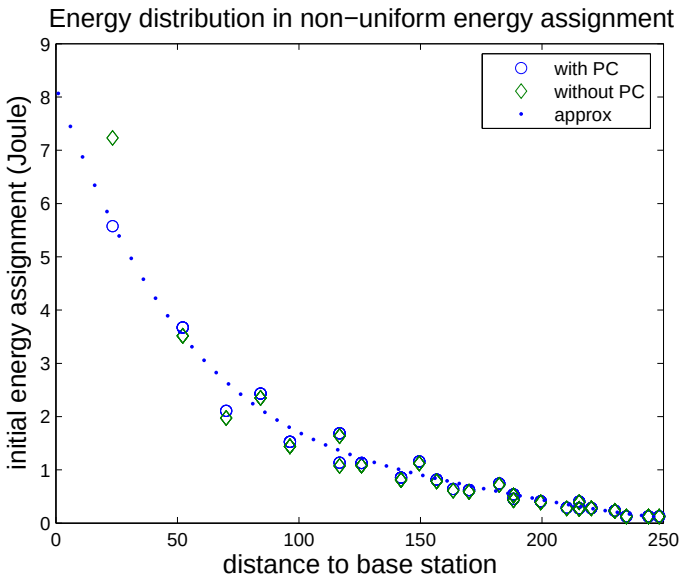

Figure 6.10: Energy distribution map for the sample scenario. Nodes far from the base station should be assigned more energy. The assignment can be approximated using a polynomial function shown by the dotted line.

bit/J without power control. The number of required sensors for the target scenario is the minimum number of sensors, (i.e., 400) if each sensor is capable of carrying a very large amount of energy.

6.4.5 Deployment strategy DS_5: non-uniform relay/sensors

If we assume sensors have the same energy capacities as those in previous uniform energy assignment schemes, we may deploy more relay/sensor nodes according to the energy map in Fig. 6.10, achieving the same goal of providing more energy at a particular point in the network. In this case, in order to meet the target lifetime, we calculate the number of required sensor using Equation 6.9: 1,388 for non-uniform relay/sensor assignment with power control and 1,684 for non-uniform relay/sensor assignment without power control.

Several benefits can be obtained from this non-uniform relay/sensor strategy. First, it avoids complex energy assignment and sensor differentiation. Second, it separates the functions of sensing and wireless communication. The duty of sensors and relay

nodes are clearer, and their design can be better separated. Third, relay nodes do not contain sensing components, and thus may be cheaper than deploying sensors. The deployment procedure of relay nodes can thus be executed after sensor deployment. Since this strategy is essentially the same as the non-uniform energy assignment when formatted into our model, we omit further discussion to avoid repetition.

6.4.6 Deployment strategy DS_6: non-uniform traffic generation

In certain sensor networks, more traffic is generated at far distances from the base station. For example, in sensor networks designed for intruder detection, the network periphery may provide the most important data, as this tells the application when an intruder has entered the network area. The majority of the work for nodes closest to the base station is to forward the traffic, rather than to generate it. The hot spot problem is alleviated automatically by this type of traffic generation pattern. Consider an extreme case in the one-dimensional scenario in Fig. 6.1. If only sensor s_N, the farthest node, generates traffic, the traffic will be forwarded hop by hop to the data sink. Next hop forwarding will be the most energy-efficient forwarding method, and there will be no energy imbalance at all.

Data aggregation can be seen as a variation of non-uniform traffic generation as well. As data are forwarded to the base station, forwarders may perform some processing and aggregate their data with the received data before forwarding. Even if the data generation rate is uniform within the network, data aggregation actually transforms it into a non-uniform traffic generation pattern where more traffic is generated from the outer areas. Again, this helps to reduce the hot spot problem.

On the contrary, in sensor networks where areas closer to the data sink are more of interest for monitoring, more traffic is generated around the data sink. This actually aggravates the hot spot problem. All the strategies mentioned earlier can be applied to alleviate the problem, and our model is still applicable to these scenarios. However, these scenarios are oriented at different types of applications, and thus they are essentially different from the previous scenarios. Therefore, we will not compare the performance of this strategy with that of the previous strategies.

6.4.7 Cost comparison of all deployment strategies

Now that we have studied the normalized lifetimes for the sample scenario and found the number of sensors required for the target scenario for each deployment strategy, we compare each of these strategies in terms of monetary cost. We assume the device costs listed in Table 6.3. The cost of sensors for different strategies is shown in the second row of Table 6.3. The cost of a normal sensor, taking the motes as an example, is $10 per unit [17]. The cost of a sensor with power control can be assumed to be $15 per unit due to the cost of the extra hardware required, and the cost of a sensor with non-uniform relay placement is $15 per unit due to the cost of individual placement. In the third row, we show a case where the base station is relatively cheap, such as when the base station is a simple Stargate node [17], while in the fourth row, we show a case where the base station becomes much more expensive, such as a high power laptop or custom-designed base station.

Table 6.3: Two cost case studies: sensor cost and extra cost.

	DS_1	DS_2	DS_3-2	DS_3-7	DS_5
c_s	15	10	10	10	15
c_{e1}	1000	2000	1000× 2	1000× 7	1000
c_{e2}	10000	20000	10000× 2	10000× 7	10000

The number of sensors required to meet the target lifetime for different deployment strategies, as discussed in the previous sub-sections, is summarized in Table 6.4. Note that the power control strategy can be easily combined with other strategies to further reduce the required number of sensors for meeting the target scenario lifetime. In this table, we list the number of sensors required for: power control (DS_1), mobile data sink with 7 movements (DS_2), the combination of mobile data sink and power control ($DS_2 + DS_1$), 2 data sinks without power control ($DS3$-2), 2 data sinks with power control (DS_3-2+DS_1), 7 data sinks without power control (DS_3-7), 7 data sinks with power control (DS_3-7+DS_1), non-uniform energy assignment (DS_4), and non-uniform relay placement without power control (DS_5).

[10]The number of required sensors is 400, while the number of required wireless nodes, including both

Table 6.4: Number of sensors N required to meet the target scenario lifetime goal for different deployment strategies.

Strategy	N(DS_i)
DS_1: Power control	3,116
DS_2: Mobile sink	2,907
$DS_2 + DS_1$: Mobile sink plus power control	2,079
DS_3-2: 2 sinks	4,808
DS_3-2+DS_1: 2 sinks plus power control	1,816
DS_3-7: 7 sinks	1,825
DS_3-7+DS_1: 7 sinks plus power control	843
DS_4: Non-uniform energy assignment	400
DS_5: Non-uniform relay placement	400 (1,684)[10]

The overall deployment cost is the summation of the cost of the N sensors and the extra cost, as shown in Equation 6.10. To make a fair cost comparison, we only compare individual strategies without any combination with power control. Fig. 6.11 compares the normalized network lifetime and deployment cost of each strategy using the target scenario. We choose strategies DS_1, DS_2, DS_3-2, DS_3-7, and DS_5[11]. Fig. 6.11 shows the normalized network lifetime of these strategies in the upper plot, and it shows the cost of these strategies given the cost values from Table 6.3 in the lower plots.

Although for some strategies the normalized lifetime is higher than that of some other strategies, when taking into account the extra cost of these strategies, they become less desirable than some of the less energy-efficient strategies. A complete evaluation of different strategies should be performed from both an energy and a cost perspective.

sensors and relay nodes, is 1,684.

[11] DS_4 is omitted since it is difficult to make a fair comparison using the minimum number of nodes, and DS_5 can represent DS_4 very well since it is essentially the same strategy from the energy assignment perspective. Also, we do not differentiate sensors and relay nodes to simplify the cost analysis.

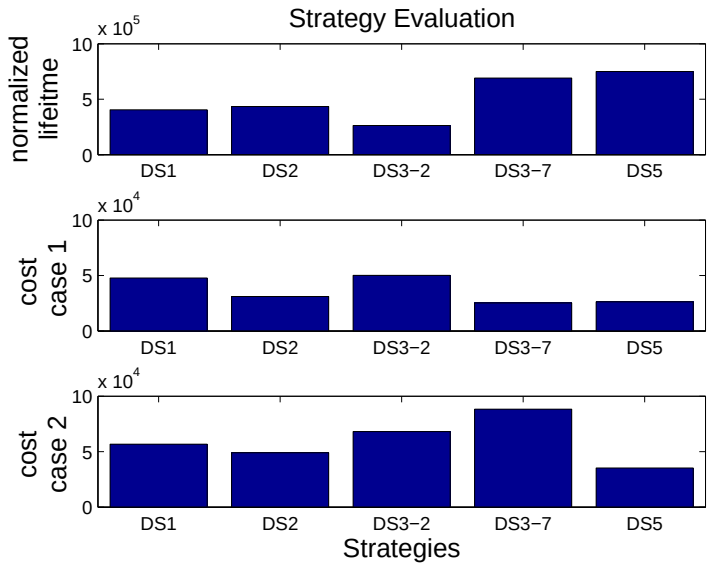

Figure 6.11: Evaluation of the following sensor network deployment strategies: power control (DS_1), mobile base station (DS_2), multiple base stations (DS_3-2, DS_3-7) and non-uniform relay assignment DS_5. The first row is the normalized lifetime. The second row is the deployment cost for the target scenario for cost case 1, and the third row is the deployment cost for the target scenario for cost case 2.

6.5 Conclusions

In this chapter, we proposed a general data routing model to evaluate multiple senor network deployment strategies. From the lifetime and cost analysis on various strategies, we conclude the following.

1. Most sensor network deployment strategies can be generalized using a linear programming model. Their differences lie in the freedom of deployment parameters and the constraints on the network parameters.

2. A good sensor network deployment strategy is one that achieves both energy balance and energy efficiency.

3. Energy imbalance becomes worse when the network size increases, or when the network goes from one to two dimensions. The maximum achievable lifetime decreases correspondingly.

4. The strategy of power control alone is not sufficient to resolve both energy imbalance and energy inefficiency. The lifetime performance improvement from power control is mainly due to energy savings from nodes close to the data sink.

5. A good strategy should operate mostly at the general optimal transmission range (32 m in this chapter).

6. The strategy of deploying a mobile data sink has some limitations on lifetime improvement, while the strategy of deploying multiple data sinks can keep improving the network lifetime until the sub-networks become one-hop networks. This is because the latter strategy has a much looser constraint than the former one.

7. The strategy of non-uniform energy assignment dominates all the other strategies by achieving both energy efficiency and energy balance simultaneously. However, it is inherently difficult to apply in practice.

8. Although more intelligent strategies may have better lifetime performances, the cost of these strategies must be fully considered because once the quality of service of a network is satisfied, cost becomes the primary concern for a practical sensor deployment plan.

Thus, this chapter has the following contributions. First, we propose a general data routing model that can be applied to many sensor network deployment strategies with little or no modifications. Second, we reveal the general lifetime trends with network factors for different strategies. Finally, we propose a general strategy evaluation method to cross-compare the normalized lifetime and cost for different strategies, which provides practical suggestions for real sensor deployment.

Chapter 7

Design of a P2P Based Smart Document System

In this chapter, we study a concrete information retrieval and data propagation example: a smart document system composed of printing devices equipped with network interfaces that can communicate with other devices in the network through a peer-to-peer architecture. The idea of this system is for a user to treat the collection of peer-aware devices as a system, whose goal is to automatically service print requests using the best device or set of devices to meet the user's requirements, such as print quality, latency in finishing a job request, or proximity to the user, at a minimum cost.

We propose using a peer-to-peer networking architecture in which devices with similar features form clusters, and these clusters are maintained in an autonomous manner. We also propose an efficient resource management module that takes the user service request features and constraints and determines the feasibility of the request as well as the optimal allocation of resources (e.g., printers) to service the request at a minimum cost. In addition, the resource management module also provides an interface for realizing the vendor/service provider's systematic goal, such as increasing revenue or attracting more customers. Our proposed smart document system is intelligent and organizes itself in a transparent fashion such that all of the above functions are implemented with little effort from the users.

We implemented a demo system using JXTA [32], a peer-to-peer platform specification developed by Sun Microsystems. The fundamental functions and services of

the “smart document system,” such as the presence service, the status service, the job service and the dispatch service, are implemented. With the current architecture and sample system, more complex functions can be easily added in the future.

7.1 Overview of a Smart Document System

In today’s business world, the omnipresence of inexpensive networking technologies and access to information has set a greater level of expectation from the customer. Users expect traditional services (such as print and system management) to be reliable, cost-effective, and available anywhere, at any time. Furthermore, they demand traditional services to seamlessly fit into their IT environment without compromising security and/or impacting their existing network policies. Information discovery and retrieval in an ad hoc network can be used to satisfy these demands.

The specific scenario we target is one in which there are a large number of networked document devices distributed in an enterprise such as an office complex or large print shop. We focus on meeting users’ expectations for reliable, high quality printing through a *smart document system* built from a collection of document devices that inter-operate with each other in a loosely coupled fashion. The idea is for the end user to treat a collection of peer-aware devices as a system, whose goals are to:

1. Provide a systematic view of services available to the user rather than focusing on an individual device, such as a printer. While accessing these services, the user is only cognizant of the desired output and not of the capabilities or the state of individual devices.

2. Adapt to meet changing customer needs, preferences, and environments.

3. Minimize the cost for providing services and provide an interface for the vendor/service provider to achieve systematic goals such as obtaining more users or maximizing revenue.

Here, we propose a networking architecture and resource management module for this smart document system. We envision the following scenarios for using this smart document system.

1. The end user describes the *features* and *constraints* of a print job, and the system decides the best use of available resources (e.g., printers) to accomplish the user's goals at a minimum cost. In this scenario the user is not concerned with the capabilities of individual devices; instead the user expects the system to deliver the best possible result at a minimum cost. Thus, satisfying the user's requirements becomes the main goal of the system. For example, a user may specify features of the print job such as the following: a. Print output quality, b. Color/black-and-white, c. Paper size, and/or d. Resolution. Additionally, the user may specify constraints of the job request, such as: a. Latency in finishing a job request, b. Proximity of the printer(s), and/or c. Maximum number of printers that can be used.

2. The system adapts to changing resources and environments. For example, if a device has a fault or a problem, the system decides how to work around this device until the problem is resolved. For example, the user's job may be re-routed to another device with the same capabilities.

3. The system adapts itself to meet the service provider's systematic goals. For example, suppose a new printing company purchases a smart document system to provide printing services to its customers. In the development phase of this new company, the manager wants to attract as many customers as possible, possibly reducing profit, but after the number of customers reaches a certain level, the manager wants to make as much profit as possible. The smart document system should allow these systematic goals to be implemented when making resource allocation and management decisions.

We have designed this smart document system using a JXTA-based architecture for peer-to-peer networking of printing devices. Devices equipped with network interfaces communicate with other devices on the same logical network through a peer-to-peer, clustered architecture. Since networked devices are plugged in to the wall, energy is not a constraint in this environment. The clusters are dynamic in nature and are formed in an ad-hoc fashion. Devices join and leave clusters as they are physically moved around and/or are shut down. Similarly, when a new device or service is added to the system, the system can take advantage of the added resource to improve the system's capa-

bilities. Also, the system can take advantage of devices with more capabilities when forming these clusters. The system is intelligent and organizes itself in a transparent fashion.

Furthermore, we have developed an efficient resource management module that takes the user service request features and constraints and determines the feasibility of the request as well as the optimal allocation of resources (e.g., printers) to service the request to satisfy some goal, such as maximizing revenue or maximizing the number of users of the system. The resource management module has two components: resource admission (RAD) and resource allocation (RAL). RAL determines which combination of available resources can provide the features specified by the service request while meeting the constraints, and it finds the resource set that satisfies the service request using a minimum cost. The cost function and the service features are provided by RAD to RAL, allowing RAL to simply find the best (i.e., least cost) solution for the problem and provide this solution to RAD. To obtain the service provider's input for the resource allocation decisions, RAD provides an interface to obtain the service provider's goal and tunes the cost function accordingly before invoking RAL. The combination of RAL and RAD provides an efficient solution for resource management.

Considering the memory constraints of some potential peers that are actually printers, we implemented the RAD and RAL modules as transportable codes and placed them in a server. A device acquires these modules from the server only when it has been selected as the cluster head. This code transportation method eliminates unnecessary memory usage from normal nodes and enables them to allocate their precious resources mostly to their printing jobs.

7.2 Smart Document System: a P2P Network

7.2.1 Goals

From the scenarios we described in the last section, we see that the smart document system utilizes a collection of peer-aware devices as a system to provide services. Therefore, we expect it to have some features that traditional printing systems do not have. We list these features in detail since they determine the best architecture for our system.

1. Resource aggregation. Instead of using a single printing device for a print request, the smart document system can choose from a variety of devices that can provide the required features specified by the user. The benefit of aggregating these devices is two-fold. First, it reduces the required printing time and reduces the queue time on certain popular devices by distributing the jobs more evenly among the available devices. Second, it reduces cost. Different machines have different operating costs. Although choosing the machines with the lowest cost seems to always be the best solution, there may be so many jobs queued in these machines that the job will not be finished in time. Cost should be reduced without sacrificing the fundamental quality of service, such as delay in completing the job.

2. Reliability. In a smart document system, single devices no longer determine the final output of a job. When one device becomes unavailable, even in the middle of servicing a job request, the problem can be circumvented by using other feasible devices to take over the job. Although the penalty may be an increase in cost and processing time, reliability is improved, which is a very important feature in a business world where the customer's needs are paramount.

3. Ad hoc collaboration. A smart document system should support ad-hoc communication. By ad hoc, we mean that devices may be turned on or off at any time, or they may move around and change their physical locations. Also, devices may communicate through wireless network interfaces and may have unreliable communication links at times. The ad hoc nature of these devices should be fully exploited.

4. Transparency. The smart document system, as a product, should reduce any unnecessary involvement of both the service provider and the customers. Service providers only need to inform the system of their current system goal, such as maximizing revenue or attracting more customers. Customers only care about the final outcome, such as print quality, printing time and printing cost. Both the service provider and the customers do not want nor need to know how the network is formed and how the service is implemented on the devices. Therefore, the smart document system should be designed to be network and device

transparent (independent).

5. Increased autonomy. In order to realize the transparency to the users, devices need to be self-organizable. Devices should form the network on their own, and they need to maintain proper operations in this network. They should be able to choose the most efficient way to process a customer request, and they should be fault tolerant when some of the devices become unavailable. When carrying out the above functions, the smart document system should be autonomous so that little extra effort from the users is required.

6. Exploit heterogeneity. Devices are different, and they affect the smart document system from several aspects. First, they have different printing features and capabilities. This printing heterogeneity will affect the resource allocation decision. Second, devices have different amounts of storage and memory. Some devices are simple printers with limited storage and memory, while some may be servers with much larger storage and memory for more extensive computing. Exploiting these differences to determine a proper role for each device will improve the overall networking and printing performance.

7.2.2 The architecture

When designing a network for a class of systems and applications to perform the same function, such as computing, data sharing and communications, either a centralized architecture or a distributed architecture is used. The centralized architecture is also known as the client/server model, as shown in the left of Fig. 7.1. In this model, a client is defined as a requester of services and a server is defined as the provider of services. For example, FTP clients can download files from an FTP server. The benefit of this centralized architecture is the simplicity from the client end.

However, the client/server model has its drawbacks. First, server failure will cause the entire system to fail. Second, the servers must be well chosen during the system configuration. They should have enough capabilities to handle multiple requests from clients simultaneously. Also, these servers must be fault tolerant when providing the service.

Peer-to-peer (P2P) is an alternative to the traditional client/server model, and is

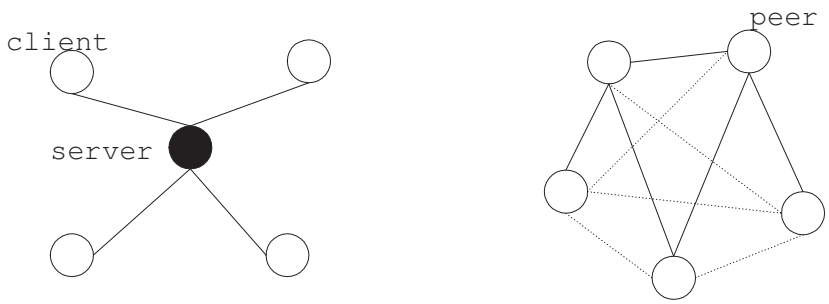

Figure 7.1: A high-level view of the client/server model and the P2P model.

shown in the right of Fig. 7.1. In a P2P network, there are no clear roles of servers and clients. All the participants are simple peers, and they all act as both servers and clients. A good survey on P2P architectures can be found in [44]. In brief, the P2P model has many features that the client/server model does not have. The major advantage of the P2P system is that it can utilize the vast available resources of all devices. It has some other nice features, as listed in [44]: cost sharing/reduction, resource aggregation, improved scalability, increased autonomy, anonymity and enabling ad-hoc communication. P2P applications cover the area of file sharing, distributed computing, collaboration and platforms.

Sometimes, the border between the P2P and the client/server models is obscured by a hybrid architecture. In this hybrid model, a server is first consulted for meta-information about the identity of the peers where the desired information is stored. P2P communications are initiated after this initial contact with the server. Also, some intermediate solutions such as Kazaa [34] use SuperPeers that contain the information of some normal peers in their vicinity. Peers consult SuperPeers for other peers' information first before flooding requests to the Internet. These SuperPeers act as peers to each other, but they act as servers for the normal peers. However, the traffic burden on these SuperPeers is small since only meta-data will be queried from them, rather than entire files as in the client/server model.

This hybrid-P2P architecture is the structure we propose for the smart document system. First, most of the features of the P2P system match the goals of the smart document system, such as resource aggregation, fault tolerance and improved scalability. Second, there are some existing open-source P2P platforms that reduce both the testbed

design period and the cost of the system. However, the implementation of RAD and RAL modules is centralized in nature. Also, to better exploit the heterogeneity of the devices, we design a hierarchal clustered P2P system, as described next.

7.2.3 Forming the clusters

Before going into the details of the clustering design, let us first briefly review the P2P procedure, which can be described as "discover-and-communicate." Take file sharing as an example. Based on the requested file name, a peer first needs to discover those peers that contain the file. Then, communication pipes are set up between the node and the discovered peers. File pieces are downloaded through those pipes. The implementation details simply rest with how the discovery is performed and how these communication pipes are set up, i.e., the information discovery and retrieval problems.

Similarly in the smart document system, when a printing request arrives, the system needs to discover all the devices that provide the features of the request, such as double-sided printing or color printing. One solution for discovering which devices can support the requested features is to query every device each time a request arrives. Precise individual device information can be achieved in this way. The penalty, however, is excessive communications and a potentially long query response delay. Another solution is to let each device update their status periodically to a certain server. Therefore, a request can be satisfied by just checking this server. This solution expedites the request processing time and distributes the communication load to idle times. However, the server has to be powerful enough both in processing and storage capability to handle all these updates. Also, the information may be obsolete if the updates are not frequent enough.

We propose a hybrid clustering scheme as illustrated in Fig. 7.2. Devices with the same features form logical clusters. For example, "cluster color" means that all the cluster members are able to print color copies, and "cluster double" means that every member can produce double-sided prints. Individual devices may join more than one cluster depending on their features, such as devices **A** and **B** in Fig. 7.2.

For the purpose of cluster maintenance, cluster peers are assigned one of the following roles: master, backup and slave, as shown in the left part of Fig. 7.2. The master peer registers all the devices and services within its cluster. It is preferable that these be

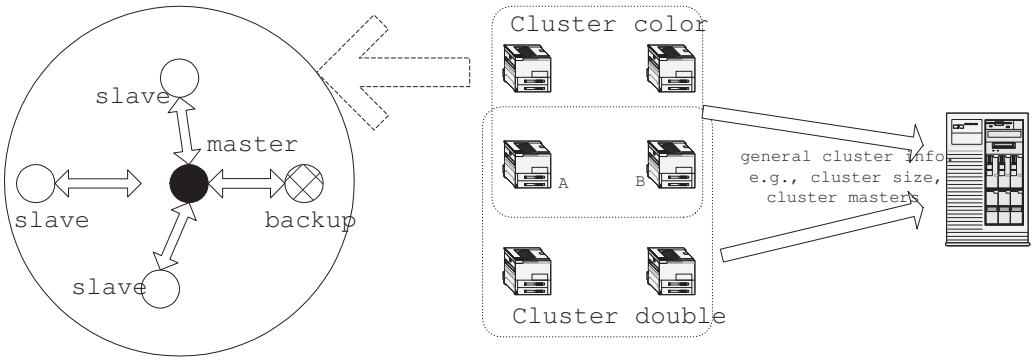

Figure 7.2: A hybrid clustering scheme for the smart document system.

devices with more storage and memory. They act just as the SuperPeers act in Kazaa. Most peers are simply slave peers. They consult the master peer if they want information about the cluster and other cluster peers. They also update their information to the master peer, either periodically or on demand. Backup peers maintain a copy of the same information as the master peers. When the master peer fails, such as leaving the network or accidentally shutting down, one of the backup peers is elected as the new master peer and takes over the cluster.

Clusters only need to register the current cluster status to the servers occasionally. Only general information such as the cluster size, the cluster master's identity and the other cluster members' information such as member names is needed. The status of each cluster member such as toner levels and paper levels will not be stored in the server; these data will be maintained by the cluster master instead.

Using this architecture, when a service request arrives at the server, the set of feasible devices can be determined by contacting the cluster master first and retrieving more details afterwards. The available resources that fit the service request can be quickly determined, which expedites the resource management procedure. Also, once the resource allocation for servicing the request is determined, the execution of the request can be easily transferred to individual devices through their cluster masters.

We have designed and implemented this architecture for a network of printing devices using a JXTA-based system. In this testbed, devices self-form clusters and maintain their roles within each logical cluster. The first role is a SUPERPEER, which is the server shown on the rightmost of Fig. 7.2. This SUPERPEER is a trusty server that

contains all the fundamental RAD and RAL algorithm packages. It periodically broadcasts a beacon so that every peer within the network is aware of its exitance and is able to communicate with it instantly when necessary.

The second important role is a MASTER within a cluster as shown on the leftmost of Fig. 7.2. It maintains the correct functionality of the cluster by broadcasting beacons, and it gathers status information from each peer within the cluster. It also communicates with the SUPERMASTER to obtain the up-to-date RAD and RAL algorithm packages and report the gross cluster information to the SUPERMASTER.

The third role is a SLAVE, which listens to the beacons and status requests from the MASTER node. It reports its current status to the MASTER, and it waits for the dispatched job pieces from the MASTER.

The last role is a BACKUP, which synchronizes with a MASTER periodically and maintains the same information as the MASTER node. It takes the role of a MASTER and becomes a MASTER when it detects a failure of the MASTER. It does not make any decisions using RAD and RAL. However, it processes the job pieces sent from the MASTER, just as SLAVE nodes do.

A node changes its role within a cluster in an autonomous manner. When a node joins a cluster, it is a SLAVE by default. If it does not hear any MASTER beacons, it promotes itself to a MASTER and retrieves the RAD and RAL packages from the SUPERMASTER. It starts to broadcast beacons to allow peers that may join later to become either SLAVES or MASTERS. If a node does not see enough BACKUPs in the cluster from a MASTER beacon, it promotes itself to a BACKUP. A node also deprecates itself from a MASTER to BACKUP or SLAVE depending on the number of other nodes in these roles within the cluster.

In the next section, we explain how we manage the available resources to complete the service request effectively, and how the resource management module assists the service provider in making the best resource allocation decisions using an economic model.

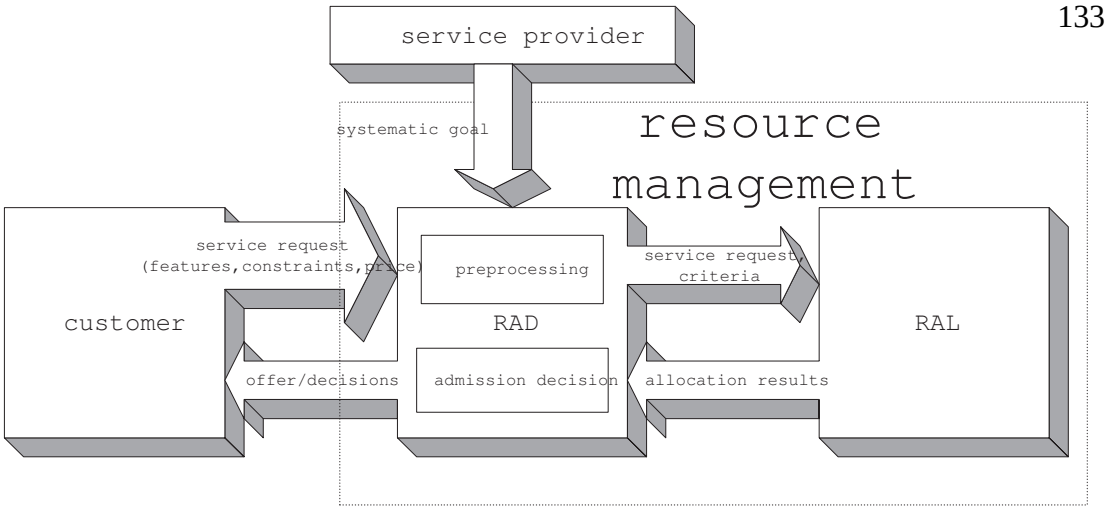

Figure 7.3: The resource management module, its components RAL and RAD and their relation with the service provider and the customer.

7.3 Resource Management

7.3.1 Overview

The goal of the resource management module is to efficiently manage and allocate available resources to meet certain quality of service requirements from the end users and achieve certain systematic goals from the service provider at the same time. The module, shown in Fig. 7.3, has two components: resource admission (RAD) and resource allocation (RAL). It has two interfaces, one for the customer and the other for the service provider.

Resource allocation (RAL) determines which resources are needed to provide certain required services. Given the service requirements from the customer and the current available resources, an efficient resource allocation scheme should be able to provide the optimal solution based on certain criteria while satisfying the basic service requirements. For example, RAL may be asked to determine how to minimize the printing cost for printing 1000 color copies in one day. In this example, the basic service requirement is to print 1000 copies in one day. The resources that can participate in the job must be color-printing devices. The criteria for the solution is to minimize the printing cost. In response to this question, RAL either provides a solution, or, if the constraints are too strict, RAL determines that no solution exists. In this case, it means

that it is impossible to finish the job in one day using the currently available resources (printers).

While RAL determines how resources can be allocated to complete a service request, the resource admission module (RAD) determines whether the requested service should be permitted. This module helps the service provider to achieve higher level goals than simply solving the problem. For a company, there may be different goals at different times. In some cases, even though there are enough resources remaining to satisfy certain service requirements, the request should be rejected because it is not profitable. Or sometimes, the request should be rejected because the services, although profitable if provided, may degrade other services and thus decrease overall customer satisfaction. Also, a service may be accepted in order to explore a larger customer market even if the service acceptance is not profitable temporarily. RAD should be able to incorporate these goals while making resource admission decisions.

There are a couple different scenarios that RAD may face. In one scenario, a customer may request certain services with an offered price. RAD simply replies yes or no. In another scenario, the customer only requests certain services. RAD offers a price for the service, and the customer decides whether or not to take the offer. When we designed RAD, we mainly considered the second scenario. The first scenario can be seen as a special case of the second scenario, where RAD knows if the offered price will be accepted before the customer replies.

RAD also has to consider the current goal from the service provider when offering the price to the customer. While a higher price is more profitable, it is more likely to be rejected by the customer. With a lower price, it is easier to obtain a customer, but this may bring little profit. The strategy of offering a good price depends on the current interest of the service provider. If a higher profit is preferred, the price should be set a little higher. If exploring new customers is important, a lower initial price may be a good start.

The resource management module operation can be described using Fig. 7.3. From the leftmost, the customer requests certain services with certain constraints and service features. For example, the customer requires 1000 color copies of a document to be printed in one day. RAD, when receiving this request, checks the goal preset from the service provider and determines the cost function for RAL to use, say to minimize

the overall resource usage (to balance resource usage among different machines is another criteria for fairness). RAL performs resource allocation to meet the request's features (color printing) according to the request's constraints (job finished in one day) by choosing a proper algorithm according to the criteria (cost function). RAL then reports the allocation results back to RAD. From these results, RAD calculates the cost of such an allocation plan, say, $10 in this case. It chooses a price offer based on the current goal, say $11 if it is willing to gain a profit of only $1 in this case to attract more potential customers.

Next, we will go into the details of RAL and RAD and explain how we implement these two components.

7.3.2 RAL

There are several scenarios for resource allocation, depending on the types of jobs and the types of resources available. Solutions vary for different scenarios. We briefly list potential scenarios and their possible solutions.

There are two elements in completing jobs using machines: the jobs and the machines. In general, to simplify the problem, we assume that one machine is able to process only one type of job at one time. Individual jobs can be independent or they can require certain time sequences. We categorize the problem into two cases.

7.3.2.1 Machine-non-differentiable jobs

In this type of problem, jobs are non-differentiable. They require the machines to have the same features to be able to process them, and the outcome is exactly the same. A simple example is to print 1000 color copies of a document. This job can be done in any machine that supports color printing. The printing results from those machines are all non-differentiable in the aspect of color.

For this type of problem, RAL determines how to allocate the job to multiple machines efficiently while satisfying certain constraints. Suppose the above 1000 color prints must be done within one day, and different machines have different costs for printing one copy. RAL will determine how to allocate these 1000 copies to different machines so that the job can be done within one day while the total cost is minimized.

In a mathematical language, we have a job with N pieces and M machines. Take the cost and processing time as an example, we assume that the ith machine is able to process k job pieces with a cost of $c_i(k)$ in time $t_i(k)$. The question usually becomes: how can we allocate N jobs to these M machines so that we can satisfy certain constraints, e.g., the cost is less than a certain threshold and the processing time is the shortest, or, the processing time is less than a certain threshold and the cost is the minimum. The final solution will indicate how many jobs should be assigned to each machine, n_i for $i \in \{1, \cdots, M\}$.

The above problem can be solved using linear programming if the cost and delay are linear with the quantity of the job pieces, n_i. Considering the fact that printing speed and printing cost are generally proportional to the printing quantities, RAL is able to provide a solution for this non-differentiable type of job.

7.3.2.2 Jobs with machine sequences

In a print shop, a job has to go through several machines in a given sequence pattern. For example, documents have to be printed first, then they need to be bound by another machine. If several jobs arrive at the same time, what should be the right sequence for these jobs to go through?

The problem we are facing here is called the job shop problem. Unlike the previous machine-non-differentiable job problem, the job shop problem is generally hard [14]. For a small number of machines, there may be solutions. For a large number of jobs and machines, researchers have proposed various heuristic solutions that meet different goals.

A complete discussion of the solutions of all these problems is beyond the scope of this chapter. However, based on the above discussion, one thing is clear: RAL must be equipped with the means to solve enough of these problems so that it can provide the best result given different constraints and criteria for various scenarios.

7.3.3 RAD

The resource admission component interacts with several sources: RAL, the customer and the service provider. RAD suggests a price to the customer after receiving a customer service request. This price is decided by integrating the systematic goal from the

service provider and the resource allocation from RAL. Economic models and marketing strategies are also included in RAD.

The procedure of RAD and how it interacts with the other three components are described as follows. First, RAD achieves the systematic goal from the service provider. Example goals include gaining the maximum revenue, attracting more users, maximizing customer satisfaction, or maximizing the usage of resources. These goals can be used concurrently with different priorities. Let us suppose for now that the service provider's goal is to maximize revenue.

Next, RAD receives a printing request from the customer. Let us continue the earlier example of printing 1000 color copies in one day. In order to obtain maximum revenue, RAD tries to reduce the printing cost. Therefore, when RAD invokes RAL, the constraint is to finish the job in one day, while the criteria is to find a good solution to minimize printing cost.

When RAL returns the resource allocation results with the cost, RAD is able to decide an appropriate price for completing the job request. Several factors are taken into account. First is the printing cost. Second is the system goal, i.e., to make a profit in this case. Third, RAD must consider the satisfaction level from the customer. Only when a customer is satisfied with both the price and the service will a price offer be accepted. However, a better service requires more resources. This leads to a larger cost and implies that a higher price must be charged. How to find a balance between the service quality and the price is the main challenge for RAD.

Before RAD makes its final offer, it may invoke RAL several times to explore different possibilities. Eventually, the price offered to the customer may contain more than one option, such as,

1. The 1000 copies can be finished in one day for $200.

2. The 1000 copies can be finished in 1.5 days for $150.

3. The 1000 copies can be finished in 0.5 days for $250.

Finally, RAD offers the selected price(s) to the customer. Different customers have different views on the price. For the same price, some of the customers may think that it is too much, while other customers may think it is reasonable. In other words, for an offered price, there is a certain probability that it is accepted.

Each systematic goal from the service provider can be expressed using the assigned resources, the offered price for the resources, the customer satisfaction degree for using the resources, and the customer acceptance rate for the price and the satisfaction degree. RAD formulates the connections and makes intelligent price offers that maximizes the goal.

7.4 Demonstration System

We implemented the “smart document system” using JXTA, an open-source peer-to-peer platform developed by Sun Microsystems. JXTA is implemented using Java, and it already implements the basic peer-to-peer functions. These functions are implemented in protocols. The protocols that are related with our project include:

1. Peer Resolver Protocol (PRP): sends a query to any number of other peers and receives a response.
2. Peer Discovery Protocol (PDP): advertises contents and discover contents (peers).
3. Peer Information Protocol (PIP): obtains peer status information.
4. Peer Binding Protocol (PBP): creates a communication path between peers.

We implement our functions as services based on these protocols, and these services are:

1. Presence Service: Built on the PDP protocol, the Presence Service allows a node to determine and update its role within its cluster, and it maintains the integrity of a cluster.
2. Status Service: Built on the PIP and PBP protocols, the status service allows a MASTER to periodically query SLAVEs about their printing status such as the amount of remaining paper, the amount of remaining ink and the cost of printing a sheet. The status information will be later used by the RAD and RAL modules for resource allocation and management.
3. Synchronization Service: Built on the PIP and PBP protocols, the synchronization service allows BACKUPs to synchronize with the MASTER periodically.

4. Job Query Service: Built on the PBP protocol, the job query service allows a user to request a job and obtain the price offer from any peer. It calls the RAL module for resource allocation and the RAD module for pricing decisions. Once the proposed price is accepted by the user, a job dispatch service will be initiated.

5. Job Dispatch Service: Built on the PBP protocol, this service allows a MASTER to divide a job into job pieces and dispatch these job pieces to different peers according to the resource allocation results obtained earlier through the RAD module. It also notifies the user about the completion of a job once the MASTER has received the completion notification of job pieces from all the dispatched peers.

We tested our demo system using four machines: one linux platform and three unix platforms. One of the machine is hardcoded as the SUPERMASTER with a machine ID 0 and is on all the time, while the remaining machines 1, 2 and 3 are normal peers. Each machine is equipped with different printing capabilities, such as paper size, printing speed and printing costs.

There are three buttons on the interface, as shown in Fig. 7.4: “who is there”, “check printer status,” and “job request,” which are used to test different services. On the lower part is a display area showing the background debugging information. To be consistent, we have machine 1 join the network and become a MASTER first, and the rest of the nodes join in the network after machine 1. Although we have done a complete test of each service, it is not easy to show them all here. Instead, we only demonstrate the fundamental services.

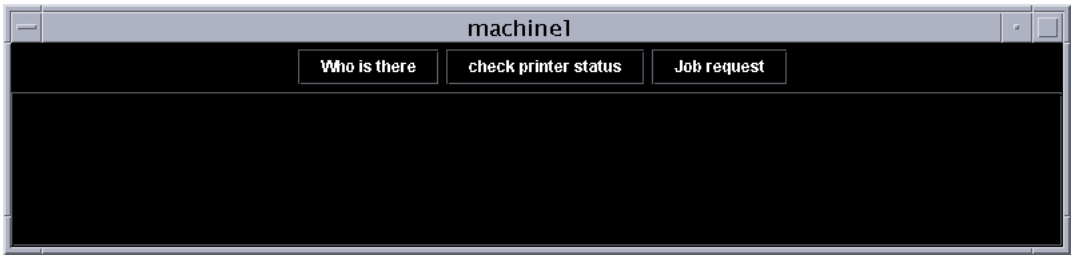

Figure 7.4: The interface of the demo system is composed of three buttons and a display area.

7.4.1 Presence service

By clicking the button “who is there,” we are able to discover which nodes are in the network and what roles they currently play. The knowledge of the presence of other peers depends on the role of the current node. A MASTER has the most knowledge, while a SLAVE has the least.

Fig. 7.5 demonstrates how the presence service discovers the roles of different nodes. Node 1 becomes a MASTER after joining the network as the first node. Node 2 joins as a SLAVE and later changes to a BACKUP. Node 3 joins as a SLAVE and remains a SLAVE.

Figure 7.5: Presence service: nodes 1, 2 and 3 join in the network and determine their roles automatically. These figures show the results of node 1, the MASTER node, executing the presence service at different times.

7.4.2 Status service

By clicking the button “check printer status,” a node displays the printer status of known peers. The knowledge of the status of other peers also depends on the role of the current node. Again, the MASTER has the most knowledge about the other printers,

and SLAVEs only know their own status. There is a periodical polling scheme running in the background, from which the MASTER obtains the updated SLAVEs' printer status.

Fig. 7.5 demonstrates the status service when nodes 1, 2 and 3 join the network one by one. From the dropdown menu, more and more printer status are known by the MASTER. If we choose to display the printer status of node 2, then we have the plot on the second row showing the printer status of node 2.

Figure 7.6: Status service: the information about the printers' status grows at the MASTER as new nodes join the network. This is implemented by the MASTER periodically polling the peers. The peers' printer status is stored in the MASTER for use in resource allocation.

7.4.3 Synchronization service

One part of the synchronization service is initiated when a SLAVE becomes a BACKUP. The BACKUP synchronizes with the MASTER periodically. The synchronization ser-

vice is shown in the display area of Fig. 7.7, when a node just turns from a SLAVE to a BACKUP.

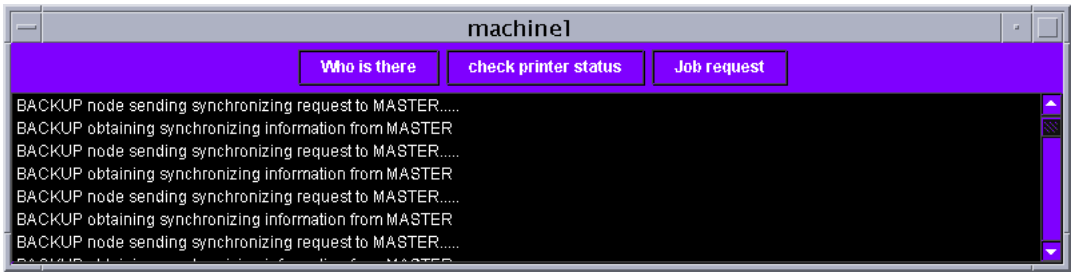

Figure 7.7: Synchronization service: a BACKUP synchronizes with a MASTER periodically.

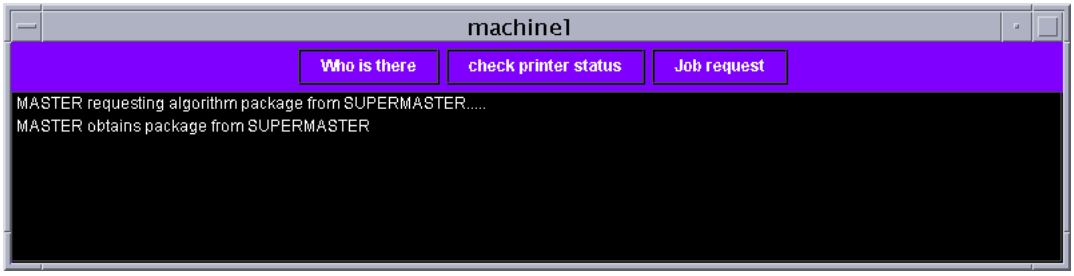

Figure 7.8: Synchronization service: a MASTER synchronizes with a SUPERMASTER.

Another part of synchronization is done when a node becomes a MASTER. The new MASTER updates and retrieves the updated RAL and RAD algorithm package from the SUPERMASTER node, machine 0 in this case (Fig. 7.8).

7.4.4 Job query service

A user can initiate a job query from any node's terminal by clicking the "job request" button. Fig. 7.9 shows the input from a MASTER node. However, a job request can be input from any machine. The user needs to input the printing requirements as well as the copies to be printed and the desired time constraints. The terminal offers a price

to the customer based on the results calculated from the RAL and RAD modules. A user can accept it or reject it. Since only the MASTER node knows the printer status of other nodes, there is a whole set of protocols running in the background to transport the job query, the price offer and the offer acceptance back and forth between requesting peers and the MASTER.

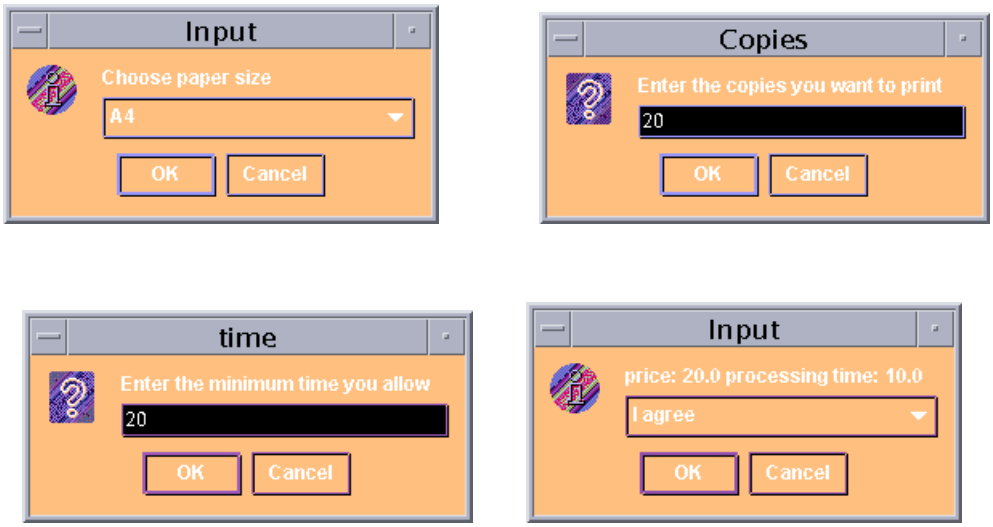

Figure 7.9: Job query service: a user queries for a job service and receives a price quote.

If different time constraints are input by the user, the price quote may change since more expensive but faster machines may be used to meet the user requirement (Fig. 7.10).

7.4.5 Job dispatch service

Once a user agrees to a price quote, the previously stored job allocation scheme is retrieved, and a job is divided into job pieces and sent to the pre-assigned machines (printers). A machine notifies the MASTER once it completes its piece of the job. When all pieces are done, the MASTER sends a job completion notification to the user at the requesting terminal (Fig. 7.11).

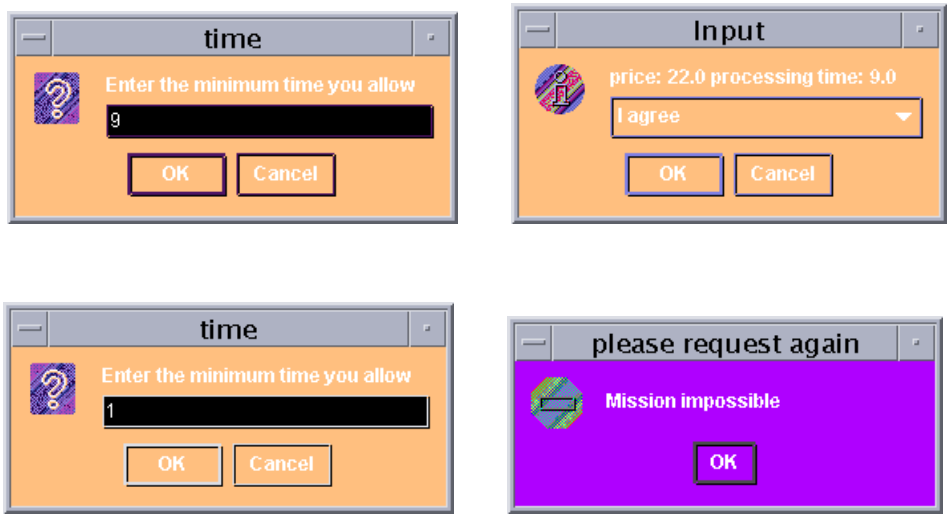

Figure 7.10: Job query service: a user queries for a job service with different time constraints and receives different price quotes.

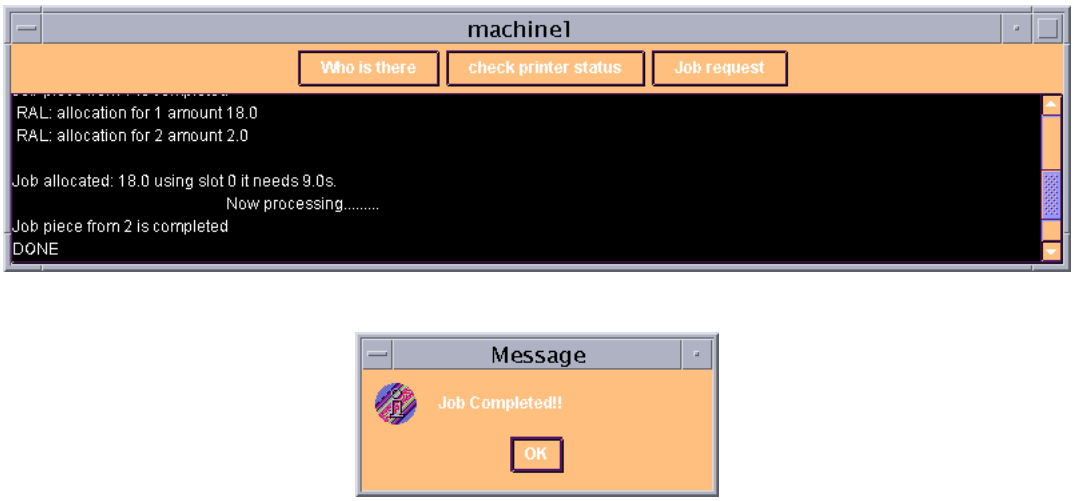

Figure 7.11: Job dispatch service: after a user accepts a price quote, the MASTER dispatches pre-calculated job pieces to pre-assigned machines. The MASTER sends a job completion notification to the user at the requesting terminal once all job pieces are completed.

7.5 Conclusions

In this chapter, we introduced a smart document system, which provides better printing services using mature network technology. We focus on the system architecture design

and the resource management module implementation. Using a P2P network architecture enables the system to effectively find available resources (information discovery), while the resource management module allocates the resources efficiently and provides resource admission functions through setting appropriate pricing for the services (information retrieval). With these functions, the smart document system not only provides better printing service to the end users, it also incorporates economic models to allow service providers to get involved in the decision making without going into the details of resource management.

We demonstrated the effectiveness and portability of our design using a demo system. In our future work, we will incorporate our proposed information discovery and retrieval techniques into the expansion of the system to a full ad hoc networking system. We will show that the efficiency of systems using current networking technologies can be greatly by using our techniques.

Chapter 8

Conclusions

As wireless ad hoc networks and sensor networks become more and more realizable, this dissertation contributes to the MANET community by focusing on resource-efficient peer discovery and data routing for both mobile ad hoc networks and static sensor networks. Furthermore, a real implementation of a service discovery and retrieval system has been demonstrated.

The major difference of this dissertation with previous efforts is that we attempt to solve the information retrieval problem starting from the factors with the largest impact through modelling the problem, and that we strive to make our solution as generic as possible and adjust our solutions to reflect other minor factors through simulations. We have formalized and shown how to optimize the solutions to several problems related to information discovery and retrieval in ad hoc networks, namely

- general peer discovery strategies that minimize searching cost
- a route discovery strategy that can be used for common routing scenarios where route caches, valid or stale, pervasively exist in the network
- a distributed route discovery scheme that discovers long lifetime routes at the very start, which is especially effective for mobile ad hoc networks where link qualities can be quantified by link lifetimes
- a global optimal data routing scheme to maximize the network lifetime of sensor networks, and corresponding alternative strategies beside data routing for further lifetime extension

- a smart document system that incorporates the ideas of information retrieval and resource management in an ad-hoc manner, which serves as a good example for practical ad hoc networking designs.

8.1 General Peer Discovery

Our study on general peer discovery assumes that nodes and targets are uniformly distributed and targets are all identical. This simplified model provides a reasonably generic peer discovery framework to for us to start with, as well as for future researchers to build more complex models on. From our study, we conclude that:

1. When there is only a single target in the network, simple flooding is a fast yet efficient solution.
2. When there is more than one target in the network, multiple searches can reduce search cost.
3. However, more than three searches is unnecessary. It is crucial to choose a good searching radius in the first attempt.
4. When discovering a few from a large number of targets, cost saving is significant. When discovering most of the targets, cost saving is negligible.
5. Our proposed ring-splitting scheme is able to achieve near-optimal performance.
6. The “searching area” in our model can be mapped to a hop value and achieve substantial searching improvement for ad hoc networks.
7. Choosing the right searching strategy highly depends on the amount of available information about the network parameters and the searching task. When there is not enough information, simple flooding is a fast solution.

8.2 Route Discovery with Route Cache

Route discovery with route caches is a peer discovery problem with non-uniform and non-identical multiple targets in the network. It does not fall into the general multi-

target discovery category. After a separate study on this special problem, we conclude that:

1. In order to reduce routing overhead from a local searching technique, a caching technique must be applied. This is essentially the same concept as in the last section, where cost saving only occurs when there are multiple target in the network.

2. Optimal local searching radius depends on the caching conditions in the network, which are affected by node mobility and traffic patterns. When the caching condition is very good, local searching should be restricted to one-hop neighbors. When the caching condition is bad, the local searching radius should be set at about half of the network radius.

3. We proposed a caching condition measurement scheme to enable dynamic local searching radius adjustment. Simulation results show that routing overhead is reduced and routing performance is improved using our adaptive local searching scheme.

8.3 Data Routing in Mobile Ad Hoc Networks

After the conclusion of our study on information discovery through peer discovery, we shift to the secondary procedure: data routing. Data routing is first studied in mobile ad hoc networks. In this type of network, each node has its own individual behavior with regard to node mobility and traffic generation. The interaction among nodes only occurs at the routing layer, where nodes have to utilize other nodes for forwarding their packets to some destinations they cannot reach by themselves. Shortest-path routing is a common routing algorithm used for these mobile ad hoc networks. This makes sense, since the shorter the path is, the fewer times a packet is forwarded from one end to the other, and the higher the efficiency of data forwarding is.

Mobility causes routes to break often in this type of network. Therefore, we suggest discovering long lifetime routes with short route lengths. A long lifetime route not only reduces the frequency of route breaks, it also enables a continuous traffic flow during route breaks by temporarily switching to a backup route during a route break. Through the study of data routing in mobile ad hoc networks, we conclude that:

1. Links often break in mobile networks due to node mobility, and once this happens, all routes containing this link are invalid.

2. The routing layer must test old route caches or discover new routes during a route break. The upper transport layer protocol also has to change its behavior to adapt to the temporary route failure. Unfortunately, TCP, the most widely used transport layer protocol, cannot adapt properly.

3. Long lifetime routes (LLRs) can reduce the frequency of route breaks. The lifetime of an LLR linearly increases with its route length.

4. An LLR with a longer route lifetime reduces the frequency of route breaks, thus reducing the data routing overhead from route discovery. However, its longer route length increases the number of hops, thus increasing the data routing overhead from packet forwarding. To reduce data routing overhead from both route discovery and packet forwarding, an LLR with the shortest path is desired.

5. We propose a global LLR (g-LLR) algorithm to discover LLRs at each route length for any pair of nodes for a given network.

6. We propose a distributed LLR (d-LLR) algorithm to discover two LLRs with short route lengths in one best-effort search. We further propose a fast-switch scheme, which uses the two discovered LLRs to maintain continuous traffic flow for upper layer protocols during route breaks.

7. Simulation results show that the LLRs discovered by d-LLR are near-optimal to the LLRs discovered by g-LLR, and they improve the route lifetime by 40% compared to random shortest-path routing.

8. Simulation results show that d-LLR is able to improve the routing performance from all aspects such as packet delivery ratio, packet delivery delay and energy consumption. D-LLR is also able to improve the TCP performance instantly without any cross-layer design. D-LLR-FS is able to further improve packet delivery delay by maintaining continuous traffic at a negligible cost in terms of packet delivery ratio.

9. The performance of d-LLR highly depends on the accuracy of link lifetime estimation. However, the performance of d-LLR-FS remains relatively steady when link lifetime estimation is erroneous. It achieves this by providing an automatic tradeoff between energy consumption and link lifetime estimation accuracy.

8.4 Data Routing in Sensor Networks

Sensor networks are often deployed to achieve one global application goal. Often times, sensor networks require many-to-one traffic patterns, which cause energy imbalances in the network. Since the premature death of nodes may cause the malfunction of the entire network, data routing must be designed to achieve energy efficiency and energy balance simultaneously. Through the study of data routing in sensor networks, we conclude that:

1. Data routing in sensor networks must be studied from the network perspective to achieve energy efficiency and energy balance simultaneously.

2. The many-to-one traffic pattern is the main source of energy imbalance.

3. A sensor network deployment strategy can be described by several network parameters such as sensor capabilities, base station options, initial energy assignment, sensor locations and traffic generation patterns.

4. We propose a general data routing model, a lifetime analysis method and a cost analysis method to evaluate a given network deployment strategy. Most sensor network deployment strategies can be generalized by our linear programming model. Their differences lie in the freedom of deployment parameters and the constraints on the network parameters.

5. A good sensor network deployment strategy is one that achieves both energy balance and energy efficiency.

6. Energy imbalance becomes worse when the network size increases, or when the network goes from one to two dimensions. The maximum achievable lifetime decreases correspondingly.

7. The strategy of optimal data routing alone is not sufficient to resolve both energy imbalance and energy inefficiency.

8. A good data routing strategy should operate mostly at the general optimal transmission range.

9. The strategy of deploying a mobile data sink has some limitations on lifetime improvement, while the strategy of deploying multiple data sinks can keep improving the network lifetime until the sub-networks become one-hop networks. This is because the latter strategy has a much looser constraint than the former one.

10. The strategy of non-uniform energy assignment dominates all the other strategies by achieving both energy efficiency and energy balance simultaneously. However, it is inherently difficult to apply in practice.

11. Although more intelligent strategies may have better lifetime performances, the cost of these strategies must be fully considered because once the quality of service of a network is satisfied, cost becomes the primary concern for a practical sensor deployment plan.

8.5 Completion of an Information Discovery and Retrieval System

Finally, in order to learn about information discovery and retrieval at the application level, we designed and implemented a peer-to-peer based "smart document system." Although the implementation of this project varies significantly with our previous research, the core idea of peer discovery and information retrieval remains the same.

1. The peer-to-peer structure is a convenient choice for general information discovery and retrieval system designs due to its features such as decentralization, scalability, ad hoc connectivity and fault resilience.

2. A hybrid peer-to-peer infrastructure simplifies the implementation of decision making and resource management.

3. JXTA is an open-source peer-to-peer platform that has been equipped with basic functions. New services can be built over these functions through Java.

4. The resource allocation and resource management modules provide interfaces to both end users and potential service providers, thus extending the number of potential buyers.

5. The RAD and RAL modules should be implemented in portable codes for easy updating and removing.

The work in this dissertation has demonstrated how to improve the efficiency of information discovery and retrieval at different network stacks and protocol architectures. Again, we believe that intelligent models of complex networks are needed to provide us a better understanding of the world and help us better design much more efficient information discovery and retrieval schemes.

8.6 Future Work

In this dissertation, we have tried to make our models as general as possible. As a result, we have made some simplifying assumptions, such as those regarding uniform node location distributions and uniform traffic distributions in our models. Although we still take minor factors into account in our simulations and model calibrations, we intentionally omit them in our model development so that we can focus on the essential factors when designing solutions to information discovery and retrieval problems. Therefore, there is still much to be done to extend and enrich our work.

For the general target discovery problem, we assumed a homogenous network without node differentiation. In another scenario, we can consider a heterogeneous network where ad hoc nodes can choose to communicate with other nodes using either the local base station or multi-hop forwarding. In this case, our target discovery problem can be generalized so as to discover how many searching attempts are required, what are the searching areas for local flooding, and what are the searching areas when using base stations. This problem is more difficult than the one studied in this dissertation in that base stations may serve as routers for one node to reach another. However, using base

stations may not be the most efficient approach since multi-hop forwarding has to be used for the source node to reach its local base station, and vice versa for the destination node. Therefore, we may start our thinking by simply dividing a network into several clusters according to the number of base stations and overlapping these clusters into one multi-tier cluster, although the problem may not be fully solved from there.

For the route caching problem, we restricted our study to limiting the overhead of route discovery. However, from the simulation results shown in Tables 5.3 and 5.4, a bad route returned from a route cache may lead to a sequence of unsynchronizations between the routing layer and the upper transportation layer, thus severely deteriorating the overall performance. Also, considering when there is moderate traffic, the overhead from data routing will dominate the overhead from peer discovery. Therefore, route caching may not be worth using at all in some cases. One possible scenario that route caching is worth using is when the route cache is very likely to be valid, and the traffic in the network is very low. In general, we should be more conservative in applying route caches than we have proposed in chapter 4. How to adjust our solutions to reflect the impact from upper layers and obtain an overall optimal performance rather than simply reducing routing discovery overhead at the routing layer is an interesting question.

Our LLR protocol performance relies heavily on the accuracy of the underlying link lifetime estimation scheme, despite the fact that we have demonstrated in Table 5.5 that LLR-FS is able to achieve good performance even when the error of link estimation is large. Therefore, our next step for LLR is to choose a good link lifetime estimation scheme from the existing solutions and experiment to discover what schemes are suitable for LLR, thus providing a complete solution for the routing layer protocol.

In our study on sensor network deployment evaluations, we only briefly discussed the effect of data aggregation as a factor of traffic generation pattern. This much simplified discussion does not justify the importance of data aggregation. In fact, data aggregation alone deserves a separate study. Integrating data aggregation and incorporating different data aggregation schemes into our deployment evaluation model are underway.

We have demonstrated the effectiveness and portability of our design using a demo system. Future work is needed to incorporate our proposed information discovery and retrieval techniques into the expansion of the system to a full ad hoc networking system

and thoroughly test our solutions on this expanded system.

Bibliography

[1] C. Avin and G. Ercal. Bounds on the mixing time and partial cover of ad hoc and sensor networks. In *Proceedings of Second European Workshop on wireless sensor networks (EWSN)*, 2005.

[2] J. Boleng, W. Navidi, and T. Camp. Metrics to enable adaptive protocols for mobile ad hoc networks. In *Proceedings of the International Conference on Wireless Networking (ICWN 2002)*, 2002.

[3] D. Braginsky and D. Estrin. Rumor routing algorithm for sensor networks. In *First Workshop on Sensor Networks and Applications (WSNA)*, 2002.

[4] N. Bulusu, J. Heidemann, and D. Estrin. Adaptive beacon placement. In *Proceedings of the 21st International Conference on Distributed Computing Systems (ICDCS-21)*, pages 489–498, 2001.

[5] T. Camp, J. Boleng, and V. Davies. A survey of mobility models for ad hoc network research. *Wireless Communications Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications, vol 2*, pages 483–502, 2002.

[6] R. Castaneda and S. Das. Query localization techniques for on-demand routing protocols in ad hoc networks. In *Proceedings of ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM'99)*, 1999.

[7] J. Chang and L. Tassiulas. Energy conserving routing in wireless ad hoc networks. In *Proceedings of the Nineteenth International Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, 2000.

[8] B. Chen, K. Jamieson, H Balakrishnan, and R. Morris. Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. In *Proceedings of 7th Annual International conference on Mobile Computing and Networking*, 2001.

[9] Y. Chen, E. Sirer, and S. Wicker. On selection of optimal transmission power for ad hoc networks. In *Proceedings of the Thirty-Sixth Hawaii International Conference on System Sciences (HICSS-36)*, 2003.

[10] P. Cheng, C.-N. Chuah, and X. Liu. Energy-aware node placement in wireless sensor networks. In *Proceedings of the Global Telecommunications Conference (GLOBECOM)*, 2004.

[11] X. Cheng, D. Du, L. Wang, and B. Xu. Relay sensor placement in wireless sensor networks. In *IEEE Transactions on Computers*, 2001.

[12] Z. Cheng and W. Heinzelman. Flooding strategy for target discovery in wireless networks. In *Proceedings of the 8th international workshop on Modeling analysis and simulation of wireless and mobile systems (MSWIM 2003)*, 2003.

[13] Z. Cheng and W. Heinzelman. Exploring long lifetime routing in ad hoc networks. In *Proc. of the 7th ACM international symposium on modeling, analysis and simulation of wireless and mobile systems (MSWIM 2004)*, 2004.

[14] E. G. Coffman, editor. *Computer and Job-shop Scheduling Theory*. John Wiley and Sons Inc., 1976.

[15] D. De Couto, D. Aguayo, B. Chambers, and R. Morris. Performance of multihop wireless networks: Shortest path is not enough. In *Proceedings of the First Workshop on Hot Topics in Networks (HotNets-I)*, 2002.

[16] D. C. Cox, R. Murray, and A. W. Norris. 800 mhz attenuation measured in and around suburban houses. *AT&T Bell Laboratory Technical Journal*, 1984.

[17] Crossbow technology. *http://www.xbow.com*.

[18] J. Deng, Y. Han, P. Chen, and P. Varshney. Optimum transmission range for wireless ad hoc networks. In *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC)*, 2004.

[19] NS-2 documentation. *http://www.isi.edu/nsnam/ns/ns-documentation*.

[20] R. Dube, C. D. Rais, K. Y. Wang, and S. K. Tripathi. Signal stability-based adaptive routing(ssa) for ad hoc mobile networks. In *IEEE Personal Communications, 4(1):36–45*, 1997.

[21] J. Elson, L. Girod, and D. Estrin. Fine-grained network time synchronization using reference broadcasts. In *Proceedings of the Fifth Symposium on Operating Systems Design and Implementation (OSDI)*, 2002.

[22] S. Coleri Ergen and P. Varaiya. Optimal placement of relay nodes for energy efficiency in sensor networks. In *Proceedings of the IEEE International Conference on Communications (ICC 2006)*, 2006.

[23] M. Gerharz, C. D. Waal, M. Frank, and P. Martini. Link stability in mobile wireless ad hoc networks. In *Proceedings of the 27th IEEE Local Computer Networks (LCN 2002)*, 2002.

[24] Z. J. Haas, M. R. Pearlman, and P. Samar. The zone routing protocol (ZRP) for ad hoc networks. *IETF MANET Internet Draft*, 2002.

[25] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan. An application-specific protocol architecture for wireless microsensor networks. *IEEE Transactions on Wireless Communications*, 1(4):660–670, October 2002.

[26] W. R. Heinzelman, J. Kulik, and H. Balakrishnan. Adaptive protocols for information dissemination in wireless sensor networks. In *Proceedings of the fifth annual ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom 99)*, 1999.

[27] G. Holland and N. H. Vaidya. Analysis of TCP performance over mobile ad hoc networks. In *Proceedings of Mobile Computing and Networking*, pages 219–230, 1999.

[28] I. Howitt and J. Wang. Energy balanced chain in wireless sensor networks. In *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC)*, 2004.

[29] Y.-C. Hu and D. B. Johnson. Caching strategies in on-demand routing protocols for wireless networks. In *Proceedings of the 6th annual ACM/IEEE international conference on Mobile computing and networking (MobiCom 2000)*, 2000.

[30] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *Proceedings of Mobile Computing and Networking*, pages 56–67, 2000.

[31] D. B. Johnson and D. A. Maltz. Dynamic source routing in ad hoc wireless networks. In Imielinski and Korth, editors, *Mobile Computing*, volume 353, pages 153–181. Kluwer Academic Publishers, 1996.

[32] JXTA. *http://www.jxta.org*.

[33] A. Kansal, M. Rahimi, D. Estrin, W. Kaiser, G. Pottie, and M. Srivastava. Controlled mobility for sustainable wireless sensor networks. In *Proceedings of The First IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON)*, 2004.

[34] Kazaa. *http://www.kazaa.com*.

[35] H. Kim, T. Abdelzaher, and W. Kwon. Minimum-energy asynchronous dissemination to mobile sinks in wireless sensor networks. In *Proceedings of the First ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2003.

[36] Y. B. Ko and N. H. Vaidya. Location-aided routing (LAR) in mobile ad hoc networks. In *Proceedings of ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'98)*, 1998.

[37] B. Krishnamachari and J. Ahn. Optimizing data replication for expanding ring-based queries in wireless sensor networks. In *Proceedings of International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks(WIOPT 06)*, 2006.

[38] L. Li, J. Halpem, and Z. J. Hass. Gossip-based ad hoc routing. In *Proceedings of IEEE INFOCOM*, pages 1707–1716, 2002.

[39] S. Lindsey, C. Raghavendra, and K. Sivalingam. Data gathering algorithms in sensor networks using energy metrics. *IEEE Transactions on Parallel and Distributed Systems*, 13(9):924–935, September 2002.

[40] J. Luo and J. Hubaux. Joint mobility and routing for lifetime elongation in wireless sensor networks. In *Proceedings of the 24th IEEE INFOCOM*, 2005.

[41] M. K. Marina and S. R. Das. Performance of route caching strategies in dynamic source routing. In *Proceedings of the Int'l Workshop on Wireless Networks and Mobile Computing (WNMC) in conjunction with Int'l Conf. on Distributed Computing Systems (ICDCS)*, 2001.

[42] A. B. McDonald and T. Znati. A path availability model for wireless ad-hoc networks. In *Proceedings of IEEE Wireless Communications and Networking Conference 1999 (WCNC'99)*, 1999.

[43] D. Mills. Internet time synchronization: the network time protocol. *IEEE Transactions on Communications*, 39(10):1482–1493, October 1991.

[44] D. S. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins, and Z. Xu. Peer-to-peer computing. *Technical Report HPL-2002-57, HP Lab*, 2002.

[45] K. Nahm, A. Helmy, and C. Kuo. TCP over multihop 802.11 networks: issues and performance enhancement. In *Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing*, 2005.

[46] Tim Oates, M. V. N. Prasad, and Victor R. Lesser. Cooperative information gathering: A distributed problem solving approach. In *Computer Science Technical Report 94-66-version 2 , University of Massachusetts*.

[47] N. Panchal and N. B. Abu-Ghazaleh. Active route cache optimization for on-demand ad hoc routing protocols. In *Proceedings of International Conference on Networking (ICN'04)*, 2004.

[48] Gnutella peer-to-peer file sharing system. *http://www.gnutella.com*.

[49] M. Perillo, Z. Cheng, and W. Heinzelman. On the problem of unbalanced load distribution in wireless sensor networks. In *Proceedings of the Global Telecommunications Conference (GLOBECOM) Workshop on Wireless Ad Hoc and Sensor Networks*, 2004.

[50] M. Perillo, Z. Cheng, and W. Heinzelman. An analysis of strategies for mitigating the sensor network hot spot problem. In *Proceedings of The Second Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services*, pages 474–478, 2005.

[51] M. Perillo and W. Heinzelman. DAPR: A protocol for wireless sensor networks utilizing an application-based routing cost. In *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC)*, 2004.

[52] C. Perkins and E. M. Royer. Ad hoc on-demand distance vector routing. In *Proceedings of IEEE Workshop on Mobile Computing Systems and Applications (WMCSA '99)*, pages 90–100, 1999.

[53] R. Ramanathan and R. Hain. Topology control of multihop wireless networks using transmit power adjustment. In *Proceedings of the Nineteenth International Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, 2000.

[54] V. Ramasubramanian, Z. J. Haas, and E. G. Sirer. Sharp: A hybrid adaptive routing protocol for mobile ad hoc networks. In *Proceedigns of ACM Symposium on Mobile Ad Hoc Networking andi Computing (MobiHOC)*, 2003.

[55] Theodore S. Rappaport, editor. *Wireless Communications Principles and Practice*. Prentice Hall PTR, 1999.

[56] V. Rodoplu and T. Meng. Minimum energy mobile wireless networks. In *Proceedings of the IEEE International Conference on Communications (ICC)*, 1998.

[57] M. Roman, C. K. Hess, R. Cerqueira, A. Ranganathan, R. H. Campell, and K. Nahstedt. Gaia: A middleware infrastructure to enable active spaces. In *Proceedings of IEEE Pervasive Computing*, pages 74–83, 2002.

[58] N. Sadagopan, B. Krishnamachari, and A. Helmy. Active query forwarding in sensor networks (ACQUIRE). *Ad Hoc Networks Journal - Elsevier Science 1st Quarter, vol 3(1)*, pages 91–113, 2004.

[59] R. Shah, S. Roy, S. Jain, and W. Brunette. Data MULEs: Modeling a three-tier architecture for sparse sensor networks. In *Proceedings of the First IEEE Workshop on Sensor Network Protocols And Applications (SNPA)*, 2003.

[60] J. Sucec and I. Marsic. An application of parameter estimation to route discovery by on-demand routing protocols. In *Proceedings of the 21st International Conference on Distributed Computing Systems (ICDCS 2001)*, pages 207–216, 2001.

[61] System and Netowrk Administration. *Sun Mircorsystems*, 1990.

[62] C.K. Toh. Associativity-based routing for ad hocmobile networks. *International Journal on Wireless Personal Communications, Vol.4, No.2*, 1997.

[63] X. Wang, G. Xing, Y. Zhang, C. Lu, R. Pless, and C. Gill. Integrated coverage and connectivity configuration in wireless sensor networks. In *Proceedings of the First ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2003.

[64] E. Woodrow and W. Heinzelman. Spin-it: A data centric routing protocol for image retrieval in wireless networks. In *Proceedings of International Conference on Image Processing (ICIP'02)*, 2000.

[65] T. Wu, M. Malkin, and D. Boneh. Building intrusion tolerant application. In *Proceedings of the 8th USENIX Security Symposium*, 1999.

[66] F. Ye, H. Luo, J. Cheng, S. Lu, and L. Zhang. A two-tier data dissemination model for large-scale wireless sensor networks. In *Proceedings of the Eighth Annual International Conference on Mobile Computing and Networks (MobiCom)*, 2002.

[67] F. Ye, G. Zhong, J. Cheng, S. Lu, and L. Zhang. Peas: A robust energy conserving protocol for long-lived sensor networks. In *Proceedings of the Twenty-Third International Conference on Distributed Computing Systems (ICDCS)*, 2003.

[68] O. Younis and S. Fahmy. Distributed clustering in ad-hoc sensor networks: A hybrid, energy-efficient approach. In *Proceedings of the Twenty-Third Annual Joint*

Conference of the IEEE Computer and Communications Societies (INFOCOM), 2004.

[69] X. Yu. Improving TCP performance over mobile ad hoc networks by exploiting cross-layer information awareness. In *Proceedings of the 10th annual international conference on Mobile computing and networking*, pages 231–244, 2004.

[70] G. Zussman and A. Segall. Energy efficient routing in ad hoc disaster recovery networks. In *Proceedings of IEEE INFOCOM*, 2003.

Appendix A

Publications

A.1 Journal Publications

Z. Cheng and W. Heinzelman, “Discovering Long Lifetime Routes in Mobile Ad Hoc Networks,” in submission.

Z. Cheng, M. Perillo and W. Heinzelman, “General Network Lifetime and Cost Models for Evaluating Sensor Network Deployment Strategies,” in submission.

Z. Cheng and W. Heinzelman, “Searching strategies for Target Discovery in Wireless Networks,” to be published in Elsevier Journal of Ad Hoc Networks.

Z. Cheng and W. Heinzelman, “Adaptive Local Searching and Caching Strategies for on-demand routing protocols in ad hoc networks,” to be published in Mobile and Wireless Networking of International Journal of High Performance Computing and Networking (IJHPCN).

Z. Cheng and W. Heinzelman, “Flooding Strategy for Target Discovery in Wireless Networks,” ACM/Baltzer Wireless Networks, 2005.

A.2 Conference Publications

M. Perillo, Z. Cheng and W. Heinzelman, “Strategies for Mitigating the Sensor Network Hot Spot Problem,” Proceedings of Collaboratecom 2005, Nov. 2004. (Invited)

M. Perillo, Z. Cheng and W. Heinzelman, “An Analysis of Strategies for Mitigating the Sensor Network Hot Spot Problem,” Proceedings of Mobiquitous 2005, July, 2005.

M. Perillo, Z. Cheng and W. Heinzelman, “On the Problem of Unbalanced Load Distribution in Wireless Sensor Networks,” Proceedings of GLOBECOM, Nov. 2004.

Z. Cheng and W. Heinzelman, “Exploring long lifetime routing in ad hoc networks,” Proceedings of the 7th ACM MSWIM 2004, Oct. 2004.

Z. Cheng and W. Heinzelman, “Adaptive local searching and caching strategies for on-demand routing protocols in ad hoc networks,” Proceedings of the 33rd MWN, Aug. 2004.

Z. Cheng and W. Heinzelman, “Searching Strategies for Multi-Target Discovery in Wireless Networks,” Proceedings of the 4th ASWN 2004, Aug. 2004.

Z. Cheng and W. Heinzelman, “Flooding Strategy for Target Discovery in Wireless Networks,” Proceedings of the 6th ACM MSWiM 2003, Sep. 2003. (Best Paper Award)

Z. Cheng, M. Perillo, B. Tavli, W. Heinzelman, S. Tilak, and N. Abu-Ghazaleh, “Protocols for local data delivery in wireless microsensor networks,” 45th IEEE MWSCAS 02, Aug. 2002. (Invited)