Improving QoS and Management in Multi-Hop Wireless Networks

by

Kyu-Han Kim

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Computer Science and Engineering)
in The University of Michigan
2009

Doctoral Committee:
Professor Kang G. Shin, Chair
Professor Dawn Tilbury
Associate Professor Jason N. Flinn
Associate Professor Brian D. Noble

To my family

ACKNOWLEDGEMENTS

I would like to express my deep appreciation to my advisor, Professor Kang G. Shin, for his unwavering support and encouragement during my study. I have benefited tremendously from his advice and have learned the fundamentals of research as well as of life. I also thank Professors Dawn Tilbury, Brian D. Noble, and Jason N. Flinn for serving on my dissertation committee, and for their valuable feedback and suggestions.

I am truly fortunate to have a loving and supportive family. My wife, You-Jung Kim, has been a friend, a supporter, and a critic throughout this entire journey. My loving one-year-old son, Ryan Seung-Yoon Kim, has given indescribable happiness every moment. My parents have constantly provided me endless love, encouragement, and support over the years. This dissertation is dedicated to them.

I am also grateful to my many talented and friendly colleagues. First, I thank the RTCL members for their friendship and supports, especially Min-gyu Cho, Zhigang Chen, Jisoo Yang, Katharine Chang, Hyoil Kim, Alex Min, Karen Hou, Pradeep Padala, Sujay Phadke, Ashwini Kumar, Karen Hou, Xin Hu, Eugene Chai, and others. I also thank my collaborators and former mentors, Dr. Dragoş Niculescu at NEC Laboratories America, Professor Raghupathy Sivakumar at Georgia Institute of Technology, and Professor Hung-Yun Hsieh at National Taiwan University. Finally, I thank the RTCL/SSL alumni and staff at the University of Michigan, including Jian Wu, Howard Tsai, Wei Sun, Mohamed El Gendy, Hai Huang, Taejoon Park, Jai-Jin Lim, Magesh Jayapandian, and Stephen Reger. I wish the very best in their careers and life.

Finally, I would also like to thank the funding agencies, including NSF, AFSOR, Intel, NEC Laboratories America, and the Ministry of Information and Communication, Republic of Korea.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF TABLES

LIST OF APPENDICES

APPENDIX

CHAPTER 1

Introduction

With ever-increasing demands for pervasive access to the Internet, multi-hop wireless networks have been actively developed and widely deployed. A recent study [76] has shown that more than 340 cities and counties in the U.S. have been deploying multi-hop or large-scale wireless networks. In addition, these multi-hop networks have created new applications and services, including municipal broadband Internet, emergency response systems, and climate/environment monitoring systems. By using inexpensive wireless links such as IEEE 802.11, multi-hop networks can easily provide network connectivity over target areas without deploying expensive wire-lines. Furthermore, because of their multi-hop nature, these networks can significantly improve spectrum utilization and network reliability by increasing spatial spectrum reuse as well as redundancy in network topology (e.g., mesh), compared to traditional cellular networks.

This thesis first explores the potential of such multi-hop wireless networks for fast, easy, and inexpensive network connectivity. Then, it proposes system-level frameworks and techniques that will significantly improve Quality-of-Service (QoS) and network management cost, which are essential to the realization of the network's potential.

1.1 Multi-Hop Wireless Networks: Model and Promise

A multi-hop wireless network is nothing more than a distributed system that consists of physically distributed wireless routers and multi-hop wireless links, as illustrated in

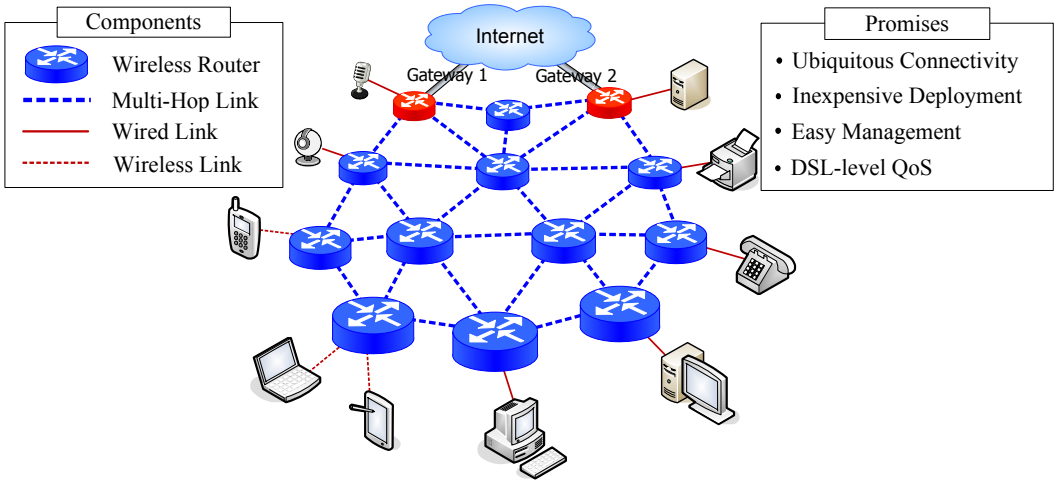


Figure 1.1: *Architecture and promises of multi-hop wireless networks*: A multi-hop wireless network consists of geographically-distributed routers, multi-hop links, and a few gateways. This multi-hop wireless network can provide several important benefits such as deployment costs over wired network counterparts.

Figure 1.1. First, wireless routers in the network are (quasi-)stationary and are deployed in street-lights/roofs (outdoor) or ceiling (indoor). Next, these routers are interconnected through multi-hop wireless links (IEEE 802.11 or IEEE 802.16), and a few routers (i.e., gateways) are connected to the Internet through wire-lines. In addition, edge routers in the network are equipped with wired (e.g., Ethernet) or wireless (e.g., IEEE 802.11) links to provide wireless connectivity to users. These settings can change, depending on the network's application scenarios. For example, for broadband Internet services in residential areas, network nodes are equipped with multi-radio interfaces for high bandwidth. For video surveillance or disaster monitoring systems, wireless links are assumed to provide guaranteed end-to-end bandwidth or delay, and network nodes are quasi-stationary for real-time deployments.

Given this model, multi-hop wireless networks can provide several benefits, including deployment and management costs, over wired networks. First, they can quickly provide wireless network connectivity to areas where real-time network connectivity is necessary for such applications as disaster monitoring and emergency response systems. Second, their coverage can easily be extended by simply placing additional wireless routers in the new areas to be covered. Third, they reduce network management cost significantly by avoiding the deployment of wired cables as well as by adopting robust routing protocols.

Finally, multi-hop wireless networks can outperform DSL (Digital Subscriber Lines)-level QoS (1 Mbps or a little more).

1.2 Challenges and Limitations in QoS Support

Despite the potential benefits in multi-hop wireless networks, a recent study has shown that multi-hop wireless networks often deliver poor QoS and, thus, end-users experience intermittent connectivity or low end-to-end throughput. Many Internet-based real-time applications such as Voice-over-IP (VoIP) and video streaming require the networks to support a certain level of bandwidth and delay. However, because wireless link-quality fluctuates over time, networks fail to meet the application QoS requirements. In addition, inaccurate or obsolete information about the underlying network condition causes the networks to be under-utilized, even if they can serve additional application traffic.

Many solutions have been proposed to improve QoS in wireless networks, but they assume simple or impractical link-quality models, which cause poor QoS or increased management costs in practice. First, mobile ad-hoc networks (MANETs) have been investigated with emphasis on maintaining connectivity during user mobility [43, 117] or with limited resources [56, 83, 105]. However, they rely on simplified wireless link models (e.g., unit-disc model), which are shown to be impractical to provision QoS [31]. Next, wireless sensor networks (WSNs) have been studied extensively. However, they focus mainly on improving the energy efficiency of network protocols [6] under simple wireless link models. Third, wireless mesh networks (WMNs) have been actively developed, and realistic wireless link models (e.g., probabilistic links) have been considered [3, 30, 36]. However, they do not fully identify and exploit the characteristics of wireless links (e.g., asymmetry). Finally, various wireless network monitoring or spectrum survey frameworks have been proposed [1, 4, 49, 88, 92] to adaptively configure QoS parameters. However, they rely mostly on off-line processing of network traces and also incur significant deployment/management costs.

1.3 Challenges and Limitations in Network Management

Another challenge in multi-hop wireless networks is to make them deal autonomously with various types of wireless link failures or environmental changes to preserve QoS. Even though multi-hop wireless networks have the advantage of easy, fast, and inexpensive deployment over wired networks, their dependency on physical locations causes various network management problems.

First, because the quality of wireless links fluctuates with time as well as space, network engineers are required to frequently monitor and fix the fluctuation-induced problem or wireless link failures. Various routing protocols, such as self-organizing and link-quality-aware routing [77, 122], have been proposed to solve the problem, they merely deal with transient link failures, and hence, networks experience frequent QoS failures in the case of long-term link failures (e.g., interference) that are common in today's *chaotic* network deployment environments [5].

Next, wireless nodes in certain locations may suffer from environmental changes, such as new obstacles or interferences, and their locations or network settings, such as channel assignments, need to be changed to satisfy the users' QoS demands. Managing manual re-location or reconfiguration of network nodes requires prohibitive effort, including extensive spectrum site-survey and manual adjustment of node locations. There are many solutions to solve these problems by tuning the underlying network parameters, such as transmission power and data-transmission rate [113, 121]. Even though in some cases these parameters are effective, relocating nodes or adding new nodes is necessary if the improvement achieved by controlling the parameters is limited by physical environments. For example, nodes using the maximum transmission power behind obstacles cannot achieve maximum network capacity, but degrade overall channel utilization, compared to relocating nodes.

Third, real-time deployment of wireless nodes puts additional burden on management. Many researchers have focused on off-line network planning and optimization [17, 59], but little has been done on real-time deployment and optimization of networks. Some researchers have recently proposed the trajectory-based deployment of wireless sensor nodes [107]. However, after the initial deployment of stationary sensor nodes, networks

cannot physically adapt to changing network conditions. Alternatively, hybrid networks consisting of stationary sensors and mobile robot nodes allow for the partial adjustment of their topology by moving a few mobile robots based on node density [111]. However, this approach does not consider the spatial characteristics of wireless links, which are critical for optimizing overall network performance.

Finally, even deployed wireless networks need to be monitored for assessment, extension, and improvement of overall network performance. Many network assessment techniques (e.g., [99]) and spectrum site-survey tools [4, 92, 110] have been proposed. Network engineers can navigate through deployment areas and evaluate the performance of wireless routers by using these solutions. Even though they can automate a part of the spectrum survey, network engineers have to repeatedly navigate through the same areas if the networks experience frequent environmental changes or QoS failures. Furthermore, as more RF-based applications and services (e.g., a localization system [119]) have been introduced, performing spectrum site-survey causes a dramatic increase in network management costs.

To reduce increased management costs and improve network QoS, this thesis proposes to enable multi-hop wireless networks to autonomously adapt to fundamental network parameters, such as link conditions and node locations.

1.4 Research Goals: QoS-aware Autonomous Management

One of fundamental approaches to enabling QoS-aware autonomous management is to incorporate *awareness* of, and *adaptation* to wireless link-conditions into multi-hop wireless networks. This thesis presents algorithms and frameworks that allow wireless networks to (i) capture the quality of wireless links and (ii) adapt to the underlying quality and users' QoS demands.

- **Link-quality awareness**: For provisioning QoS, highly accurate link-quality information is necessary. The quality of wireless links is a metric for network QoS. Modeling wireless links could provide theoretical understanding of wireless link characteristics, but such models often fail to capture characteristics specific to heterogeneous deployment environments. This research, by contrast, takes a measurement-

based approach for making wireless systems accurately capture environmental factors about link conditions. From the measurement results, networks can obtain site-specific spectrum propagation and thus yield highly accurate link-quality information.

- **Channel- and radio-assignment adaptation**: To continuously support QoS in the presence of link failures, channel/radio adaptation algorithms are necessary. Based on the link-quality awareness, wireless networks can adaptively reconfigure their settings by using diverse network resources. Considerable research to make use of network-level diversity—multi-path [77, 122]—exists. By contrast, this thesis focuses on the use of multi-channel and multi-radio diversities. By using multiple orthogonal channels—13 in IEEE 802.11a and 3 in IEEE 802.11b/g—and multiple interfaces (or radios) available in wireless routers, multi-hop wireless networks can solve critical channel-related link failures under changing environments or QoS demands.

- **Node-placement optimization and adaptation**: To enable QoS-aware deployment with minimum costs, optimization and adaptation algorithms about node placement are needed. The performance of wireless links is affected by the physical location of neighboring network nodes. Even after the deliberate placement of the nodes, network engineers still need to adjust their locations not only for improving network performance but also for dealing with frequent wireless link failures. This thesis studies the feasibility of designing new wireless routers that can make use of such spatial diversity for adaptive satisfaction of user QoS demands.

- **Spatial spectrum-condition awareness**: Spatial spectrum-conditions need to be monitored to reduce management costs and to support other applications, including indoor localization system. Spectrum site-surveys provide useful information for both network management and network QoS. Furthermore, many applications such as an RF-based localization system require the spatial spectrum footprints of the wireless networks. This thesis designs, implements, and evaluates a novel spectrum site-survey system, especially for challenging indoor environments (e.g., office

buildings).

1.5 Research Contributions

Motivated by the above research goals, the primary contributions of this thesis are to design, develop, implement, and evaluate *monitoring* and *adaptation* algorithms and frameworks, through which multi-hop wireless networks can collect link condition information (self-monitoring), reconfigure channel/radio assignment (self-reconfiguration), optimize node locations (self-optimization), and provide a tool to automate spectrum site-surveys (autonomous survey). As shown in Figure 1.2, this thesis focuses mainly on wireless mesh and relay networks, where QoS and management costs are crucial to operation. Moreover, the thesis proposes systems that can be implemented and prototyped using open-source software and off-the-shelf products to demonstrate their feasibility and practicality.

1.5.1 Self-Monitoring

To provision end-to-end network QoS, multi-hop wireless networks must be able to keep track of wireless links conditions (i.e., awareness illustrated in Figure 1.2 (b)). This thesis presents an efficient and accurate link-quality monitoring framework for an IEEE 802.11-based wireless mesh network. As a QoS-building block, this framework measures the link quality and provides quality information to other network applications and services (e.g., routing protocol).

The proposed monitoring framework addresses three core challenges: efficiency, accuracy, and link-asymmetry. First, the framework improves both efficiency and accuracy in measurements by introducing novel hybrid measurement techniques. Next, this proposed framework enables the identification and use of asymmetric wireless links, which are important for QoS, but have often been ignored. Third, this framework includes cross-layer designs that improve measurement accuracy, keeping the framework easy to deploy over existing wireless mesh networks.

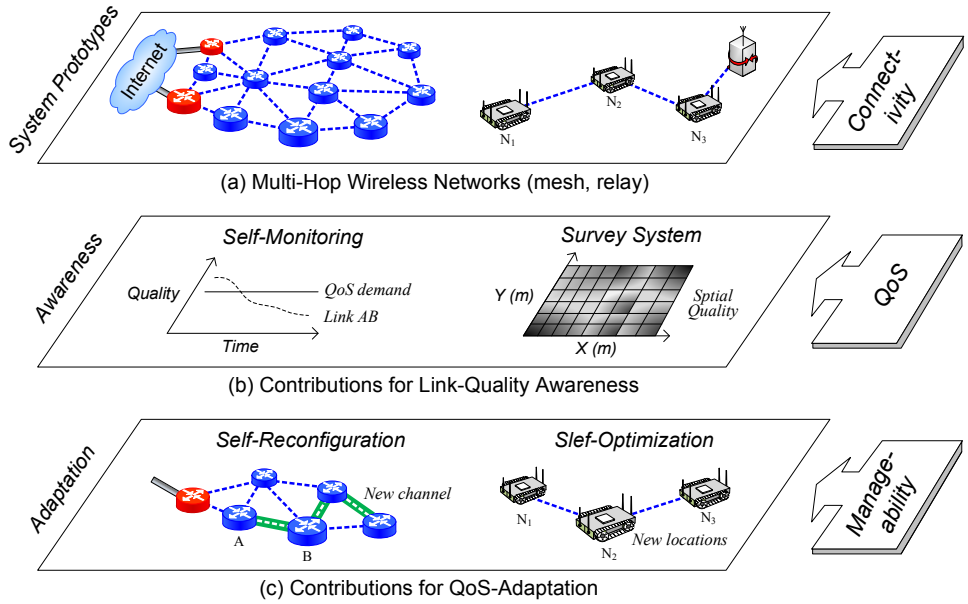


Figure 1.2: *Main research contributions*: Multi-hop wireless networks can achieve QoS and autonomous managements through self-monitoring, self-reconfiguration, self-optimization, and automatic survey on wireless link-conditions.

1.5.2 Self-Reconfiguration

Due to frequent wireless link-failures, multi-hop networks must be able to recover from failures by reconfiguring network resources, such as multi-channel and multi-radio interfaces, to continue meeting QoS demands (i.e., adaptiveness depicted in Figure 1.2 (c)). This thesis presents a localized self-reconfiguration algorithm that enables wireless routers in a faulty area to cooperatively reconfigure their channel and link assignments with minimum network disruption.

This proposed framework handles three key challenges: monitoring, localized planning, QoS-aware planning. First, the framework enables real-time network reconfiguration by detecting link failures based on self-monitoring. Next, the framework allows a group of local wireless routers to generate a reconfiguration plan that matches locally-available network resources with changing QoS demands. Third, the proposed framework minimizes the adverse effects of local reconfiguration on other parts of networks (i.e., ripple effects) by incorporating spatial spectrum reuse into the reconfiguration planning.

1.5.3 Self-Optimization

Another dimension of adaptation is to use the spatial diversity of network nodes for optimizing the performance of a multi-hop network (adaptiveness illustrated in Figure 1.2 (c)). This thesis studies the feasibility of using the physical mobility of wireless routers for improving QoS and proposes a mobile autonomous router system. This system enables each wireless router, mounted on a robot, to autonomously move and determine its best reception location and form relay networks with neighboring routers.

The proposed system handles four core challenges: spatial link-quality measurement, spatial probing algorithm, localization, and feasibility. First, the system accurately measures spatial wireless link-conditions through grid-based measurements. Second, by using the measured spatial link-quality information, the system efficiently finds the best reception positions through a strategy of hierarchical navigation. Third, this system also includes an error-tolerant positioning system in which each router can maintain accurate node location information during navigation. Finally, the proposed system is designed to be feasible and has been implemented and prototyped with a commodity mobile robot and an IEEE 802.11-based wireless router.

1.5.4 Automatic Survey

Even after deployment, multi-hop wireless networks and their spectrum usage need to be periodically monitored (awareness depicted in Figure 1.2(b)). This thesis presents a novel spectrum survey system that automates labor-intensive spectrum site-survey by using robot-based mobility.

The proposed systems address three key challenges in spectrum survey: repeatability, efficiency, and adaptability. First, the system not only extracts spatio-temporal variance in spectrum conditions, but also generates repeatable spectrum maps. Second, the system minimizes the measurement space by adaptively using different, but complementary, measurement techniques. Third, the system intelligently determines measurement granularity (e.g., measurement distance) that creates a trade-off between efficiency and accuracy in spectrum site-survey.

1.6 Organization of the Thesis

The rest of this thesis is organized as follows. Chapter 2 presents an efficient and accurate link-quality monitor (EAR). EAR efficiently monitors wireless link-quality parameters such as packet-delivery ratio and data-transmission rate in a WMN. Chapter 3 proposes localized self-reconfiguration algorithms (LEGO). LEGO allows a multi-radio WMN to autonomously recover from local link failures. Chapter 4 presents a mobile autonomous router system (MARS). MARS enables a wireless router to characterize spatial link-quality and allows a group of mobile wireless routers to form a real-time multi-hop relay network. Chapter 5 proposes a spectrum survey robot (Sybot). Sybot automates labor-intensive spectrum site-survey for large-scale wireless networks. Finally, the thesis concludes with Chapter 6.

CHAPTER 2

Efficient and Accurate Measurement of Link-Quality in Wireless Mesh Networks

2.1 Introduction

Due to their deployment in large and heterogeneous areas and their use of open wireless media, wireless links often experience significant quality fluctuations and performance degradation or weak connectivity [3, 31]. To deal with such wireless link characteristics, there have been significant efforts to improve the network performance by reducing the overheads associated with unexpected link-quality changes. For example, ExOR [14, 15] is a routing protocol that tries to reduce the number of retransmissions via cooperative diversity among neighboring nodes. MASA [120] is a MAC-layer approach that tries to minimize the overhead in recovering lost frames via nearby "salvaging" nodes. Finally, NADV [65] is a link metric that assists a geographic routing protocol to choose the relay node by optimizing the trade-off between proximity and link quality.

In addition to the above efforts, accurate measurement of wireless link quality is essential to dealing with link-quality fluctuations for the following reasons. First, the above-mentioned three solutions rely heavily on accurate link-quality information to select the best relay nodes. Second, applications, such as video streaming and VoIP, also need the link-quality information to support QoS guarantees over WMNs. Third, diagnosing a network, especially a large-scale WMN, requires accurate long-term statistics of link-quality

information to pinpoint the source of network failures, and reduce the management overhead [88]. Finally, WMNs commonly use multiple channels [11, 36, 90], and determining the best-quality channel among multiple available channels requires the information on the quality of each channel.

There are, unfortunately, several limitations in using existing techniques to measure the quality of links in WMNs. First, Broad-cast-based Active Probing (BAP) has been widely used for link-quality-aware routing [15, 30, 36]. Although it incurs a small overhead (e.g., 1 packet per second), broadcasting does not always generate the same quality measurements as actual data transmissions due to different PHY settings (e.g., modulation). Thus, BAP provides inaccurate link-quality measurements. Moreover, its use of an identical type of probing in *both* directions of a link generates *bi-directional* results, thus un-/under-exploiting link asymmetry. Second, unicast-based probing provides accurate and *uni-directional* results owing to its resemblance to the use of actual data transmissions, but it incurs significant overheads. Finally, passive monitoring [65] is the most efficient and accurate since it uses actual data traffic. However, it also incurs the overhead of probing idle links.

To overcome the above limitations of existing measurement techniques, we propose a high-accuracy and low-overhead distributed measurement framework, called EAR, that has the following three salient features. First, EAR consists of three complementary measurement schemes—passive, cooperative, and active monitoring—that commonly use *unicast* for its accuracy and "opportunistically" exploit the egress/cross traffic of each node for efficiency. Using unicast, all three schemes measure link-quality under the same setting as the actual data transmission, thus yielding accurate results. By exploiting data traffic in the network as probe packets, and dynamically and adaptively selecting the most effective of the three schemes, EAR not only reduces the probing overhead, but also decreases the measurement variations, thanks to the large number of "natural" probe (i.e., real traffic) packets.

Second, EAR's link-quality measurement is made *direction-aware* to effectively capitalize on link asymmetry. Wireless link quality is often asymmetric due to such environmental factors as hidden nodes, obstacles and weather conditions [30, 69, 73]. The better-

quality direction of an asymmetric link might often be good enough to transmit data frames in that direction, instead of taking a longer detour path. By direction-aware measurement of link quality from actual data transmissions and ACK receptions, EAR can identify and exploit link asymmetry, thus improving the utilization of network capacity.

Finally, EAR is designed to run in a fully-distributed fashion and to be easily deployable on existing IEEE 802.11x-based WMNs. It runs on each node and periodically measures the quality of link to each of its neighbors to maintain up-to-date link-quality information. On each node, EAR is implemented at the network layer and a device driver, and intelligently uses several features of the MAC layer, such as transmission results and data rate, by interacting with the MAC Management Information Base (MIB) [50]. Moreover, this design does not require any system change or MAC firmware modification, thus making its implementation and deployment easy.

We conduct an in-depth evaluation of EAR via both *ns-2*-based simulation and experimentation on a Linux-based implementation. Our simulation results show that EAR's unicast-based techniques decrease the root mean-square error in measurements by at least a factor of 4 over the broadcast-based approach, while reducing the overhead by an average of 50%, even in large-scale WMNs. Moreover, EAR's direction-aware link-quality measurement enables the opportunistic use of asymmetric links and helps the underlying routing protocol find the best-quality relay node, thus improving channel efficiency by up to 49%.

EAR is implemented as a routing component along with the device driver of Orinoco 802.11b, and then evaluated on our experimental testbed. Experimental results show that EAR effectively exploits existing application traffic in measurement (up to 13 times more probing packets than BAP's). In addition, our measurement results show that there exist many asymmetric links in different time scales (from a few to dozens of minutes), and that EAR's uni-directional measurement helps the routing protocol improve the end-to-end throughput by up to 114%.

The rest of this chapter is organized as follows. Section 2.2 describes the motivation of this work. Section 2.3 presents the EAR architecture and algorithms. Section 2.4 evaluates EAR using *ns-2*-based simulation, and Section 2.5 describes our implementation and

experimental results on our testbed. Section 2.6 discusses the remaining issues associated with EAR, and finally concludes the chapter.

2.2 Motivation

We first advocate the importance of accurate measurements of varying wireless link quality to WMNs. Then, we identify the limitations in applying existing measurement techniques to WMNs.

2.2.1 Why Accurate Link-Quality Measurement?

Wireless link quality varies with environmental factors, such as interference, multi-path effects and even weather conditions [3, 46, 60]. Especially, in multi-hop WMNs, due to their usual deployment in large and heterogeneous areas, wireless link quality fluctuates significantly, and thus, the various network protocols, such as the shortest-path and geographic routing protocols, designed under the strong link-quality assumption[1] often suffer performance degradation or weak connectivity [3, 31, 60].

Accurate link-quality measurement is essential to solve the problem associated with varying link-quality in WMNs, as one can see from the following use-cases.

- *Selection of the best relay node*: Accurate link-quality information can reduce the recovery cost of lost frames caused by link-quality fluctuations. For example, ExOR [14, 15] and MASA [120] attempt to reduce the number of transmissions with the help of intermediate relay nodes in retransmitting lost frames. Both solutions are based on *capture effects* that allow in-range nodes to cooperatively relay "overheard" frames, but one key question is how to select the relay node that has the best link-quality.
- *Supporting Quality-of-Service (QoS)*: Wireless link-quality information enables applications and network protocols to effectively meet users' QoS requirements. For example, applications, such as VoIP and IPTV, can dynamically adjust their service level that can be sustained by varying link-quality in the network. On the other hand, link-

[1] For example, *if I can hear you at all, I can hear you perfectly.*

quality-aware routing protocols [30, 36] can accurately locate a path that satisfies the QoS (e.g., throughput and delay) requirements based on the link-quality information.

- *Network failure diagnosis*: Link-quality statistics can be used to diagnose and isolate faulty nodes/links (or faulty areas) in WMNs, facilitating network management [29, 88]. WMNs covering shopping malls, a campus or a city, usually consist of a number of nodes, and each node must deal with site-specific link conditions. Thus, WMNs require a clear picture of local link conditions for network troubleshooting.
- *Identifying high-quality channels*: Link-quality information helps WMNs identify high-quality channels. WMNs usually use multiple channels to reduce interference between neighboring nodes [36, 46, 90]. However, due to the use of shared wireless media, link-quality differs from one channel to another, and hence, determining the best-quality channel is of great importance to channel-assignment algorithms, such as those in [8, 59].

Motivated by these and other use-cases, we would like to address how to measure link-quality and how beneficial accurate measurements can be in utilizing the given network capacity.

2.2.2 Limitations of Existing Techniques

There has been a significant volume of work on link-quality measurement. We discuss pros and cons of using existing techniques for WMNs.

Accuracy and efficiency A measurement technique must yield accurate results at as low a cost as possible. First, Broadcast-based Active Probing (BAP) has been widely used for adopting link-quality-aware routing metrics such as Expected Transmission Count (ETX) [30] and Expected Transmission Time (ETT) [36]. It uses simple broadcasting of probe packets from each node and derives link-quality information by multiplying the percentage of successful transmissions in each direction. Although it is inexpensive, broadcasting uses a fixed and low data rate (2Mbps), which is more tolerant of bit errors than other rates, and which may differ from the actual data-transmission rate (e.g., 11Mbps). Thus, as we

will show later (in Figure 2.13), BAP yields less accurate link-quality information than a unicast-based approach (e.g., 10.2% error by broadcast vs. 1.6% error by unicast).

Next, the unicast-based approach to measuring link bandwidth [34, 35, 36] can yield accurate results as it uses the same data rate for probing a link as that for actual data transmissions over the link. However, frequent probing of link to each neighbor incurs a higher overhead than BAP. As the number of neighbors increases, probe packets might throttle the entire channel capacity.

Finally, without injecting probe packets, passive monitoring yields accurate link-quality measurements without incurring any overhead. Signal-to-Noise Ratio (SNR) monitoring may be the cheapest, but it is shown to be not strongly related to actual link-quality [3]. Self-monitoring [65] could be attractive due to its use of actual data-frame transmission results. However, it also incurs a large overhead in probing links when there are no data packets sent over them.

Link-asymmetry-awareness Measurement schemes must be able to identify and exploit wireless link asymmetry that results from interference, obstacles, or weather conditions [30, 69, 73]. For example, if there is interference in the vicinity of node *A*, then signals from a remote node *B* to *A* might be disrupted, whereas signals from node *A* are normally strong enough to overcome the interference. While *B* might reach *A* via node *C* that has high-quality links to both *A* and *B*, node *A* can use the direct link to *B*, thus saving network resources.

First, BAP has limited asymmetry-awareness. It was originally designed to be aware of link asymmetry [30, 36]. BAP independently measures the quality[2] of the link's both directions, and then multiplies them. However, the results are bi-directional—giving the same link quality in both directions—due to the same type of probing used in both directions, and often under-estimate the quality of asymmetric links. Figure 2.1 is a sample measurement result of BAP over an asymmetric link on our testbed. The figure shows that although one direction of link has good quality (upper curve), the measurement result via bi-directionality often under-rates the quality of the link's both directions (lower curve).

[2]For the time-being we use the delivery ratio of data frames of link A→B as the link quality. We will elaborate on this in Section 2.3.2.

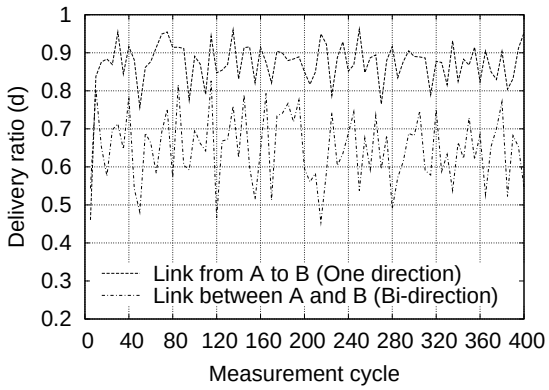


Figure 2.1: Measurement results with BAP for 4000s: BAP often under-estimates the quality of an asymmetric link between A and B due to its bi-directionality, even though one direction (from A to B) has good quality for data transmission.

Even though BAP might overcome this limitation using multiple types of probing, such an approach incurs additional overheads, and using broadcast may still under-/over-estimate link-quality as we will show in Section 2.5.3.2.

Next, unicast-based probing and passive monitoring are usually uni-directional in the sense that their measurement includes the delivery ratio of data and ACK frame transmissions. Thus, the measurement results accurately reflect the link-quality of actual data transmission. Again, in the first example, because uni-directional results are calculated by using the high-quality link direction and the reverse-direction quality of ACK transmissions, the results are accurate regardless of the opposite direction's quality, and allow node *A* to directly transmit packets to node *B* without taking a detour path.

Flexibility and feasibility Measurement techniques must be flexible enough to cope with time-varying link-quality. First, *aperiodic* measurements, which capture link-quality only for a certain period as in [47, 75], might be the simplest way to monitor link conditions. However, it yields poor measurement accuracy in wireless environments due to frequent link-quality fluctuations or requires significant efforts to determine the optimal measurement period.

On the other hand, the simple *on-demand* link-quality measurement used in MANETs [56, 83] might be cost-effective. However, it mainly focuses on link connectivity (i.e., a binary value) instead of actual wireless link quality. Even though several approaches

(e.g., [43]) have been proposed to elaborately measure link-quality using SNR, their main purpose is to maintain stable connectivity, rather than adapting to the link dynamics in real time.

Finally, the measurement techniques have to be easily implement-able and deployable in existing WMNs. BAP and unicast-based approaches can be implemented at any protocol layer without requiring any significant system change. Passive monitoring can be developed in the network and MAC layers. However, it needs to access and exploit the information from the MAC layer, which might not be available to the public [49].

2.3 The EAR Architecture

This section details the architecture of EAR. First, the design rationale and main algorithm of EAR are outlined. Next, we define the link quality that EAR deals with, and then describe its three measurement schemes. Finally, we analyze the complexity of EAR.

2.3.1 Overview of EAR

EAR is a low-overhead and high-accuracy measurement framework that is aware of asymmetric wireless links and also easily deployable in 802.11-based WMNs. EAR has the following distinct characteristics.

- *Hybrid approach*: EAR adaptively selects one of three measurement schemes (passive, cooperative, and active) to opportunistically exploit existing application traffic as probe packets. If there is no application traffic over a link, EAR uses active probing on the link at a reasonable cost. Otherwise, EAR switches itself to passive or cooperative monitoring that gratuitously uses existing traffic for collecting the link-quality information.
- *Unicast-based uni-directional measurement*: EAR uses *unicast* (instead of broadcast) in each direction of a link for measuring its quality. Unicast, which uses the same settings as the actual data transmissions, allows different schemes to generate homogeneous measurements. Moreover, since the quality of each link's direction is inde-

pendently measured via unicast, the measurement results are uni-directional.

- *Distributed and periodic measurement*: EAR independently measures the quality of link from a node to its every neighbor in a fully-distributed way. This measurement is also taken periodically to cope with the varying link-quality, and the measurement period is also adapted based on a link-quality history.
- *Cross-layer interaction*: EAR is composed of "inner EAR" (*i*EAR) that periodically collects and derives link-quality information in the network layer and "outer EAR" (*o*EAR) that monitors egress/cross traffic at the device driver. These two components interact across the two layers to intelligently exploit MAC-layer information without any modification of MAC's firmware.

EAR's overall operation can be described in four sequential steps as shown in Algorithm 1. First, during a measurement period (M_x), every node monitors link quality using one of passive, cooperative, and active measurement schemes per neighbor. Then, at the end of M_x, a node records the measured link quality and exchanges the information with neighboring nodes, if necessary. Next, during an update period (U_x), nodes process link-quality reports from their neighbors, if any. Finally, after an ordered pair of M_x and U_x (called the *measurement cycle*, C_x[3]), each node updates its local link-state table with directly and indirectly measured link-quality information, and then decides on its measurement scheme for the next cycle.

2.3.2 Link-Quality of Interest

EAR focuses on link cost and capacity as link-quality parameters, which are defined as follows. First, the link cost is defined as the inverse of the delivery ratio (d) of MAC frames. This definition reflects the expected transmission count of each data frame. Specifically, the cost ($\mathbb{C}$) of link A→B is calculated by

$$\mathbb{C} = \frac{1}{d_i} \text{ and } d_i = (1-\alpha) \times d_{i-1} + \alpha \times \frac{N_s}{N_t} \tag{2.1}$$

[3]We set C_x to 10 seconds (=9s (M_x) + 1s (U_x)) in our evaluation.

Algorithm 1 EAR at node i during C_x

(1) During a Measurement-Period, $t \in (C_{x-1}, M_x)$
  for every neighbor node j **do**
    $S_{ij} \leftarrow$ a monitoring scheme for the link from node i to node j
    if S_{ij} == PASSIVE or ACTIVE **then**
      monitor egress traffic to node j
    else if S_{ij} == COOPERATIVE **then**
      monitor egress traffic from node i to node k that node j overhears
    end if
    if node i received a cooperation request (ℓ) from node j **then**
      overhear cross traffic from node j to node ℓ
    end if
  end for

(2) At the end of a Measurement-Period, $t = M_x$
  for every neighbor j **do**
    record measurement results from node i to node j
    if node i received a cooperation request (ℓ) from node j **then**
      send node j a report of overhearing traffic from node j to node ℓ
    end if
  end for

(3) During an Update-Period, $t \in (M_x, M_x + U_x)$
  process a measurement report(s) from other nodes, if any

(4) End of an Update-Period, $t = M_x + U_x$ (or, $t = C_x$)
  for every neighbor j **do**
    calculate the quality of link from node i to j using Eq. (2.1)
    run the transition algorithm (in Figure 2.2) for node j
    if transition to COOPERATIVE **then**
      choose node k that node j can overhear
      send a cooperation request (k) to node j
    else if transition to ACTIVE **then**
      schedule active probe packets
    end if
  end for

where d_i is the smoothed delivery ratio, α a smoothing constant,[4] N_s the number of successful transmissions, and N_t the total number of transmissions and retransmissions during a measurement period of the i-th cycle.

EAR also measures link capacity by using the data rate obtained from MAC frame transmissions. The data rate can be an upper bound of capacity that the link can achieve, and is used to derive a net capacity along with link cost via such metrics as ETX [30] and ETT [36]. In EAR, the rate is derived based on the recent statistics of dominantly-used rate at the MAC layer during the previous measurement cycle. This is done jointly with the collection of the link cost (N_s, N_t). Upon completion of data transmission to its neighbor, EAR updates the frequency of the data rate used. At the end of the measurement cycle, EAR uses the frequency to infer the MAC's current data rate for the neighboring node. This simple algorithm enables EAR to work with any rate-control scheme (e.g., *fixed, auto*) in MAC and yields accurate link-capacity information without incurring any communication overhead.

Note that even though EAR can be easily extended to measure other parameters, such as delay and jitter, as described in Section 2.6.1, we will focus on the link cost and capacity as main link-quality parameters in the remainder of this chapter.

2.3.3 Hybrid Approach

As mentioned earlier, EAR consists of passive, cooperative and active measurement schemes, which are complimentary to each other. On the one hand, all of these schemes unicast probe packets through which any of the schemes provides consistent measurement results. On the other hand, although one scheme (i.e., active probing) provides accurate measurement results (e.g., 7% error in d as we will see in Section 2.4.2) compared to BAP (34%), the other schemes can further improve the accuracy (1.5%) by opportunistically exploiting a node's egress/cross traffic, if any.

Figure 2.2 depicts the EAR's hybrid measurement approach based on the three schemes. When a measuring node (m) has egress traffic, T_{egg}, to a neighbor node (n), m *passively*

[4]We set α to 0.3, but other values are also evaluated in Section 2.5.

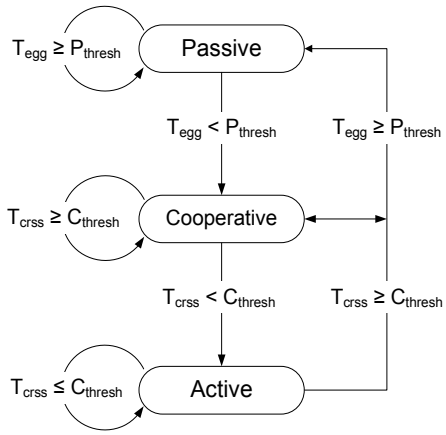


Figure 2.2: Three measurement schemes and their inter-transitions: EAR consists of passive, cooperative, and active measurement phases. Based on the amount of egress/cross traffic (T_{egg}, T_{crss}), EAR adaptively switches from one measurement scheme to another. P_{thresh} and C_{thresh} are the thresholds for passive and cooperative schemes, respectively.

monitors the traffic. When T_{egg} decreases below a certain threshold, P_{thresh}, m finds another neighbor node to which m has egress traffic and that n can overhear the traffic, and *cooperatively* (with node n) measures the quality of link $m \rightarrow n$. Finally, when the actual traffic over the link is low ($< C_{thresh}$), m *actively* measures link quality by unicasting probe packets over the link. Next, we give a detailed account of each measurement scheme with its rationale.

Passive measurement via egress traffic

When there is enough egress traffic, EAR favors passive monitoring over active monitoring for its accuracy and efficiency. The passive scheme (e.g., [65, 104]) can collect accurate and stable link-quality information from a large volume of existing data traffic without incurring any overhead. By contrast, many active schemes (using either broadcast or unicast probe packets as in [30, 34, 35]) must consume network resources for probing, yet cannot provide as accurate results as the passive scheme (that uses the actual traffic).

In a WMN, there is usually enough egress and relay traffic through each node. EAR employs the passive scheme to accurately measure link quality by capitalizing on this real traffic while minimizing the measurement overhead. There are, however, several design issues to be resolved before using the scheme as follows.

- *Heterogeneous packet sizes*: The packet size greatly affects the delivery ratio [3], and thus, a measurement scheme has to derive the ratio by using packets of same or similar size in order to obtain accurate and consistent link cost. EAR's passive scheme monitors packets within a 100-byte range of each of three popular sizes used in the Internet [112]—60, 512 and 1448 bytes—and derives the link cost corresponding to each size. EAR can also measure the link costs for other packet sizes similarly, or by using the estimation technique in [65].
- *Network-level vs. MAC-level*: Passive monitoring can be implemented at either the network layer or the MAC layer. The network layer solution is simple, but requires a neighboring node's feedback on each successful packet delivery. This consumes network bandwidth, and its result is oblivious of the retransmission results at the MAC layer. EAR eliminates this overhead by placing itself at a device driver and monitoring transmission results based on MAC's built-in ACK mechanism without additional cost or MAC modification (see Section 2.5.1).
- *Use of MAC information:* EAR obtains (and uses) MAC information via a device driver's interface to get around the difficulty of modifying MAC firmware. Proprietary MAC firmware makes it very difficult, if not impossible, for designers to modify MAC for direct use of channel information. Through a device driver's interface, EAR can access MAC management variables—`TxRetryLimitExceeded`, `TxSingleRetryFrames`, and `TxMultipleRetryFrames`[5]—to infer transmission results.

Suppose, as an example, that node *A* has (statistically) enough egress traffic to node *B*. Then, *A* requests its device driver to record the status of each of its packet transmissions. The device driver then keeps track of the three variables of MIB for the traffic, and derives the number of successful transmissions (N_s), the total number of transmissions (N_t), and the data rate. Next, at the end of a measurement period (M_i), EAR at the network layer obtains the measurement results from the device driver. Finally, at the end of an update period (U_i), it derives link quality using Eq. (2.1).

[5]Note that these variables are specified in IEEE 802.11 standard [50], and most of 802.11 chipsets, including Prism, Hermes and Atheros, provides interfaces to access these variables from a device driver or above [68, 87].

Cooperative measurement using cross traffic

EAR switches to cooperative monitoring when a measuring node (e.g., *B* in Figure 2.3) has no egress traffic to a neighbor node (*C*), but to others (*A*). We call the neighbor node with no traffic a "cooperative" node. Due to the broadcast nature of wireless media, the cooperative node (*C*) can overhear the traffic from the measuring node (*B*) to the other neighbors (*A*)—we call the traffic *cross traffic*. The overhearing result is then used for the measuring node to derive the quality of link $B \rightarrow C$. This scheme not only helps the measuring node avoid the active probing, but also improves the measurement accuracy by using a large amount of cross traffic. Note that all nodes in WMNs are assumed to faithfully cooperate. Preventing malicious behaviors, such as DoS attacks, is beyond the scope of this work.

To incorporate this scheme into EAR, we must resolve the following design issues.

- *Overhearing cross traffic*: The promiscuous mode in IEEE 802.11 NIC allows each node to overhear data frames destined for nodes other than itself. Due to the broadcast nature of wireless media, packets with the same network ID (or ESSID) can be captured by MAC and sent up to the upper layer. EAR at a device driver can choose this mode upon making/accepting a cooperation request, and monitor the cross traffic immediately.
- *Selective overhearing*: A cooperative node has to selectively overhear cross traffic whose data rate is the same as the rate from a measuring node to itself as if it were the destination of the traffic. Because the data rate affects greatly the delivery ratio as we will show in Section 2.5.3.2, overhearing all cross traffic with different rates yields inaccurate and noisy results. In EAR, the measuring node (*B*) selects, based on its local information, neighbor nodes (*A*) that the cooperative node (*C*) has to monitor, and then includes the selection in its cooperation request message (i.e.,`CooperateREQ`(A)) sent to the cooperative node.
- *Ambiguity of retransmissions*: Retransmissions cause both the measuring node and the cooperative node ambiguity in counting overheard packets. In Figure 2.3, because the cooperative node (*C*) cannot receive duplicate frames from its MAC layer even

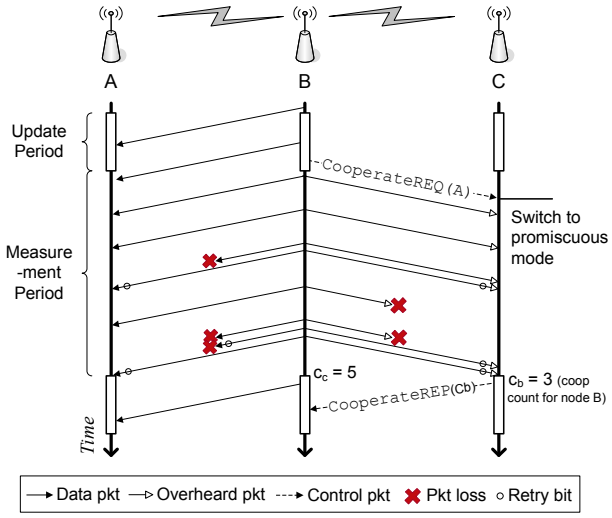


Figure 2.3: Example of node *B*'s cooperative monitoring with node *C*. Once node *B* requests cooperation with *C*, node *C* switches its NIC into a promiscuous mode and starts overhearing traffic from node *B* to node *A*. Then, it sends overheard results back to node *B*. Note that due to the ambiguity of retransmitted packets, the cooperative scheme only counts the overheard packets whose retry bit is not set.

in the promiscuous mode, the measuring node *B* cannot use the retransmitted packets for measurements (e.g., the fourth overheard packet). Also, if there are multiple retransmissions, the cooperative node cannot count the total number of packets that are successfully delivered to node *C*, due to a single retry bit in the frame and the ignorance of duplicate frames at MAC (e.g., the last overheard packet delivered to node *C*).

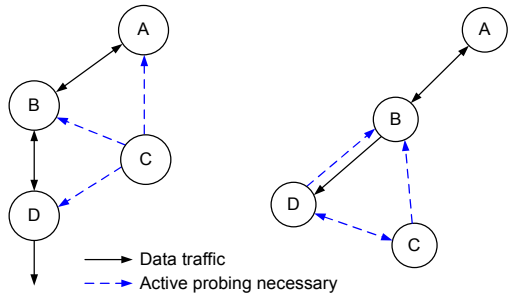
Let's consider the example in Figure 2.3. In the first update period, node *B* decides to use the cooperative scheme, based on the algorithm in Figure 2.2, to measure the quality of link *B*→*C* by using traffic *B*→*A*. Next, `CooperateREQ`(A) is sent to node *C*. On receiving the request, node *C* switches its NIC mode to the promiscuous mode, and starts to overhear the traffic from *B* to *A*. At the same time, node *B* also begins counting first-time successful transmissions (C_c) within the cross traffic. In the second update period, a report of overheard results (`CooperateREP`(C_b)) from *C* is sent to *B*, and then a new delivery ratio (i.e., $\frac{C_b}{C_c} = \frac{3}{5}$) is calculated.

Active measurement using shared unicast

When there is no egress/cross traffic, EAR switches to active monitoring and opportunistically sends unicast probe packets to neighbor nodes. Since it uses unicast-based probing, EAR can collect more accurate results than broadcast-based probing. On the other hand, by employing "cooperative" monitoring, EAR can reduce the active probing overhead to as low as BAP's overhead (e.g., 1 packet per second). Also, it can further reduce the probing overhead by adaptively adjusting the probe frequency based on the history of the link's quality.

To incorporate this scheme into EAR, one must address the following design issues.

- *Minimize the interference caused by probing traffic*: There are cases when a node needs to do active probing of link to one of its neighbors even though a channel is heavily used by others. For example, in Figure 2.4 (a), a channel is used by A, B and *D*, but *C* needs active probing of links to the other three, and (b) *D* has enough ingress traffic (e.g., video streaming), but it needs to probe links to *B* and *C*. EAR reduces the probing overhead by sharing the probe packets via cooperative monitoring. In Figure 2.4 (a), C probes only the link to A and also measures the quality of links to B, D through cooperation with *B* and *D*, which overhear the probing traffic from *C* to *A*. Note that this is different from BAP in the sense that probe packets are transmitted at the same rate as that of data transmissions (as opposed to a broadcasting rate).
- *Reduce the probing overhead on stable and idle links*: If a link has a small quality-variance and experiences low activities, EAR need not trigger active probes often. Thus, it uses an activity-based backoff timer that (i) is exponentially increased upon its expiration, with an upper bound ($window$), if the variance/activity has been below a minimum threshold, and (ii) linearly decreases every measurement cycle. On the other hand, if there has been either the minimum activity or quality-fluctuation, EAR resets the timer to 1 and triggers the active probing.
- *Need to probe at different rates*: A measuring node that uses several data rates to its neighbors cannot 'share' probe packets with all neighbors. Instead, the measuring node needs the same number of sets of probes as the number of data rates the node



(a) Isolated node case (b) Edge node case

Figure 2.4: Need for active monitoring: In Figure 2.4(a), C does not have any egress/cross traffic and thus, needs active probing while other nodes don't. In Figure 2.4(b), D has enough ingress traffic, but needs active probing of the opposite direction.

uses for its neighbors, which might, in turn, generate lots of probe packets during one measurement cycle. To reduce this possibility, EAR distributes a set of probing packets over several cycles during which it is not scheduled to probe links due to its backoff timer. Because links are idle under the active scheme and the backoff timer increases exponentially, there are usually enough unused cycles to accommodate all sets. If the number of sets is greater than the number of unused cycles, EAR schedules probes for all data rates in a round robin fashion over available cycles so that every rate has an equal chance to be probed.

Let's consider an illustrative example. Suppose node C in Figure 2.4(a) switches to active monitoring. Based on its link-quality variance and data-rate history, C classifies A and B to be in the 11 Mbps-group and D the 2 Mbps-group, respectively. Also, based on the backoff timer, C schedules the active probing to the 11 Mbps-group first. During the first update period, C broadcasts a cooperation request (`ActiveCooperateREQ(B)`) indicating B's cooperation. Then, for the following measurement period, C triggers the active probing to A and measures the quality of link $C \rightarrow A$ and $C \rightarrow B$ through passive and cooperative monitoring, respectively. In the second measurement period, C schedules the active probing to the 2 Mbps-group (i.e., D) based on the above scheduling rule. In the third update period, if the links $C \rightarrow A$ and $C \rightarrow B$ show stable quality and had no activity, EAR skips its probing for the 11 Mpbs-group and schedules the probing for the next group (i.e., D) if its backoff timer has been expired.

2.3.4 Complexity of EAR

The operation of EAR consumes less network resources than the broadcast-based approach due mainly to its use of hybrid monitoring. As the egress/cross traffic increases, EAR in each node passively monitors its traffic at the sender side, eliminating the need for transmitting probe packets. With cooperative monitoring, EAR only requires a periodic report message per cycle from a cooperating node to the measuring node. Since the cooperating node shares a *hello* message to send a report every cycle, its overhead is negligible. Finally, even in case of active monitoring, EAR's resource consumption is less than that of the broadcast approach due to its exponential active-timer, triggering active probing less frequently.

2.4 Performance Evaluation

We conducted extensive simulation to evaluate EAR. We first describe our simulation model and then present the simulation results in terms of accuracy, scalability and link-asymmetry-awareness.

2.4.1 The Simulation Model & Method

ns-2 [82] is used in our simulation study, and the simulation was run on topologies of Figure 2.5. First, T1 and T2 are used for evaluating the accuracy of both EAR and BAP. Second, T3 and T4 are used to measure the benefits of EAR's link-asymmetry-awareness. Finally, random topologies are used to evaluate EAR's scalability. Note that nodes in all topologies do not move (as in mesh networks), and two adjacent nodes are separated by 150–200 m.

In all simulation runs, we used the shadowing radio propagation model in the *ns-2* to simulate varying wireless link quality as suggested in [60] and adjusted the standard deviation of the model as a link-quality parameter. The standard deviation is based on the values in [82], and a wireless channel is modified so that each direction of the channel can be set to the different values to simulate asymmetric link-quality. CMU 802.11 wireless

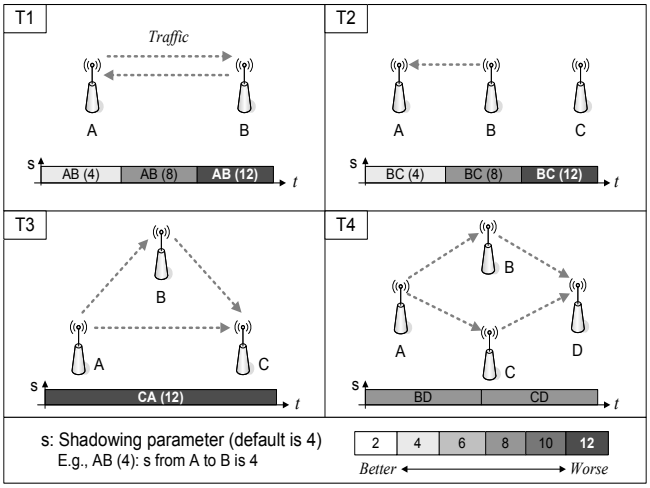


Figure 2.5: The simulation topologies: We used four topologies with different traffic and shadowing model values. T1 and T2 are used for evaluating the accuracy of EAR. T3 and T4 are used for EAR's asymmetry awareness and accurate node selection. The bottom of each topology shows the change of link quality in a time domain.

extensions in *ns-2* were used as the MAC protocol.

For close interaction with a routing protocol, we implemented EAR in both the network layer and the device driver as described in Section 2.3. We also implemented Dijkstra's algorithm for path selection with link-quality-aware routing metrics, including ETX [30] and ETT [36], and used the sequenced flooding mechanism [84] for disseminating the measured link quality. Note that dissemination of link-quality measurements is not within the scope of this work (development of efficient dissemination mechanisms for WMNs is part of our future work).

Throughout the simulation, the following parameter settings were used. First, RTS/CTS handshake at the MAC layer was disabled to study the effects of link-quality fluctuations and co-channel interferences. Second, UDP flows were mainly used to emulate users' traffic with an exponential distribution and a packet size of 1000 bytes. Third, a default MAC data rate was set to 11 Mbps. Finally, all experiments were run for 1000 seconds, and the results of 10 runs were averaged unless specified otherwise.

2.4.2 Accuracy

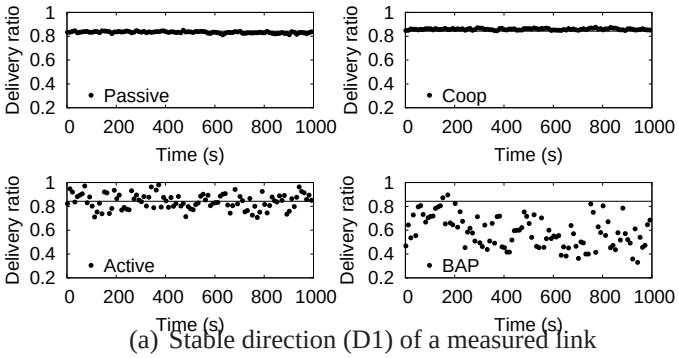
We show the accuracy of EAR with fluctuating and asymmetric link-quality and compare it with the accuracy of BAP.

2.4.2.1 EAR

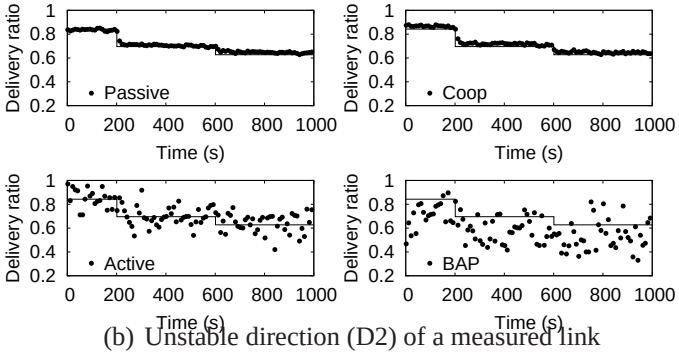
We evaluated the accuracy of each of EAR's measurement schemes while varying link-quality. To simulate time-variant asymmetric link-quality, we set the quality of a link's one direction (D1) to the default value, 4, of the shadowing model, while the opposite direction (D2)'s quality is set to 4 during [0s, 200s), to 8 during [200s, 600s), and to 12 during [600s, 1000s]. Given this scenario, we used T1 of Figure 2.5 to evaluate the passive scheme by measuring the delivery ratio, while running one UDP flow in both directions at 1.0 Mbps. We also used the above settings without UDP traffic for the active scheme. Finally, we used the topology T2 and only one UDP flow from *B* to *A* for the cooperative scheme; while changing the quality of link *B*→*C* to the same as D2, we measured the delivery ratio over the link between *B* and *C*.

Figure 2.6 shows the progression of the delivery ratio measured by EAR and BAP for stable (D1) and unstable (D2) directions of a link. First, EAR's passive and cooperative schemes show almost the same results as the ideal case as shown in two upper figures of Figures 2.6 (a) and 2.6 (b). Specifically, the root mean-square errors of the passive scheme's delivery ratio ($rmse_d$) are 0.012 for D1 and 0.015 for D2, and those for the cooperative scheme are 0.017 and 0.021. Moreover, they quickly adapt themselves to the change of link quality—*rmse* of the ratio's standard deviation ($rmse_s$) is 0.002 and 0.003 for the passive scheme, and 0.002 and 0.006 for the cooperative scheme, respectively—thanks to the use of a large portion of existing traffic as probe packets.

On the other hand, the accuracy of the active scheme lies between the previous two schemes' and BAP's accuracies. For example, for stable direction (D1), the active scheme increases $rmse_d$ (0.064) by a factor of 4 over the passive scheme, whereas BAP increases $rmse_d$ (0.287) by a factor of 26. Even though the active scheme increases the error rate due to a small number of probe packets, the error rate (7%) is much lower than BAP's (34%).



(a) Stable direction (D1) of a measured link



(b) Unstable direction (D2) of a measured link

Figure 2.6: Accuracies of EAR and BAP: Figure 2.6(a) shows that EAR yields accurate results close to the ideal values (solid lines), while BAP generates fluctuating and skewed results, affected by unstable direction. Figure 2.6(b) shows that EAR accurately measures the link quality even with unstable link states, whereas BAP shows inaccuracies and large variances.

Moreover, the active scheme successfully captures link-quality asymmetry (in contrast to BAP), as shown in two lower figures of Figures 2.6(a) and 2.6(b).

2.4.2.2 BAP

We also evaluated the accuracy of BAP for the purpose of comparison with EAR. We used topology T1 in Figure 2.5 with no traffic, and measured bi-directional link quality based on the link cost in Eq. (2.1). As shown in two BAP figures in Figure 2.6, BAP yields poor measurement accuracy (i.e., $rmse_d$ is 0.287 for D1 and 0.158 for D2), due mainly to the bi-directional nature of BAP. BAP's accuracy is 4 times worse than the active scheme's and 26 times worse than the passive scheme's. On the other hand, even though BAP is sensitive to varying link-quality (i.e., D2) and yields measurements relatively close to the ideal case, its variance is still (around twice) larger than the active scheme's, as shown in

Figure 2.6(b).

2.4.3 Scalability

We show the efficiency and scalability of EAR in terms of the number of network nodes, the number of flows, and traffic patterns.

2.4.3.1 Effects of the number of neighboring nodes

We evaluated the efficiency and scalability of EAR with a large number of neighboring nodes. To simulate a large and dense WMN, we varied the number of nodes from 2 to 96 in an area of 200 m $\times$ 200 m. We measured a node's average message rate during a measurement cycle, including the number of control and active probe packets. In this simulation, we did not transport any traffic to evaluate the EAR's worst-case overhead (i.e., the active scheme) and did compare it with BAP through which each node injects one probe packet (of 1448 bytes) per second.

Even in the worst case, EAR measurement overheads are, on average, only one half of BAP's, thanks to its activity/variance-based backoff timer. While maintaining a given measurement variance, EAR effectively avoids unnecessary probing of idle links. As shown by Figure 2.7, in case of low node density (1–10 nodes), EAR's overhead is a one-sixth (e.g., $window$=4, or "EAR-W4") of BAP's. Even though the timer expires easily (thus increasing the overhead) as the number of nodes increases (10–70 nodes), EAR also reduces the overheads by an average of 50% (e.g., $window$=16, or "EAR-W16") of BAP's, by adjusting the maximum window size of the timer. Even in a highly dense environment ($>$ 70 nodes), EAR's overhead does not surpass BAP's.

2.4.3.2 Effects of the number of flows/traffic patterns

We also evaluated the effects of the number of flows and traffic patterns on the measurement overhead. Measurements were taken on 32 nodes randomly distributed in an area of 1 Km $\times$ 1 Km. To show the effects of the number of flows, we randomly chose i pairs of nodes, where i is in $\{1, 2, \ldots, 15\}$, and ran one UDP flow at 300 Kbps for each pair.

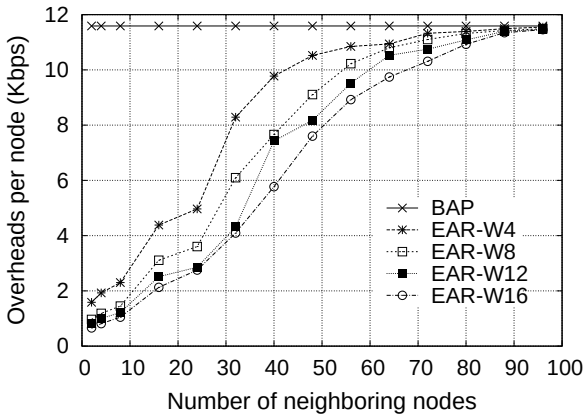


Figure 2.7: Effects of the number of neighboring nodes

Next, to show the effects of traffic patterns, we chose i pairs of nodes with random, sink-to-many, many-to-sink, and many-to-sink/sink-to-many traffic patterns, and ran one UDP flow for each pair. Here, *sink* and *many* are a central node and randomly-chosen nodes, respectively.

Even when the amount of egress/cross traffic increases, EAR reduces its measurement overhead by using the traffic as measurement packets. As shown in Figure 2.8, EAR's average overhead decreases by 27.8% with different numbers of UDP flows under the random traffic pattern. This savings mainly comes from EAR's hybrid approach which effectively exploits existing traffic (also see Figure 2.12).

The traffic pattern is also an important factor in the overall measurement overhead. In Figure 2.8, given the same amount of traffic, EAR under the random traffic pattern reduces the overhead most among all patterns due mainly to the increased chance of having a large number of relay nodes. It rarely has one central node, such as *sink* that transmits or relays most of the traffic. On the other hand, in the sink-to-many pattern, EAR reduces overheads by at most 9.7% because of the smaller number of relay nodes resulting from *sink* and the short average path length (i.e., 1.71) between *sink* and *many*. Finally, under the many-to-sink pattern, due to the increased number of nodes that transmit/relay traffic, resulting from *many*, EAR reduces the probing overhead by up to 33.4% as the number of flows increases.

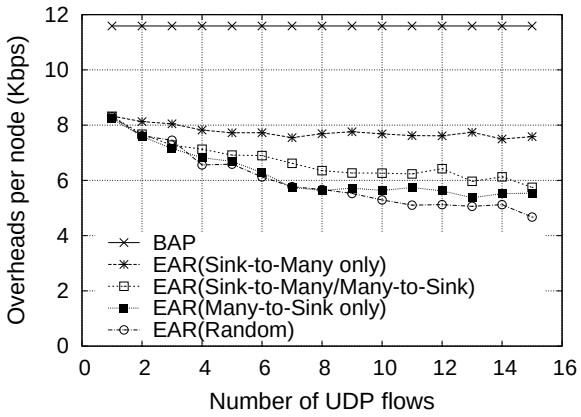


Figure 2.8: Effects of the number of flow/traffic patterns

2.4.4 Link-asymmetry-awareness

We now show two notable benefits—network efficiency and selection of a relay node—of EAR's accurate and direction-aware link-quality measurements.

2.4.4.1 Opportunistic use of asymmetric links

we evaluated how much asymmetry-awareness contributes to network efficiency. We used the topology T3 in Figure 2.5 and set the following parameters based on asymmetric links observed from our testbed (see Section 2.5.3.3). First, we set link quality between *A* and *B* and between *B* and *C* to a good condition (s=4) in *both* directions, but set the quality of link *A*→*C* to a good condition (s=4) and the quality of link *C*→*A* to a bad condition (s=12). In addition, we set data rates between *A* and *B*, and the link *A*→*C* to 5.5 Mbps, and the rate between nodes *B* and *C* to 11 Mbps to mimic asymmetric links. Next, we used two routing metrics, ETX and ETT, which use the measured link quality for making routing decisions. Finally, we ran one UDP flow on link *A*→*C* at 5.5 Mbps, and measured the flow's goodput (N_d) and the total number of transmissions (N_t) at the MAC layer.

Link-asymmetry-awareness, gained from EAR's direction-aware measurements, not only enhances end-to-end throughput, but also improves network efficiency by up to 49.1% even on a single asymmetric link. Table 2.1 shows the measured goodput (N_d), the total network capacity used (N_t) and network efficiency (defined as normalized goodput with respect to total packet transmissions, or N_d/N_t). First, ETX with EAR improves network

efficiency by up to 23.9% thanks to EAR's uni-directionality, while ETX with BAP often takes a detour around asymmetric links, as a result of BAP's bi-directionality. Even though BAP is not aware of link asymmetry, since ETX penalizes longer paths [36], ETX with BAP unintentionally uses link asymmetry more often than expected.

Table 2.1: Benefits of EAR's asymmetry-awareness: EAR improves network efficiency over BAP by up to 49.1%. ETX and ETT are used as routing metrics.

	(i) BAP	(ii) EAR	$\|(i) - (ii)\|$ (Benefits)
N_d *	333,553	370,450	46,897 (11.1%)
N_t **	482,362	432,936	49,426 (10.2%)
N_d/N_t	0.691	0.856	0.165 (**23.9 %**)

(a) Network efficiency with ETX

	(i) BAP	(ii) EAR	$\|(i) - (ii)\|$ (Benefits)
N_d	302,781	370,455	67,674 (22.3%)
N_t	527,835	432,929	94,906 (18.0%)
N_d/N_t	0.574	0.856	0.282 (**49.1%**)

(b) Network efficiency with ETT

Second, by considering the data rate in link quality, ETT with EAR effectively uses asymmetric links, and improves network efficiency over ETT with BAP by up to 49.1%. Since ETT and ETX with EAR constantly identify/use asymmetric links, their performance is the same as shown in the third column ((ii)s) of Table 2.1. By contrast, ETT with BAP shows a worse performance than ETX with BAP because ETT favors a path with the least sum of transmission time over the shortest-length path, often chosen by ETX. Due to BAP's underestimation of the delivery ratio on an asymmetric link, ETT with BAP overestimates the transmission time over the asymmetric link. Thus, ETT with BAP always takes a detour via *B* in the topology T3, consuming more network resources (e.g., BAP's $\frac{N_d}{N_t}$ is only 0.574 in Table 2.1 (b)).

2.4.4.2 Selection of the best relay node

We also evaluated how effectively EAR helps routing protocols [15, 30, 36] find the best relay nodes. We used the topology T4 in Figure 2.5, and ran one UDP flow from *A* to *D* at a maximum rate with two randomly-selected background UDP flows of 0.5 Mbps. In the first run, to simulate asymmetric links and varying link quality, we initially set the quality of

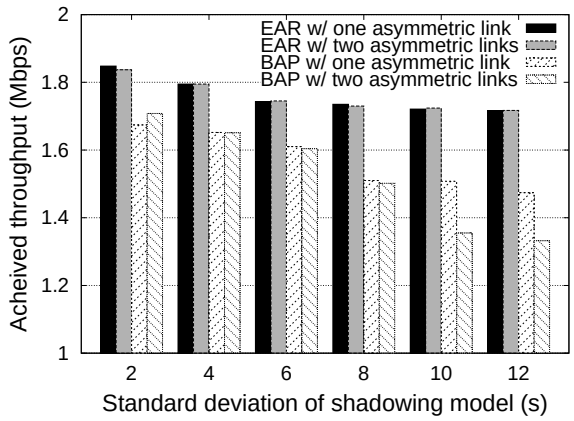


Figure 2.9: Effects of accurate node selection: EAR makes an about 30% improvement in achieved throughput via selection of the best relay node.

link B→D to a worse condition s (>4) for 500 seconds, while keeping the others in a better condition (s=4). Then, the worse-conditioned link became better (s=4), while the quality of link C→D became worse. In the second run, we also applied the same changes of s to link D→B to simulate another asymmetric link with the same traffic. Finally, we measured the goodput of the UDP flow while varying s and tracking the relay node selection by both EAR and BAP.

Using EAR's accurate and direction-aware measurements, routing protocols can improve the goodput by up to 28.9% through constantly finding the best relay nodes in the presence of varying link-quality and link-asymmetry. As shown in Figure 2.9, for all values of s, EAR achieves the goodput of the better-conditioned link. Specifically, in both runs, EAR improves the goodput over BAP by up to 18.6% (one asymmetric link) and 28.9% (two asymmetric links), respectively. EAR accurately selects the best relay nodes (i.e., node *C* for 0–500s and node *B* for 500–1000s in the topology T4), whereas BAP often chooses worse-relay nodes (e.g., node *B* for 54% of 0–500s period and node *C* for 45% of 500–1000s period, when s=12) mainly because of BAP's bi-directional link-quality measurement and large measurement variance.

2.5 System Implementation and Experimentation

We also implemented EAR in Linux-based systems and evaluated it on our experimental testbed. We first give the architectural details of this implementation, and then describe our experimentation setup. Finally, we present the experimental results.

2.5.1 Implementation Details

We implemented EAR in Linux-based systems with both Pentium-based devices (e.g., laptops) and StrongARM-based devices (e.g., Stargates and iPAQs) and Lucent IEEE 802.11b NIC.

*i*EAR at the network layer

As shown in Figure 2.10, *i*EAR is implemented in the network layer as a loadable module of netfilter [79] and is composed of the following six components. First, *task queue with timers* is responsible for releasing periodic EAR messages, such as cooperation request/reports, and triggering measurement/update events. Next, *message and task processor* processes the EAR messages and dispatches them to the corresponding task functions in *i*EAR. If necessary, it sends/receives periodic reports and requests to/from neighboring nodes.

When measurement timers expire, *measurement components* in the middle of Figure 2.10 take measurements and derive link states as follows. First, the measurement scheme selected by EAR records the measurement results obtained from the *o*EAR (stamper), and then exchanges the results with neighboring nodes during the update-period, if necessary (exchanger). Finally, it updates link states and determines which measurement scheme to use for the next measurement period (transitioner).

Link-state table and disseminator updates the local link-state table at the end of measurement cycle. Then, the updated information is periodically disseminated[6] to every other node through a sequenced flooding message and is reflected to other nodes' link-state table.

[6]We set a dissemination timer to 30s in our evaluation.

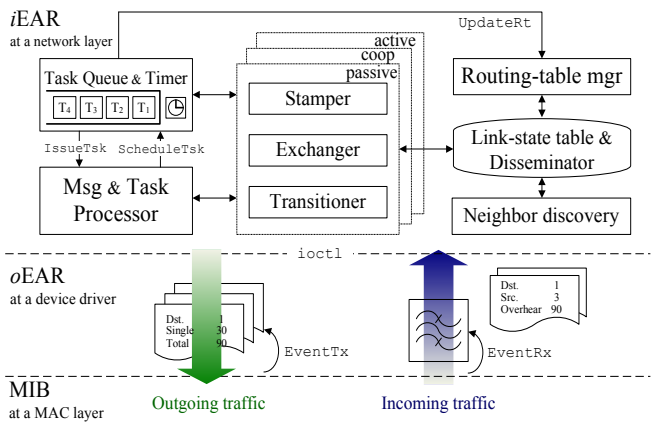


Figure 2.10: EAR's software architecture: EAR is composed of an *i*EAR at the network layer and an *o*EAR at a device driver.

Based on the update information, the *routing-table manager* locally calculates new routing paths with Dijkstra's algorithm and invokes a kernel function that updates the kernel routing table, if there are route changes. Finally, *neighbor discovery* maintains neighbors by exchanging periodic *hello* messages.

*o*EAR at a device driver

*o*EAR is implemented as sub-functions in an Orinoco 802.11 linux device driver, and is composed of two monitoring functions (i.e., outgoing and incoming traffic monitoring) and several interfaces with *i*EAR and MIB, as shown in Figure 2.10. First, *outgoing traffic monitoring* observes the egress traffic to each neighboring node and collects transmission statistics such as N_s, N_t and a data rate, based on MAC MIB information. Next, *incoming traffic monitoring* overhears cross traffic. When there is a cooperation request from *i*EAR, *o*EAR switches the mode of NIC into a promiscuous mode and begins overhearing the cross traffic between two neighbors. Finally, *o*EAR has several *interfaces* through which it requests transmission/reception results from the MAC layer (i.e., `EventTx, EventRx`) and periodically delivers collected statistics to *i*EAR (i.e., *ioctl*).

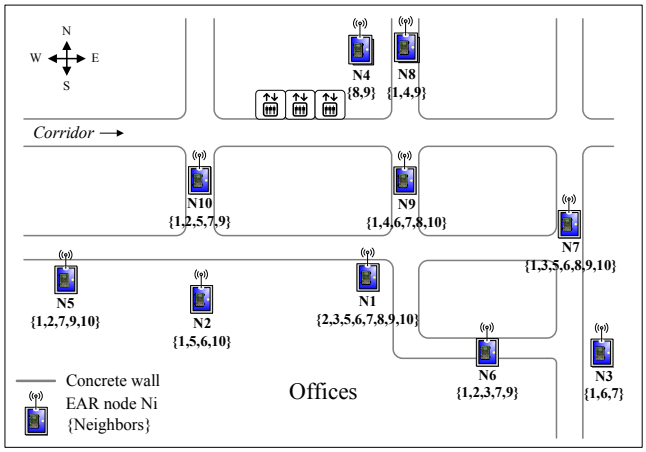


Figure 2.11: EAR testbed: 10 EAR nodes are placed on either ceiling panels or high-level shelves to send/receive strong signals in the same floor of our Department building (70 m $\times$ 50 m).

2.5.2 Experimental Setup

To evaluate our implementation, we constructed a testbed in the Electrical Engineering and Computer Science (EECS) Building at the University of Michigan. This building has rooms with floor-to-ceiling walls and solid wooden doors, and has relatively straight corridors. This environment provides enough multi-path effects from obstacles and interference from public wireless services.

In this environment, we deployed 10 nodes in the topology of Figure 2.11. We placed 5 laptops ($N1$–$N5$) in different offices and 5 stargates ($N6$–$N10$) along the corridors. All nodes were deliberately placed on either ceiling panels or high-level shelves to send/receive strong signals to/from neighbors.

All nodes were equipped with the same Lucent IEEE 802.11b PCMCIA card and were equipped with EAR. Each card operated at channel 11 (2.462 Ghz), less crowded channel in the building, and was set to use a built-in automatic rate control algorithm (i.e., *auto*) for its data rate. Next, each node dynamically loaded EAR into both the device driver (*o*EAR) and the network layer (*i*EAR). Finally, BAP was implemented and tested for the purpose of comparison.

2.5.3 Experimental Results

Using the above setup, we first show how effectively EAR uses real data traffic with its hybrid approach for measuring link quality. Then, we show that by using the data traffic, EAR's unicast-based approach measures link quality more accurately than the broadcast-based approach. Finally, we show that EAR's uni-directional link quality effectively identifies link asymmetry, and improves the efficiency of utilizing the channel capacity over BAP's bi-directional link quality.

2.5.3.1 Effective exploitation of data traffic

We evaluated the effects of EAR's hybrid approach by measuring the number of probe packets per link. We ran several different numbers (n_f) of UDP flows at 100 Kbps for 40 minutes, each pair of which were randomly chosen once every 4 minutes. While increasing n_f, we measured the average number of packets (n_p) used for measurement of each link's quality per cycle and derived the percentage of cycles during which each measurement scheme is used. Figure 2.12 plots representative links with different amounts of measurement packets.

While the broadcast-based approach uses a fixed number of probe packets (i.e., 10) per cycle, the hybrid approach in EAR indeed increases n_p as the number of flows increases. As shown in Figure 2.12, n_p of links with high egress traffic approaches 130, and n_p of links with high cross traffic grows up to 135 packets. On the other hand, n_p of links with low traffic is even smaller than BAP's since EAR reduces active probing based on an exponential backoff timer.

Next, the percentage of each measurement scheme per link depends on the link's geographical location and traffic pattern. In Figure 2.12, links with low traffic are located in edge nodes in our testbed such as N3, N4 and N5, whereas links with high egress traffic are located at center nodes such as N1, N2 and N9, where large flows are often relayed. On the other hand, links with high cross traffic can be placed at center nodes that have lots of relay traffic, but might not use some links to transmit the traffic often.

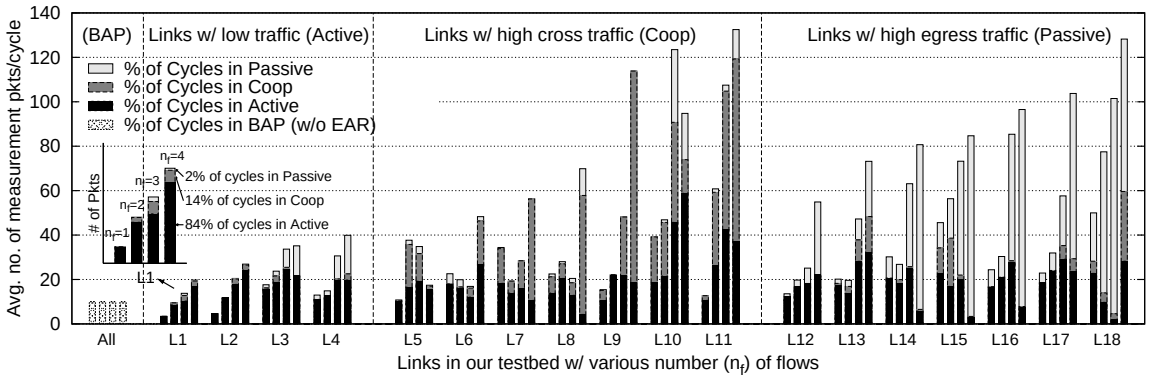


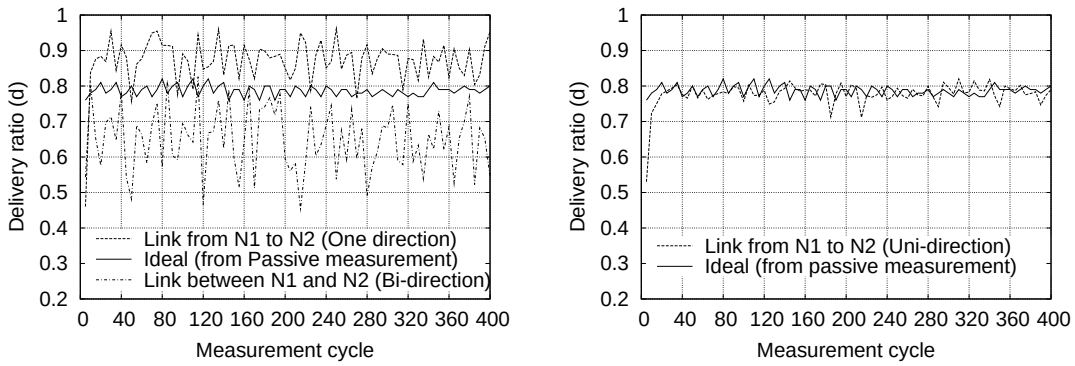
Figure 2.12: Benefits of EAR's hybrid approach: EAR effectively exploits existing traffic for their measurements through the hybrid approach. While increasing the number (n_f) of UDP flows in our testbed, we measured the average number of probe packets that are used for measuring the quality of each link, and the number of cycles (in percentage) used by each measurement scheme.

2.5.3.2 Improved accuracy with unicast packets

We also evaluated the accuracy improvement of unicast-based measurement in EAR over the broadcast-based measurement. We used two adjacent nodes (N1, N2) and measured the delivery ratio of link N1→N2 with both BAP and EAR's active probing for 400 cycles (i.e., 4000s). As a reference (called 'Ideal'), we separately ran one UDP flow at 1 Mbps from N1 to N2 and measured the delivery ratio by EAR's passive scheme. Note that the passive scheme provides accurate results as it derives link-quality information from the transmission of a large number of actual data packets.

Due to its low, fixed data rate, BAP yields less accurate results than the unicast-based approach. The top line in Figure 2.13(a) shows the progression of one direction quality of link N1→N2 with broadcast probing. Since actual data transmission uses 11 Mbps, BAP generates a higher delivery ratio than the ideal does due to its low data rate (i.e., 2 Mbps), which is more tolerant of bit errors. By contrast, owing to the use of unicast packets, EAR's measurement results (average is 0.778 (1.6% error), standard deviation 0.032) are closer to the ideal results (0.791, 0.014) than those (0.872 (10.2% error), 0.064) of BAP, as shown in Figure 2.13(b).

On the other hand, the bi-directional link-quality information derived from BAP (the bottom line in Figure 2.13(a)) provides worse results than the ideal case. This is due to



(a) Broadcast-based link-quality measurement (b) Unicast-based link-quality measurement

Figure 2.13: Broadcast vs. unicast measurement accuracy: Broadcast usually yields high link quality because it uses lower (thus reliable) data rate than that for actual data transmission (top curve in Figure 2.13(a)). By contrast, EAR's active probing provides almost the same results as ideal passive monitoring as shown in Figure 2.13(b).

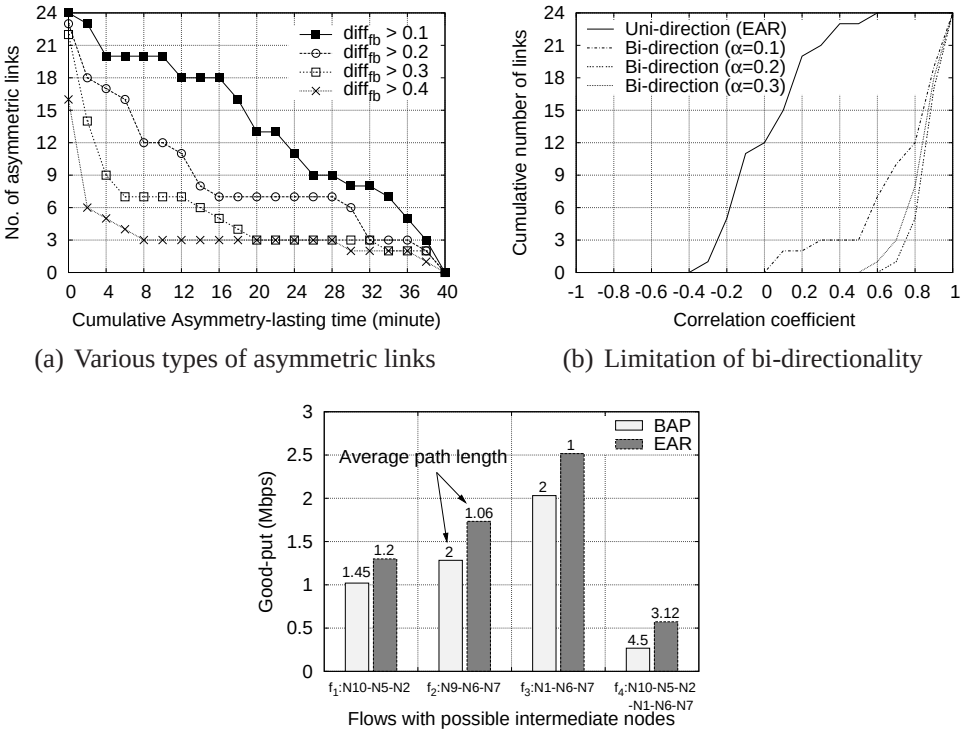
the poor quality of link N2→N1 and yields under-estimated quality of link N1→N2. This bi-directionality is evaluated in the following experiment.

2.5.3.3 Gains of uni-directionality on link asymmetry

Before showing the uni-directionality benefits on asymmetry, we first measured the asymmetry of wireless links in our testbed and evaluated the limitation of BAP's bi-directionality on the asymmetry. To this end, we repeated the experiment in Section 2.5.3.1. This time, we fixed n_f to three, and measured the delivery ratio of all links in each direction as well as bi-direction with BAP.

From extensive measurements, we found that wireless links often have significant link asymmetry and show various interesting characteristics, in terms of lifetime and degree of asymmetry. Figure 2.14(a) shows the number of links in our testbed that have different asymmetry lifetimes with different link quality in each direction. For the case of diff$_{fb}$>0.1 (i.e., $|d_{forward} - d_{backward}|$ >0.1), a small degree of asymmetry occurs very often for short (4 minutes) to long periods (40 minutes). On the other hand, some links experience a high degree of asymmetry (e.g., diff$_{fb}$>0.4) for more than 25 minutes of a 40-minute runtime.

Observing the various link-quality asymmetry, we found that bi-directional link quality measured by BAP is often affected by the worse-quality direction of an asymmetric



(a) Various types of asymmetric links

(b) Limitation of bi-directionality

(c) Benefits of uni-directionality

Figure 2.14: Benefits of uni-directionality's on link-asymmetry awareness: Wireless link quality is often asymmetric as shown in (a), and the good-quality direction of an asymmetric link is under-estimated since the bi-directional result is affected mainly by the poor-quality direction as shown in (b). By contrast, EAR's uni-directional link quality improves capacity efficiency as shown in (c)

link, thus yielding under-estimated results. To illustrate this, we derived the correlation coefficient (ρ) between bidirectional link quality and the quality of a worse direction link measured by BAP. As shown in the bi-direction cases of Figure 2.14(b), BAP generates skewed measurement results. More than 75% of links are closely related to poor asymmetric links ($\rho > 0.8$). By contrast, EAR's unidirectional link quality is independent of each other direction (Solid line in Figure 2.14(b)). More than 75% of links show weak correlation ($-0.2 < \rho < 0.2$).

Finally, we evaluated the improvement of EAR's uni-directional link quality on utilization of asymmetric links. We began with a simple case using three nodes (N2, N5 and N10) and one UDP flow from N10 to N2. Since the quality of link N2→N10's one direction is worse than the opposite direction, BAP's under-estimated bi-directional measurement makes the flow detour through N5. By contrast, EAR's uni-directional link quality enables

the flow to directly route to N2, improving the good-put by 27.45% as shown in Figure 2.14(c). Similarly, N9→N6→N7 and N1→N6→N7 have 35.2% and 12.87% good-put improvements, respectively.

We further evaluated how much uni-directionality improves the overall network performance. This evaluation is done with six nodes (N1, N2, N5, N6, N7, and N10), two asymmetric links (N2→N10, and N1→N7), and one UDP flow from N10 to N7. As shown in the f_4's result of Figure 2.14(c), EAR's asymmetry awareness improves the network efficiency over BAP by up to 114%, mainly by finding shorter paths (e.g., N10→N2→N1→N7) with asymmetric links than detouring paths (e.g., N10→N5→N2→N1→N6→N7).

2.6 Conclusion

We first discuss some of the remaining issues associated with EAR and then make concluding remarks.

2.6.1 Remaining Issues

Disseminating link-quality information: Although this chapter focused on how to measure link quality in WMNs, dissemination of the measured link-quality information is an equally important problem. Broadcast-based sequenced flooding [84] is one popular solution to this problem in small networks. There are also a couple of well-known approaches to the dissemination problem in MANETs [26, 105]. However, the information dissemination in WMNs has several challenges to overcome, including scalability and fault-tolerance. We will address these issues in a separate forthcoming work.

Measuring other link-quality parameters: In this work, the packet-delivery ratio and data rate—suitable for high-throughput metrics—are considered as the link-quality parameters. However, QoS parameters, such as delay and jitter, should be measured to support real-time applications. These parameters can be accurately measured by EAR, based on MIB [50] and NIC buffer clearing time [55]. Thus, along with the high-throughput parameters, EAR can support such applications as VoIP and IPTV that use the time-related parameters.

2.6.2 Concluding Remarks

In this chapter, we have presented a novel link-quality measurement framework, called EAR, for wireless mesh networks. EAR is composed of three complementary measurement techniques—passive, cooperative, and active monitoring—which minimize the probing overhead and provide highly accurate link-quality information by exploiting each node's egress and cross traffic. Moreover, based on accurate and direction-aware link-quality measurements, EAR identifies and exploits under-utilized asymmetric links, thus improving the utilization of network capacity by up to 114%. Finally, EAR is designed to be easily deployable in existing IEEE 802.11-based wireless mesh networks without any change of MAC firmware or system kernel compilation. EAR has been evaluated extensively via both *ns-2*-based simulation, and experimentation on a Linux-based implementation, demonstrating its superior accuracy and efficiency over existing measurement techniques.

CHAPTER 3

Self-Reconfigurable Multi-Radio Wireless Mesh Networks

3.1 Introduction

Wireless mesh networks (WMNs) have also been evolving in various forms (e.g., using multi-radio/-channel systems [11, 36, 61, 89]) to meet the increasing capacity demands by a variety of applications such as public safety, environment monitoring, city-wide wireless Internet services, and other emerging applications [7, 73, 74]. However, maintaining the performance of such WMNs is still a challenging problem, due mainly to heterogeneous and fluctuating wireless link conditions [3, 5, 123] in many cases. For example, other co-existing wireless networks may cause some links of a WMN to experience significant channel interference. Varying Quality-of-Service (QoS) demands from new mobile users can lead to mismatched network resource allocation. Spectrum etiquette or regulation can restrict access of some frequency channels in a certain area (e.g., hospital) or during a certain time period [71].

Even though real-time recovery from such wireless link failures is essential for maintaining network performance, previous approaches still suffer critical limitations as follows. First, existing network configuration algorithms [8, 17, 59] can provide (theoretical) guidelines for network planning. However, they may require "global" network configuration changes for every local link failure, thus incurring a very high management overhead or network disruption. Next, a *greedy* channel-assignment algorithm [91] can reduce the requirement of network changes for failure recovery, but one local greedy reconfiguration

could cause QoS degradation or another failures at neighboring nodes, triggering "propagation" of QoS failures (i.e., *ripple effect*). Finally, fault-tolerant routing protocols, such as local re-routing [77] or multi-path routing [26], can be adopted to avoid those side effects. However, their reliance on detour paths or redundant transmissions can require more network resources than network reconfiguration.

To overcome the above limitations, we propose a novel Localized sElf-reconfiGuration framewOrk (LEGO) that allows a multi-radio WMN (mr-WMN) to autonomously reconfigure its local network settings (e.g., channel and radio assignment) for real-time recovery from wireless link failures. In its core, LEGO includes the planning algorithm that generates a localized reconfiguration plan using a constraint-graph-based *top-down* approach. The top-down approach allows LEGO to minimize changes of healthy network settings, while allowing for local changes around the failure location. Specifically, LEGO first identifies network configuration changes available around a faulty area based on current channel and radio associations. Then, by imposing current network settings as constraints, the planning algorithm reduces the number of changes that each reconfiguration plan has to make.

Next, LEGO is also equipped with monitoring and reconfiguration protocols that essentially automate failure recovery in real time. The planning algorithm requires accurate link-quality information, and any reconfiguration plan generated has to be applied. Running in every mesh node, the monitoring protocol in LEGO periodically measures wireless link-conditions using efficient wireless probing techniques [57] and provide the condition information to the planning algorithm. In addition, using the information, LEGO periodically checks and detects link failures. Finally, upon detection of failures, LEGO triggers a distributed group formation algorithm that minimizes disruption during network reconfiguration through cooperation and synchronization among local mesh nodes.

LEGO is implemented and evaluated extensively via experimentation on our multi-radio WMN testbed as well as via *ns-2*-based simulation. Our evaluation results show that LEGO outperforms existing failure-recovery methods, such as static or greedy channel assignments, and local re-routing as follows. First, LEGO's planning algorithm effectively identifies reconfiguration plans that maximally satisfy the applications' QoS demands, ac-

commodating two times more flows than static assignment. Next, LEGO avoids the ripple effect in its planning via accurate bandwidth estimation (96% accuracy). Third, LEGO's on-line reconfiguration protocols improve network throughput and channel-efficiency by up to 26% and 92%, respectively, over the local re-routing scheme.

The rest of this chapter is organized as follows. Section 3.2 describes the motivation behind this work. Section 3.3 provides the design rationale and algorithms of LEGO. Section 3.4 describes the implementation and experimentation results on LEGO. Section 3.5 shows in-depth simulation results of LEGO. Section 3.6 discusses the remaining issues associated with LEGO and concludes this chapter.

3.2 Motivation

We first describe the need for self-reconfigurable multi-radio WMNs (mr-WMNs). Next, we discuss our network model and assumptions. Finally, we explain the limitations of existing approaches to achieving self-reconfigurability in mr-WMNs.

3.2.1 Why is Self-Reconfigurability Necessary?

Maintaining the health of WMNs' performance remains a challenging problem [88]. One of the fundamental reasons for this difficulty is the dynamic (or probabilistic) and distributed nature of wireless links, which, in turn, causes frequent and severe channel-related link failures. However, these failures, which are critical to the WMNs' performance, can be overcome by allowing an mr-WMN to autonomously reconfigure channel and radio[1] assignments, as in the following examples:

- *Recovering from link-quality degradation*: The quality of wireless links in WMNs can degrade (i.e., *link-quality failure*), due to severe interference from other co-located wireless networks [3, 60]. For example, Bluetooth, cordless phones, and co-existing wireless networks operating on the same or adjacent channels cause significant and varying degree of losses or collisions in packet transmissions, as shown in Figurere

[1]We interchangeably use "radio" and "interface" in this chapter.

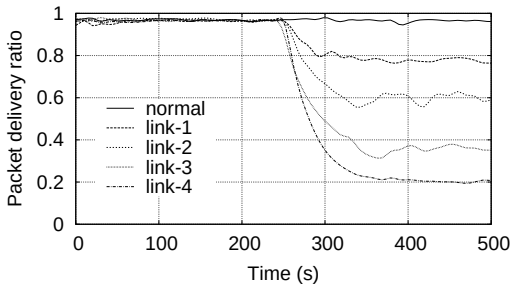


Figure 3.1: *Wireless link failures due to interference:* Given an interferer that uses adjacent channels from 250s, neighboring links experience different levels of degradations in packet-delivery ratio (i.e., link-quality failures). Note that the measurements are taken in our testbed using IEEE 802.11a NIC.

3.1. By switching the tuned channel of a link to other interference-free channels, local links can recover from such a link failure.

- *Satisfying varying QoS demands*: Links in some areas may not be able to accommodate increasing QoS demands from end users (*QoS failures*),[2] depending on spatial or temporal locality [48]. For example, links around a conference room may relay too much data/video traffic during the session. Likewise, relay links outside the room may fail to meet requests of attendees' Voice-over-IP calls during a session break. By re-associating their radios/channels with under-utilized radios/channels available nearby, links can avoid the failures.

- *Coping with heterogeneous channel availability*: Links in some areas may not be able to access certain wireless channels during a time period (*spectrum failures*), due to spectrum etiquette or regulation [18, 71]. For example, some links in a WMN need to vacate current channels if channels are being used for emergency response near the wireless links (e.g., hospital, public safety). Such links can seek and identify alternative channels available in that area.

Motivated by these three and other possible benefits of using reconfigurable mr-WMNs, in the remaining of this chapter, we would like to develop a system that allows mr-WMNs to autonomously change channel and radio assignments (i.e., *self-reconfigurable*) to recover from the above-mentioned channel-related link failures.

[2]We consider link bandwidth as a QoS parameter of interest.

3.2.2 Network Model and Assumptions

Multi-radio WMN: A network is assumed to consist of mesh nodes, IEEE 802.11-based wireless links, and one control gateway. Each mesh node is equipped with n radios, and each radio's channel and link assignments are initially given (e.g., see Figurere 3.2) through global channel/link assignment algorithms [8, 59, 89]. Multiple orthogonal channels are assumed to be available. For example, an IEEE 802.11a/b/g combo PCMCIA card can tune 16 orthogonal channels. The interference among multiple radios in one node is assumed to be negligible via physical separation among antennas or by using shields.

QoS support: During its operation, each mesh node periodically sends its local channel usage and the quality information for all outgoing links via management messages to the control gateway. Then, based on this information, the gateway controls admission of requests for voice or video flows. For admitted flows, the information on QoS requirements is delivered to the corresponding nodes for resource reservation through the RSVP protocol [16]. Next, the network runs routing protocols such as WCETT [36] or ETX [30] to determine the path of the admitted flows. This routing protocol is also assumed to include route discovery and recovery algorithms [56, 77, 83] that can be used for maintaining alternative paths even in the presence of link failures.

Link failures: Channel-related link failures that we focus on result mainly from narrowband channel failures. These failures are assumed to occur and last in the order of a few minutes to hours, and reconfiguration is triggered in the same order. For short-term (lasting for milliseconds) failures, fine-grained (e.g., packet-level or milliseconds) dynamic resource allocation might be sufficient [11, 61], and for a long-term (lasting for weeks or months) failures, network-wide planning algorithms [8, 17, 59] can be used. Note that hardware failures (e.g., node crash) or broadband-channel failures (e.g., jamming) are beyond the scope of this work.

3.2.3 Limitations of Existing Approaches

Given the above system models, we discuss the pros and cons of using existing approaches to self-reconfigurable WMNs.

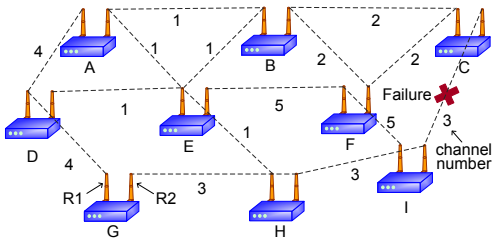


Figure 3.2: *Multi-radio WMN*: A WMN has an initial assignment of frequency channels as shown above. The network often experiences wireless link failure and needs to reconfigure its settings.

Localized reconfiguration: Network reconfiguration needs a planning algorithm that keeps network changes necessary for link failure recovery as local as possible, as opposed to changing the entire network settings. Existing channel assignment and scheduling algorithms [8, 17, 59] provide holistic guidelines such as throughput bounds and schedulability for channel assignment during a network deployment stage. However, the algorithms do not consider the degree of configuration changes from previous network settings, and hence they often require *global* network changes to meet all the constraints, akin to edge coloring problems [44]. Even though these algorithms are suitable for static or periodic network planning, they may cause network service disruption and thus are unsuitable for dynamic network reconfiguration that has to deal with local link failures.

Next, the *greedy* channel-assignment algorithm, which considers only local areas in channel assignments (e.g., [91]), might do better in reducing the scope of network changes than the above-mentioned assignment algorithms. However, this approach still suffers from the ripple effect, in which one local change triggers the change of additional network settings at neighboring nodes (e.g., nodes using channel 3 in Figure 3.2), due to association dependency among neighboring radios. This undesired effect might be avoided by transforming a mesh topology into a tree topology, but this transformation reduces network connectivity as well as path diversity among mesh nodes.

Finally, interference-aware channel-assignment algorithms [89, 108] can minimize interference by assigning orthogonal channels as closely as possible geographically. While this approach can improve overall network capacity by using additional channels, the algorithm could further improve its flexibility by considering both radio diversity (i.e., link

association) and local traffic information. For example, in Figure 3.2, if channel 5 is lightly-loaded in a faulty area, the second radio of node *C* can re-associate itself with the first radio of node *I*, avoiding configuration changes of other links.

QoS-awareness: Reconfiguration has to satisfy QoS constraints on each link as much as possible. First, given each link's bandwidth constraints, existing channel-assignment and scheduling algorithms [8, 59, 89] can provide approximately optimal network configurations. However, as pointed out earlier, these algorithms may require global network configuration changes from changing local QoS demands, thus causing network disruptions. We need instead a reconfiguration algorithm that incurs only local changes while maximizing the chance of meeting the QoS demands. For example, if link EH in Figure 3.2 experiences a QoS failure on channel 1, then one simple reconfiguration plan would be to re-associate R1 of node H to R2 of node E in channel 5, which has enough bandwidth.

Next, the greedy algorithm might be able to satisfy particular links' QoS demands by replacing a faulty channel with a new channel. However, neighboring links, whose channel has been changed due to ripple effects (e.g., links GH and HI in Figure 3.2), may fail to meet QoS demands if the links in the new channel experience interference from other co-existing networks that operate in the same channel.

Cross-layer interaction: Network reconfiguration has to jointly consider network settings across multiple layers. In the network layer, fault-tolerant routing protocols, such as local re-routing [77] or multi-path routing [26], allow for flow reconfiguration to meet the QoS constraints by exploiting path diversity. However, they consume more network resources than link reconfiguration, because of their reliance on detour paths or redundant transmissions. On the other hand, channel and link assignments across the network and link layers can avoid the overhead of detouring, but they have to take interference into account to avoid additional QoS failures of neighboring nodes.

3.3 The LEGO Design

This section details LEGO. We first present its design rationale and overall algorithm. Then, we detail LEGO's reconfiguration algorithms. Finally, we discuss the complexity of

LEGO.

3.3.1 Overview

LEGO is a distributed system that is easily deployable over IEEE802.11-based mr-WMNs. Running in every mesh node, LEGO supports self-reconfigurability via the following distinct features:

- *Localized planning*: Based on multiple channels and radio associations available, LEGO generates reconfiguration plans that allow for changes of network configurations only in the vicinity of link failures, while retaining configurations in areas remote from failure locations.
- *QoS-aware planning*: LEGO effectively identifies QoS-aware and cost-effective reconfiguration plans by (i) estimating the QoS-satisfiability of generated reconfiguration plans and (ii) deriving their expected benefits in channel utilization.
- *Link-quality monitoring and fault detection*: LEGO accurately monitors the quality[3] of links of each node in a distributed manner. Furthermore, based on the measurements and given links' QoS constraints, LEGO detects local link failures in real time.
- *Cross-layer planning*: LEGO actively interacts across the network and link layers for planning. This interaction enables LEGO to exploit a routing protocol for maintaining connectivity during recovery period and to include a re-routing for reconfiguration planning.

Algorithm 2 describes the operation of LEGO. First, LEGO in every mesh node monitors the quality of its outgoing wireless links at every t_m (e.g., 10 sec) and reports the results to a gateway via a management message. Second, once it detects a link failure(s), LEGO in the detector node(s) triggers the formation of a group among local mesh routers that use a faulty channel, and one of the group members is elected as a leader using the well-known bully algorithm for coordinating the reconfiguration. Third, the leader node sends a planning-request message to a gateway. Then, the gateway synchronizes the planning

[3]For example, the quality parameters include packet-delivery ratio or data-transmission rate as will be explained in Section 3.3.2.

Algorithm 2 LEGO Operation at mesh node i

(1) Monitoring period (t_m)
1: **for every** link j **do**
2: measure link-quality (lq) using passive monitoring;
3: **end for**
4: send monitoring results to a gateway g;

(2) Failure detection and group formation period (t_f)
5: **if** link l violates link requirements r **then**
6: request a group formation on channel c of link l;
7: **end if**
8: participate in a leader election if a request is received;

(3) Planning period (M, t_p)
9: **if** node i is elected as a leader **then**
10: send a planning request message (c, M) to a gateway;
11: **else if** node i is a gateway **then**
12: synchronize requests from reconfiguration groups M_n
13: generate a reconfiguration plan (p) for M_i;
14: send a reconfiguration plan p to a leader of M_i;
15: **end if**

(4) Reconfiguration period (p, t_r)
16: **if** p includes changes of node i **then**
17: apply the changes to links at t;
18: **end if**
19: relay p to neighboring members, if any

requests—if there are multiples requests—and generates a reconfiguration plan for the first request. Fourth, the gateway sends a reconfiguration plan to the leader node and the group members. Finally, all nodes in the group execute the corresponding configuration changes, if any, and resolve the group. We assume that during the formation and reconfiguration, all messages are reliability delivered via a routing protocol and per-hop timer.

3.3.2 Top-Down Approach

LEGO generates reconfiguration plans by using channel and radio diversities in a *top-down* manner. Here, a reconfiguration plan is defined as a set of links' configuration changes necessary for a network to recover from a link(s) failure on a channel and to satisfy the QoS requirements, and there can be multiple reconfiguration plans for each failure recovery. Existing channel assignment and scheduling algorithms [8, 59, 89] consider QoS constraints on all links in its initial planning, and thus need to search a large configuration

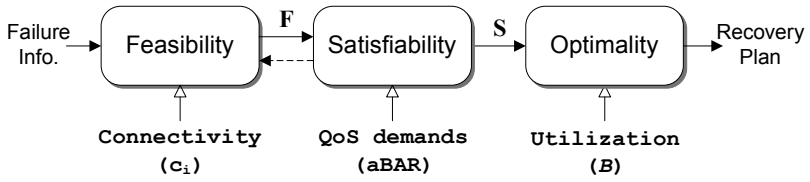


Figure 3.3: *Top-down approach in LEGO's planning*

space for an optimal solution. This approach often becomes an NP-complete problem [89], and moreover, one requirement change might lead to completely different network configurations. By contrast, as depicted in Figurere 3.3, LEGO first considers loose constraints (e.g., connectivity) in its planning to reduce search space and generates a *set* of feasible reconfiguration plans. A feasible plan is defined as a reconfiguration plan whose changes can handle link failures but are not proved to meet the QoS requirements. Then, within the set, LEGO applies strict constraints (e.g, bandwidth) to choose a reconfiguration plan that satisfies the QoS requirements and that improves channel utilization most.

Feasible plan generation: Generating feasible plans is essentially to search all legitimate changes in links' configuration and combinations thereof, while minimizing configuration changes of mesh routers around a faulty area. Thus, the plan generation algorithm in LEGO finds a set of configuration changes that maximally maintain original links' connectivity and that avoid the use of a faulty channel for the failed links (e.g., nodes using channel 3 in Figure 3.2).

To find such a set of configuration changes for the reconfiguration group, the planning algorithms follow a two-step procedure, which consists of (1) enumeration of per-link changes and (2) combination of the enumerations. Specifically, in Step 1, the algorithms generate legitimate link changes given current network configuration as well as available channel/radio diversities. For each (faulty or non-faulty) link that uses a faulty channel in the group, the algorithms apply primitive link changes, which is defined in Table 3.1, assuming that neighboring links' configurations do not change. The generated link changes are then examined by the following constraints for both avoiding the use of a faulty channel and improving network utilization:

(c_1) $l_n^i > 0$ for every radio i in node n;

Table 3.1: *Definition of link-change in LEGO*. Each change represents a primitive link change in channel, association, or route. Multiple changes can be jointly used to represent changes of multiple links.

Primitive changes	Description
Channel switch $(S(A_i, B_j)_{\alpha\to\beta})$	Radios A_i and B_j of link AB switch their channel (α) to other channel (β).
Radio switch $(R(A_i, B_j)_{\alpha\to\beta})$	Radio A_i in node *A* re-associates with radio B_j in node *B*, tuned in channel (β).
Detouring $(D(A_i, B_j))$	Both radios A_i and B_j of link *AB* remove their associations and use a detour path.

(c_2) $c_n^i \neq c_n^j$ for all different radios in node n; and

(c_3) $c_n^i \neq f_c$ for radio i with a faulty link.

where l_n^i is the number of links associated with the radio i in node n, c_n^i the channel of radio i in node n, and f_c the faulty channel. c_1 implies that all radios of a node must have at least one association to improve radio resource utilization. Next, c_2 enforces that each radio of a node has to use a non-interfering channel with other radios. Finally, c_3 requires new configurations to use the non-faulty channel[4] if a node has a faulty link. Let us consider an example illustrated in Figure 3.4. As shown in the figure (gray columns), Step 1 generates a set of possible changes per link while assuming other links' configurations do not change, and then excludes changes that violate the above constraints. For example, D(I,H) may or may not violate c_3 depending on neighboring changes but is valid in current network configurations. On the other hand, D(C,I) obviously causes one radio of node C to violate c_3.

Next, in Step 2, the algorithm integrates all possible combinations between per-link changes of adjacent changes' sets, as shown in Figure 3.4 (white columns). These combinations are basically to connect valid neighboring per-link changes that maintain network connectivity by imposing the following constraints:

(c_4) $l_n^i > 0$ for all radios that have changes; and

(c_5) $c_n^i = c_n^{i'}$ for a joint node n of adjacent links.

where i and i' are channels assigned by two adjacent links' changes. c_4 prevents waste of

[4]We will also consider the case where only a small number of non-faulty channels are available in Section 3.6.1.

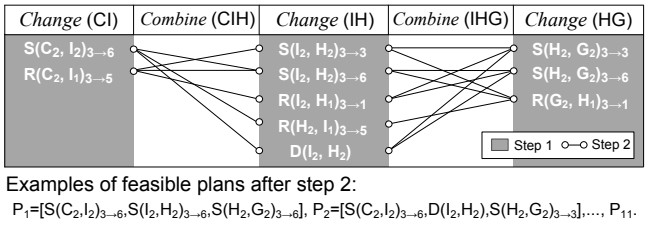


Figure 3.4: *Example of network planning*: LEGO generates per-link changes (blue columns) and then connects them for feasible reconfiguration plans (white columns) for recovery of the failure in Figure 3.2.

a radio. c_5 ensures that adjacent links have at least one common channel. For example, adjacent links' plans $S(C,I)_{3\rightarrow6}$ and $S(H,I)_{3\rightarrow3}$ in Figure 3.4 cannot be connected because each plan requires the same radio of node *I* to set up different channels. Now, the planning algorithm has 11 feasible reconfiguration plans ($\mathbb{F}$) by traversing connected changes of all links considered in the planning, as illustrated in Figure 3.4.

QoS-satisfiability evaluation: Among a set of feasible plans $\mathbb{F}$, the planning algorithms now need to identify QoS-satisfying reconfiguration plans by checking if the QoS constraints are met. Although each feasible plan ensures that a faulty link(s) will use non-faulty channels and maintain its connectivity, the plan might not satisfy the QoS constraints or even cause cascaded QoS failures on neighboring links. To filter out such plans, LEGO checks the following two requirements: (i) the links reconfigured by a feasible plan have to meet their bandwidth requirements; and (ii) reconfigured links must not cause the violation of neighboring links' QoS requirements.

To check these constraints, LEGO needs to accurately estimate each link's capacity and its available channel air-time. In multi-hop wireless networks equipped with CSMA-like MAC, each link's achievable bandwidth (or throughput) can be affected by both link capacity and other links' activities that share the channel air-time. Even though numerous bandwidth estimation techniques have been proposed, they mostly focus on the average bandwidth of each node in a network [30, 117] or the end-to-end throughput of flows [26]. By contrast, LEGO first estimates an individual link's capacity (C), based on measured or cached link-quality information—packet-delivery ratio and data-transmission rate measured by passively monitoring the transmissions of data or probing packets—and the formula derived in Appendix A.

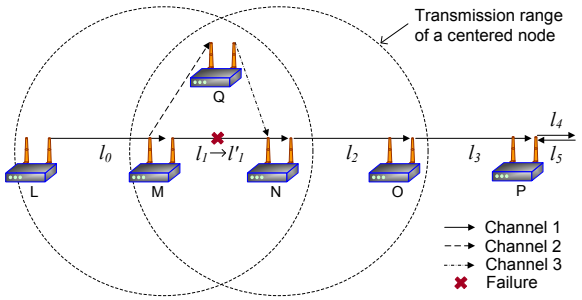


Figure 3.5: *Busy Air-time Ratio (BAR) of a directed link*: BAR (e.g., l_1) is affected by activities of neighboring links (l_0, l_2,and l_3) in channel 1 and is used to evaluate QoS satisfiability of a link.

Having the estimated capacity (C) of each link, LEGO defines and uses the expected busy air-time ratio of each link to check link's QoS satisfiability. Assuming that a link's bandwidth requirement (q) is given, the link's busy air-time ratio (BAR) can be defined as BAR = $\frac{q}{C}$ and must not exceed 1.0 (i.e., BAR $<$ 1.0) for a link to satisfy its bandwidth requirement. In addition, if multiple links share the air-time of one channel, LEGO determines QoS satisfiability by calculating aggregate BAR (*aBAR*) of end-radios of a link as follows:

$$aBAR(k) = \sum_{l \in L(k)} \frac{q_l}{C_l} \tag{3.1}$$

where k is a radio ID, l a link associated with radio k, $L(k)$ the set of directed links within and across radio k's transmission range. Fig. 3.5 illustrates the usage of *aBAR* for evaluating a feasible plan. Assuming BAR of each directed link (l_i) is 0.2 (e.g., $\frac{2Mbps}{10Mbps}$) in a tuned channel, the *aBAR* of each radio tuned in channel 1 does not exceed 1.0, satisfying each link's QoS requirement or the constraint (i).

Second, to identify the ripple effect from a plan (constraint (ii)), LEGO also estimates the QoS-satisfiability of links one-hop-away from member nodes whose links' settings will be changed by the plan. If these one-hop-away links are still QoS-satisfiable, the effects of the changes do not propagate thanks to spatial reuse of channel. Otherwise, the effects of local changes will propagate, causing cascaded QoS failures. For example, assuming that

BAR(l_1) increases from 0.2 to 0.4 (l_1') in Fig. 3.5. To accommodate this increase, reconfiguration plans that have a detour path through node *Q* do not affect the QoS-satisfiability of the neighboring nodes. On the other hand, plans with radio switches (e.g., R(L_2, M_1)$_{1\rightarrow 2}$) satisfy the QoS of link MN but cause *aBAR(O_{R2})* to exceed 1.0, resulting in cascaded QoS failures of links beyond node O.

Choosing the best plan: LEGO now has a set of reconfiguration plans that are QoS-satisfiable without causing the ripple effect, and needs to choose the best plan within the set. For this selection, the planning algorithms in LEGO define and use a benefit function $B(p)$ that quantifies the improvement of channel utilization a reconfiguration plan p can make.

The benefit function is defined as $B(p) = \frac{1}{n}\sum_{k=1}^{n}\beta(k)$, where $\beta(k)$ is the relative improvement in the air-time usage of radio k. This definition allows the benefit function to identify a reconfiguration plan that improves the overall air-time usage most among multiple plans based on the following quantitative improvement index $\beta(k)$:

$$\beta(k) = \begin{cases} \epsilon_1(k) - \epsilon_2(k) & \text{if } \epsilon_1(k), \epsilon_2(k) > \delta \\ \epsilon_2(k) - \epsilon_1(k) & \text{if } \epsilon_1(k), \epsilon_2(k) < \delta \\ \epsilon_1(k) + \epsilon_2(k) - 2\delta & \text{if } \epsilon_1(k) > \delta > \epsilon_2(k) \\ 2\delta - \epsilon_1(k) - \epsilon_2(k) & \text{if } \epsilon_2(k) > \delta > \epsilon_1(k) \end{cases} \tag{3.2}$$

where $\epsilon_1(k)$ and $\epsilon_2(k)$ are estimated *aBAR*s of a radio k in existing configurations and in new configurations, respectively, and δ the desired channel utilization.[5] Eq. (3.2) implies that if a reconfiguration plan makes overall links' channel utilization closer to the desired utilization δ, then $\beta(k)$ gives a positive value, while giving a negative value otherwise. Suppose δ is 0.5 as shown in Fig. 3.6; LEGO favors a plan that reconfigures links to have 50% available channel air-time (e.g., plan 1 in the figure). If a plan reconfigures a WMN to make the links heavily utilized while idling others (e.g., plan 2), then the benefit function consid-

[5]Note that δ is a system parameter set by network administrators, depending on the application scenarios of a WMN.

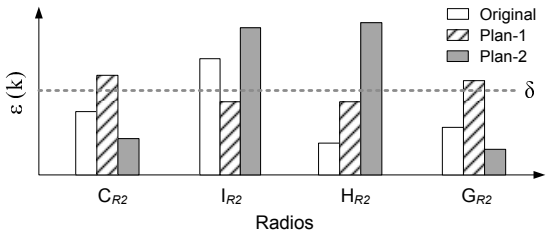


Figure 3.6: *Benefit function*: B prefers a reconfiguration plan that improves overall channel utilization close to the desired parameter δ.

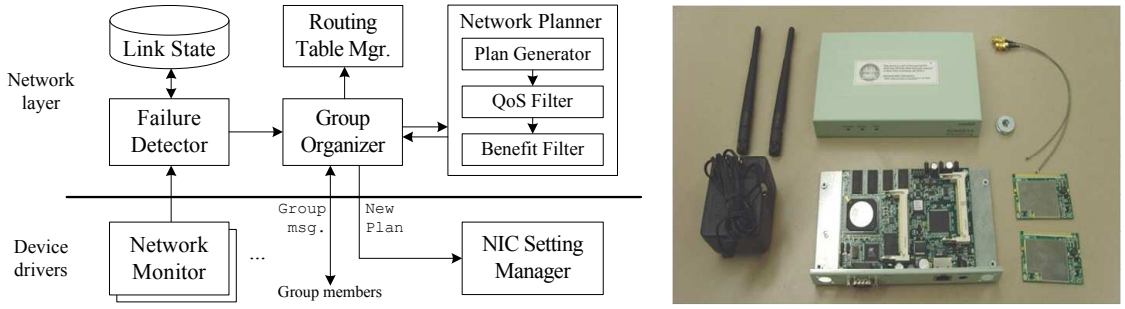
ers the plan ineffective, placing the plan in a lowly-ranked position. The effectiveness of β and δ will be discussed further and evaluated in Section 3.4.3.5.

3.3.3 Complexity of LEGO

Thanks to its distributed and localized design, LEGO incurs reasonable bandwidth and computation overheads, compared to the centralized counterpart as follows. First, the network monitoring part in the reconfiguration protocols is made highly efficient by exploiting existing data traffic and consumes less than 12 Kbps probing bandwidth (i.e., 1 packet per second) for each radio. In addition, the group formation requires only $O(n)$ message overhead (in forming a spanning tree), where n is the number of nodes in the group. Next, the computational overhead in LEGO mainly stems from the planning algorithms. Specifically, each node is responsible for generating its possible link plans, which incurs $O(n+m)$ complexity, where n is the number of available channels and m the number of radios. Next, a gateway node needs to generate and evaluate feasible plans, which incurs search overhead in a constraint graph that consists of $O(l(n+m))$ nodes, where l is the number of links that use a faulty channel in the group.

3.4 System Implementation

We have implemented LEGO in a Linux OS and evaluated it in our testbed. We first explain the implementation details, and then present important experimental results on LEGO.



(a) LEGO software architecture in each node (b) LEGO hardware prototype

Figure 3.7: *LEGO's implementation and prototype*: (a) LEGO is implemented across network and link layers as a loadable module of Linux 2.6 kernel. (b) LEGO software is then installed in Soekris wireless routers and evaluated extensively in our multi-radio WMN testbed.

3.4.1 Implementation Details

Figure 3.7(a) shows the software architecture of LEGO. First, LEGO in the network layer is implemented using netfilter [79], which provides LEGO with a hook to capture and send LEGO-related packets such as group-formation messages. In addition, this module includes several important algorithms and protocols of LEGO: (i) *network planner*, which generates reconfiguration plans; (ii) *group organizer*, which forms a local group among mesh routers; (iii) *failure detector*, which periodically interacts with a network monitor in a device driver and maintains an up-to-date link-state table; and (iv) *routing table manager*, through which LEGO obtains or updates states of a system routing table.

Next, LEGO components in a device driver are implemented in an open source MADWiFi device driver [68]. This driver is designed for Atheros chipset-based 802.11 NICs [9] and allows for accessing various control and management registers (e.g., `longretry`, `txrate`) in the MAC layer, making network monitoring accurate. The module in this driver includes (i) *network monitor*, which efficiently monitors link-quality and is extensible to support as many multiple radios as possible; and (ii) *NIC manager*, which effectively reconfigures NIC's settings based on a reconfiguration plan from the group organizer.

3.4.2 Experimental Setup

To evaluate our implementation, we constructed a multi-hop wireless mesh network testbed on the fourth floor of the Computer Science and Engineering (CSE) building at the University of Michigan. The testbed consists of 17 mesh nodes and has multiple (up to 5) links. Each node is deliberately placed on either ceiling panels or high-level shelves to send/receive strong signals to/from neighboring nodes. On the other hand, each node will experience enough multi-path fading effects from obstacles and interference from co-existing public wireless networks.

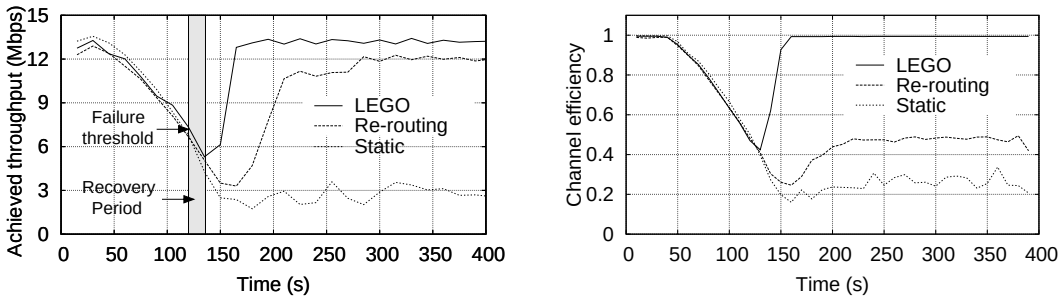
As shown in Figure 3.7(b), each mesh node is a small-size wireless router—Soekris board 4826-50 [106] (Pentium-III 266 Mhz CPU, 128 MB memory). This router is equipped with two EMP IEEE 802.11 a/b/g miniPCI cards and 5-dBi gain indoor omni-directional antennae. Each card operates at IEEE 802.11a frequency with a pseudo ad-hoc mode, and is set to use fixed data rate and transmission power. Next, all nodes run the Linux OS (kernel-2.6), a MADWiFi device driver (version 0.9.2) for wireless interfaces, and the LEGO implementation. In addition, ETX [30] and WCETT [36] routing metrics are implemented for routing protocols. Finally, the Iperf measurement tool [51] is used for measuring end-to-end throughput, and the numbers are derived by averaging the experimental results of 10 runs, unless otherwise specified.

3.4.3 Experimental Results

We evaluated improvements achieved by LEGO's planning algorithms, including throughput, channel efficiency, QoS satisfiability, and reduction of ripple effects.

3.4.3.1 Throughput gains

We first stuided throughput gains via LEGO's real-time reconfiguration. We run one UDP flow at a maximum rate over a randomly-chosen link in our testbed, while increasing the level of interference every 10 seconds. We also set the QoS requirement of every link to 6 Mbps, and measure the flow's throughput progression every 10 seconds during a 400-second run. For the purpose of comparison, we also ran the same scenario under



(a) Throughput gains (b) Channel-efficiency gains

Figure 3.8: *Gains in throughput and channel efficiency:* LEGO effectively reconfigures the network around a faulty link, improving both network throughput and channel-efficiency by up to 26% and 92%, respectively. By contrast, local re-routing causes degradation in channel-efficiency due to the use of a detour path, and static channel-assignment does not react to faults in a timely manner.

the local re-routing with a WCETT (Weighted Cumulative Expected Transmission Time) metric [37], and static channel-assignment algorithms. Note that we do not intentionally run a greedy algorithm in this single-hop scenario, because its effect is subsumed by LEGO. We, however, compare it with LEGO in multi-hop scenarios in Section 3.4.3.4.

Figure 3.8(a) compares the progression of link throughput achieved by the above three methods. LEGO effectively reconfigures the network on detection of a failure, achieving 450% and 25.6% more bandwidth than static-assignment and local re-routing, respectively. LEGO accurately detects a link's QoS-failure using link-quality monitoring information, and completes network reconfiguration (i.e., channel switching) within 15 seconds on average, while the static-assignment experiences severe throughput degradation. Note that the 15-second delay is due to one measure-cycle (10 seconds) plus the 5-second reconfiguration delay. This delay mainly stems from a back-off timer (0.5 second per link) for exchanging the group formation messages and can be further reduced. On the other hand, the local re-routing improves the throughput by using a detour path, but still suffers from throughput degradation because of an increased loss rate along its detour path.

3.4.3.2 Channel-efficiency gains

LEGO also improves channel efficiency (i.e., the ratio of the number of successfully-delivered data packets to the number of total MAC frame transmissions) by more than 90%

over the other recovery methods. Using the data collected during the previous experiment, we derive channel efficiency of the UDP flow by counting the number of total MAC frame transmissions and the number of successful transmissions.

As shown in Figure 3.8(b), LEGO improves channel efficiency by up to 91.5% over the local re-routing scheme, thanks to its on-line channel reconfiguration. On the other hand, using static-channel assignment suffers poor channel utilization due to frame retransmissions on the faulty channel. Similarly, the local re-routing often makes traffic routed over longer or low link-quality paths, thus consuming more channel resources than LEGO.

3.4.3.3 QoS-satisfaction gain

LEGO increases chance to meet varying QoS demands. To show this gain, we first assign links and channels in our testbed as shown in Figure 3.2. Here, nodes G, A, and C are a gateway, a mesh router in a conference room, and a mesh router in an office. We assume that mobile clients are in the conference room and request video streams through the router A during a meeting, and after the meeting, they return to the office room and connect to the router C. While increasing the number of video streams, we measure the total number of admitted streams after network reconfiguration. We use static, WCETT routing metric that finds a path with diverse channels and LEGO for reconfiguration.

LEGO's QoS-aware reconfiguration planning improves the chance for a WMN to meet the varying QoS demands, on average, by 200%. As shown in Figure 3.9, a static channel-assignment algorithm cannot support more bandwidth than the initial assignment (e.g., 9.2 Mbps from C to G). In addition, using the WCETT metric helps find a path that has channel diversity (e.g., WC_1 in Figure 3.9 favors the path C→F→I→H→G, and WC_2 favors the path C→F →E→H→G), but consumes more channel resource. On the other hand, LEGO effectively discovers and uses idle channels through reconfiguration, thus satisfying QoS demands by up to three times more than static-assignment algorithms (e.g., the bandwidth from C to G in Figure 3.9).

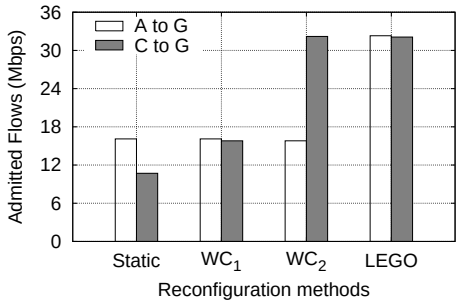


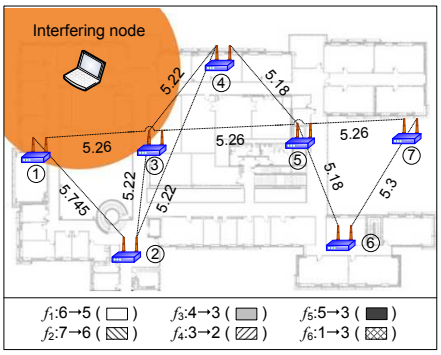
Figure 3.9: *QoS-satisfaction gain*: LEGO reconfigures networks to accommodate varying QoS demands in time and space.

3.4.3.4 Avoidance of ripple effects

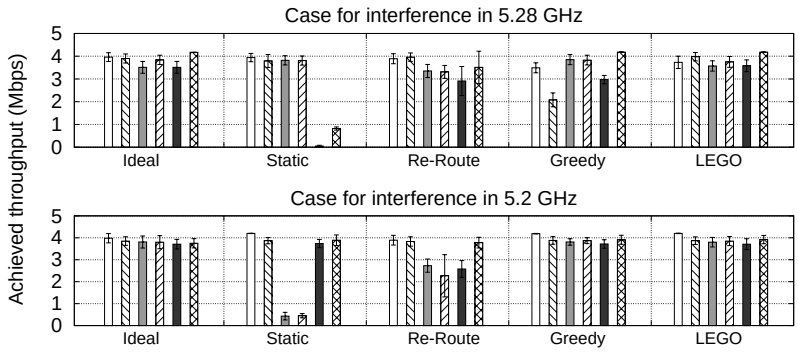
We also studied LEGO's effectiveness in avoiding the ripple effects of network reconfiguration. Figure 3.10(a) shows initial channel and flow assignments in a part of our testbed. In this topology, we run 6 UDP flows ($f_1, \cdots, f_6$) each at 4Mbps, and measure each flow's throughput while injecting interference into a target channel. We run same scenarios with 2 different interference frequencies (5.28 and 5.2 Ghz) to induce failures on different links. In addition, we use four failure-recovery methods (i.e., static, local re-routing, greedy, and LEGO) for comparison.

Since LEGO considers the effects of local changes on neighboring nodes in its planning via *aBAR*, it effectively identifies reconfiguration plans that avoid the ripple effects. Figure 3.10(b) shows the average throughput of each flow after network reconfiguration with each of the four recovery schemes. First, with interference with 5.28 Ghz, nodes 1, 3 and 5 experience link-quality degradation. Via LEGO's reconfiguration, each flow achieves an average 98% of the ideal throughput, whereas the flows via local re-routing achieves 82% of the throughput because of the use of detour paths (f_5:5→4→3, f_6:1→2→3). On the other hand, while partially recovering from the original link failures, the greedy approach causes throughput degradation of neighboring links. This is because one local greedy channel-switching (from 5.26 to 5.32 Ghz) requires the neighboring links' channel (e.g., between nodes 5 and 7) to change, creating interference to other neighboring nodes' link (e.g., between nodes 6 and 7) that use adjacent channels (e.g., 5.3 Ghz).

Next, in the second interference case (5.2 Ghz), LEGO also identifies QoS-satisfying reconfiguration plans, achieving 97% throughput of the ideal (the lower figure in Fig-



(a) Channel and flow assignment



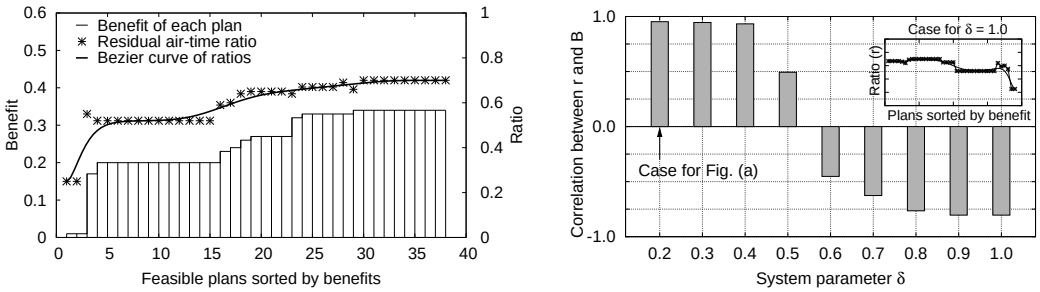
(b) Achieved throughput of f_1-f_6 through each recovery mechanism

Figure 3.10: *LEGO's avoidance of ripple effects*: LEGO finds a local reconfiguration plan that avoids the ripple effects by considering neighboring nodes' channel utilization, whereas the greedy channel-switching (the upper figure in Figure 3.10(b)) and local re-routing (the lower figure in Figure 3.10(b)) often cause throughput degradation of neighboring nodes.

ure 3.10(b)). On detection of an interference, LEGO switches the channel on the "problem" links according to its planning algorithm. Naturally, this result (configuration and throughput) is the same as that achieved by the greedy method. On the other hand, the local re-routing causes heavy channel contention for detour paths, degrading neighboring flows' performance (i.e., f_5) as well as others' (f_3, f_4).

3.4.3.5 Effectiveness of the benefit function

Finally, we studied the effectiveness of LEGO's benefit function B in finding the most beneficial plan among QoS-satisfying plans. Furthermore, we evaluated the effect of the system parameter δ in the benefit function. First, we analyzed the output from the benefit function on QoS-satisfying plans (38 plans) for the previous link-failure experiment (5.28



(a) Output of the benefit function ($\delta = 0.2$) (b) Effects of δ in the benefit function

Figure 3.11: *Effectiveness of the benefit function in LEGO:* (a) the benefit function effectively ranks plans in the order of desired channel utilization (at δ=0.2—prefer the plan with the highest r), and (b) δ is an effective parameter for the benefit function to rank the plans.

Ghz case). Here, we set δ to 0.2 for LEGO to favor a plan that makes more bandwidth available via reconfiguration. Then, we calculated the per-radio residual air-time ratio ($r = 1.0 - aBAR$) of the network reconfigured under each plan. Note that the benefit is a relative merit among a given set of plans, so we focus on the ordering of plans, not their benefit numbers.

Figure 3.11(a) shows the calculated benefits (bars) of the 38 plans sorted in an increasing order along with the corresponding r (dots). As shown in the figure, the benefit function can accurately rank the reconfiguration plans in such a way that the plans providing higher r values are ranked higher—which is controlled by the assigned system parameter δ=0.2. Note that a Bezier curve in the upper figure shows the trend of r in B.

Next, to study the effects of various δ values, we calculate the correlation coefficient (ρ) between r and the benefit of the selected plan, while increasing δ from 0.2 to 1.0. As shown in Figure 3.11(b), δ effectively reflects the desired channel utilization in ranking the QoS-satisfying plans. If δ is low, the benefit function gives priority to plans with high residual air-times (e.g., $\rho = 0.95$ when $\delta = 0.2$). By contrast, a high δ value causes LEGO to favor plans with high channel utilization ($\rho = -0.84$ when δ =1.0), showing the effectiveness of δ.

3.5 Performance Evaluation

We have also evaluated LEGO via in-depth simulation. We first describe our simulation methodology, and then present the evaluation results on LEGO.

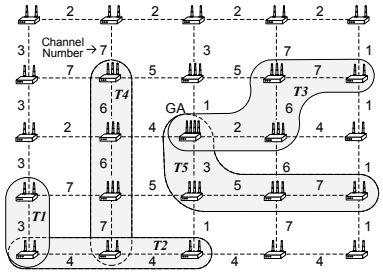
3.5.1 The Simulation Model & Methods

We used ns-2 [82] in our simulation study. Throughout the simulation, we use a grid topology with 25 nodes in an area of 1Km$\times$1Km. In the topology, adjacent nodes are separated by 180 m and each node is equipped with a different number of radios depending on its location. A gateway is equipped with four radios, one-hop-away nodes from a gateway have three radios, and other nodes use two radios.

For each node in these topologies, we use the following network protocol stacks. First, the shadowing propagation model [94] is used to simulate varying channel quality and multi-path effects. Next, CMU 802.11 wireless extension is used for a MAC protocol with a fixed data rate (i.e., 11 Mbps) and is further modified to handle multiple radios and multiple channels. Finally, a link-state routing protocol, a modification of DSDV [84], and multi-radio-aware routing metric (WCETT [36]) are used for routing.

Given these settings, LEGO is implemented as an agent in both the MAC layer and a routing protocol as explained in Section 3.3. It periodically collects channel information from MAC, and requests channel switching or link-association changes based on its decision. At the same time, it informs the routing protocol of network failures or a routing table update.

There are several settings to emulate real-network activities. First, to generate users' traffic, one UDP traffic with either a gateway or a randomly-chosen mesh nodes is used. Each flow runs at 500 Kbps with a packet size of 1000 bytes. Second, to create network failures, uniformly-distributed channel faults are injected at a random time point. Random bit-error is used to emulate channel-related link failures and lasts for a given time period. Finally, all experiments are run for 3000 seconds, and the results of 10 runs are averaged unless specified otherwise.



(a) Scenarios for diverse QoS requests

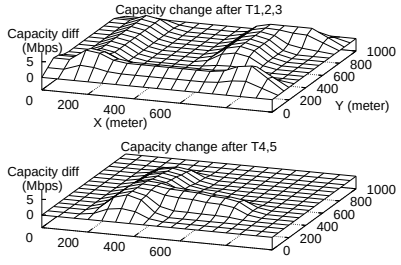
Table-1: $\delta = 0.8$

Time	T1		T2		T3		T4		T5	
	(i)	(ii)	(i)	(ii)	(i)	(ii)	(i)	(ii)	(i)	(ii)
Admission	*no*	*yes*	*no*	*yes*	*no*	*yes*	*no*	*yes*	*no*	*yes*
Available capacity(Mb/s)	0.5	3.5	0.5	2.0	1.0	3.0	3.0	7.5	0.5	1.5

Table-2: $\delta = 0.4$

Time	T1		T2		T3		T4		T5	
	(i)	(ii)	(i)	(ii)	(i)	(ii)	(i)	(ii)	(i)	(ii)
Admission	*no*	*yes*	*no*	*yes*	*no*	*yes*	*no*	*yes*	*no*	*yes*
Available capacity(Mb/s)	0.5	5.5	0.5	5.5	1.0	3.0	0.5	1.5	0.5	1.5

(b) Reconfiguration results



(c) Snapshots of changes

Figure 3.12: *Satisfying varying QoS constraints*: Figure 3.12(a) shows requests with different QoS requirements. Next, Figure 3.12(b) explain improved (or changed) network capability (i) before and (ii) after reconfiguration. Finally, Figure 3.12(c) depicts results of network reconfiguration given the scenarios.

3.5.2 Evaluation Results

3.5.2.1 Effectiveness of the QoS-aware planning

We measured the effectiveness of LEGO in meeting varying QoS requirements, especially for a large-scale multi-radio wireless network. We use a grid topology that initially is assigned to symmetric link capacity as shown in Figure 3.12(a). while changing QoS constraints in gray areas at different times (i.e., $T1, ..., T5$), we evaluate the improvement in available capacity LEGO can make.

As shown in the tables of Figure 3.12(b), LEGO reconfigures a wireless mesh network to meet different QoS requirements. Before each QoS changes, the gray areas can only accept 1 to 9 UDP flows. On the other hand, after reconfiguration, the network in the areas can admit 4 to 15 additional flows, improving the average network capacity by 3.5 times. Figure 3.12(c) also confirms the benefits from LEGO's reconfiguration given different requests. For each request, LEGO increases local capacity through reconfigurations.

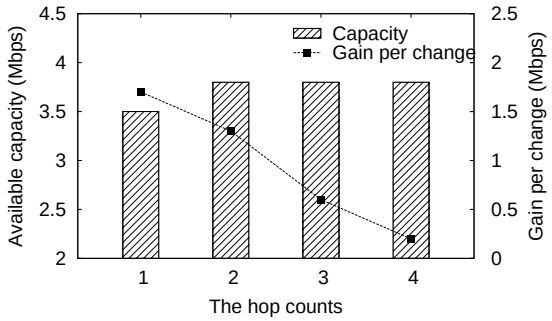


Figure 3.13: *The impact of reconfiguration range*: The hop length can help LEGO search exhaustive reconfiguration plans. However, the benefit from the increased length is small, whereas the number of total changes for the reconfiguration increases.

3.5.2.2 Impact of the reconfiguration range

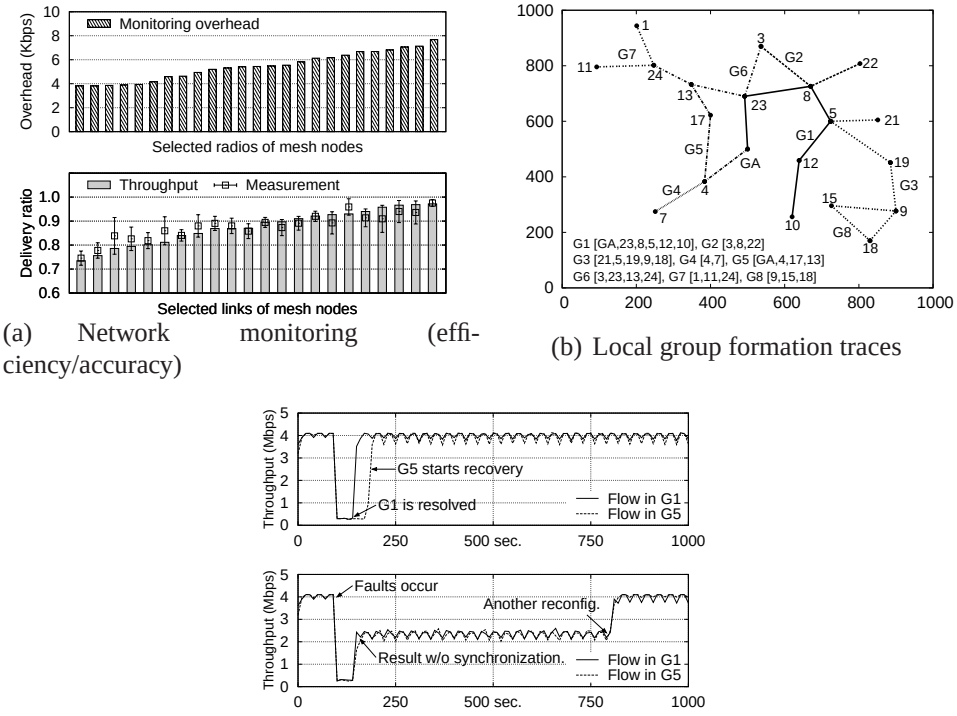
We evaluated the impact of the reconfiguration range. We use the same experiment settings as the previous one and focus on reconfiguration requests at $T1$. As we increase the hop count from each group member, we measure the capacity improvement achieved by the reconfiguration plans. In addition, we calculate the improvement per change as the cost-effectiveness of reconfiguration planning with different hop counts.

Figure 3.13 plots the available capacity after reconfiguration over different hop counts. As shown in the figure, LEGO can improve the available bandwidth by increasing the reconfiguration range. However, its improvement becomes marginal as the range increases, because of the limited number of radios around the faulty link. Furthermore, because reconfiguration plans with a larger range are required to incur more changes in network settings, the bandwidth gain per change degrades significantly.

3.5.2.3 Efficiency in link-condition monitoring

We evaluated how efficiently and accurately a monitoring protocol in LEGO measures network state information. We measure the active probing overhead of LEGO. At the same time, we measure a packet-delivery ratio and compare it with actual link throughput to evaluate its probing accuracy.

As shown in Figure 3.14(a), LEGO efficiently monitors the network state, while providing highly accurate link-quality information. The monitoring overhead (the upper figure)



(a) Network monitoring (efficiency/accuracy)

(b) Local group formation traces

(c) Effects of synchronization (with vs. without)

Figure 3.14: *Effectiveness of reconfiguration protocols*: LEGO efficiently and accurately monitors the network state (Figure 3.14(a)). Next, it locally forms a reconfiguration group in a distributed way (Figure 3.14(b)). Finally, LEGO's synchronization at a gateway allows the network to recover from multiple channel faults (Figure 3.14(c)).

of each radio in a node ranges from 3.8 Kbps to 7.6 Kbps—0.07% of 11 Mbps—thanks to adaptive measurement based on link history [57]. Next, the thus-measured information is close to the actual link throughput (the lower figure) with an average of only 3.8% error in the delivery ratio.

3.5.2.4 Effectiveness of reconfiguration under multiple link failures

We also evaluate the benefits of LEGO's local group formation, especially in the presence of multiple channel faults. We randomly move nodes in the grid topology of Figure 3.12(a) to generate a random topology, and introduce multiple channel faults (i.e., $F1, \ldots, F8$). We trace the result of LEGO's group formation and identify the benefit from the use of the synchronization at the gateway.

Figure 3.12(b) shows different local reconfiguration groups for different channel faults. As shown in the figure, LEGO locally forms the group and reconfigures the network. Moreover, thanks to the synchronization mechanism in the gateway, LEGO can effectively mask the adverse effects of the race condition on network information. For example, if G1 and G5 are formed simultaneously, both might might reconfigure the channel of their problematic links to the same channel, degrading network throughput as shown in the lower figure of Figure 3.12(c). By contrast, LEGO can synchronize multiple groups with a reasonable delay (10 sec.), improving end-to-end throughput by 92% (the upper figure).

3.6 Conclusion

We first discuss some of the remaining issues associated with LEGO and then make concluding remarks.

3.6.1 Remaining Issues

Joint optimization with flow assignment and routing: LEGO decouples network reconfiguration from flow assignment and routing. Reconfiguration might be able to achieve better performance if two problems are jointly considered. Even though there have been a couple of proposals to solve this problem [8, 89], they only provide theoretical bounds without considering practical system issues. Even though its design goal is to recover from network failures as a best-effort service, LEGO is the first step to solve this optimization problem, which we will address in future work.

Use of LEGO in IEEE 802.11b/g WMNs: LEGO is mainly evaluated in IEEE 802.11a networks, where 13 orthogonal channels are available. However, LEGO can also be effective in a network with a small number of orthogonal channels (e.g., 3 in IEEE 802.11b/g). Because LEGO includes a link-association primitive, it can learn available channel capacity by associating with idle interfaces of neighboring nodes, and it further limits the range of a reconfiguration group (e.g., nodes within 4 hops).

3.6.2 Concluding Remarks

This chapter has presented a localized self-reconfiguration framework (LEGO) that enables a multi-radio WMN to autonomously recover from wireless link failures. LEGO generates an effective reconfiguration plan that requires only local network changes by exploiting channel and radio diversity as well as a constraint graph. Furthermore, LEGO effectively identifies reconfiguration plans that satisfy applications' QoS constraints, admitting up to two times more flows than static assignment, through QoS-aware planning. Next, LEGO's on-line reconfigurability allows for real-time failure detection and network reconfiguration, thus improving channel-efficiency by up to 92%. Our experimental evaluation on a Linux-based implementation and *ns-2*-based simulation demonstrated the effectiveness of LEGO in recovering from local link failures and in satisfying applications' diverse QoS demands.

CHAPTER 4

Mobile Autonomous Router System for Dynamic (Re)formation of Wireless Relay networks

4.1 Introduction

In multi-hop wireless networks, fading and shadowing often degrade the quality of wireless links and require time-consuming and expensive network deployment and management efforts, especially for manual adjustment of nodes' placements and configurations [107]. Significant efforts have been made to improve Quality-of-Service (QoS) and reduce management costs of wireless relay networks. For example, measurement-driven deployment of relay nodes determines their placement positions that meet the required network QoS [19]. Rate-adaptation and transmission-power-control algorithms allow for dynamic selection of modulation schemes and transmit-power levels at fixed positions [101, 113]. Multiple-input-multiple-output (MIMO) or multiple interfaces enable a node to exploit spatial diversity by adaptively choosing the best antenna or antenna element [2, 102].

In spite of these efforts, wireless relay networks still suffer from several fundamental limitations. First, changes in the physical environment of a wireless relay network (e.g., due to dynamic obstacles and interferences) often calls for *manual* link-quality measurement and node relocation over a large coverage area which are tedious, time-consuming and costly. Second, even if relay nodes can make simple movements to improve link bandwidth, these adjustments determined by geographic distance or node density [25, 111] are not

guaranteed to improve the overall QoS [3, 31]. Third, relay nodes may be able to relocate themselves "optimally" (in some sense) by using simple Signal-to-Noise Ratio (SNR) or traffic volume information, but accurate characterization of spatial wireless link-quality and efficient node relocation are key to the optimal relocation of relay nodes [115].

To overcome the above limitations and challenges, we study the feasibility of using a commodity mobile robot for each wireless relay node. Specifically, we propose a *mobile autonomous router system* (MARS) that enables a wireless relay node to (i) characterize wireless link conditions over the physical space and (ii) seek and relocate to, the best reception position to form string-type relay networks. First, MARS is equipped with a measurement protocol that defines and characterizes spatial wireless link-quality. Mounted on a mobile robot, each MARS node moves and measures wireless link-quality over the deployment space using the measurement protocol. Furthermore, MARS captures unique correlations of link-quality with environmental factors, such as distance, obstacles, or interference sources, which are useful in reducing measurement space (see Section 4.5.3).

Next, MARS includes a spatial probing algorithm through which the node can efficiently find its optimal position that satisfies the bandwidth demand on its link. Alternating between measurements and movements, this algorithm guides the robot to identify 'interesting' space to probe. This space is then explored at progressively finer resolutions until a locally optimal position is found. Moreover, the algorithm enables a set of MARS nodes to *cooperatively form and adjust wireless relay networks* in case link conditions or QoS demands change.

Finally, MARS includes a light-weight positioning system that provides location information to the robot, and is currently implemented for indoor environments. This positioning system does not require any expensive infrastructure support, such as cameras and other sensors, but uses natural landmarks, which are easily obtainable with a semi-automated collection procedure (as detailed in Section 4.6.3). Moreover, even though MARS is designed for challenging indoor environments, such as office buildings or large retailer shops, it is flexible enough to use any type of positioning system, depending on deployment scenarios (e.g., Global Positioning System (GPS) in outdoor environments).

A prototype of MARS has been built with commodity mobile robots and IEEE 802.11-

based wireless routers, and its algorithms have been implemented in Linux using a combination of kernel- and user-space functionalities. Our experimental evaluation results on a prototype implementation of MARS indicate that MARS achieves, on average, 95% accuracy for spatial measurements, and finds locally optimal locations with 3 times less measurement overhead than exhaustive spatial probing. Finally, the positioning system in MARS achieves an average location error of 7cm.

The rest of this chapter is organized as follows. Section 4.2 describes the motivation behind this work. Section 4.3 presents the software architecture and a hardware prototype of MARS. Sections 4.4–4.6 detail the core components of MARS. Section 4.7 presents the evaluation results of our MARS implementation. The chapter concludes with Section 4.8.

4.2 Motivation

We first argue for the need of a mobile autonomous router system (MARS) in wireless relay networks, and then discuss why existing techniques cannot be used for MARS.

4.2.1 Why Mobile Autonomous Routers?

Due to their open and continuously-changing deployment environments, wireless relay networks often incur high measurement and (re)configuration costs [7, 24, 98]. Even after their deployment, relay networks may experience severe QoS degradation, require additional measurements, and/or need to adjust placement of the relay nodes, as their physical environment changes. Even though various techniques (e.g., transmission-power control, rate adaptation) and technologies (MIMO, multi-radios) have been proposed, their bandwidth improvement is essentially limited by the surrounding environment. For example, assuming that nodes already use their maximum transmission power, a *stationary* node behind the wall might not be able to improve the bandwidth of link to another node on the opposite side of the wall using the existing techniques.

In contrast, by utilizing their mobility, mobile wireless relay routers can overcome the spatial dependency of link-quality. Being aware of diverse link-quality at different loca-

tions, mobile wireless routers can automatically improve network performance and also offer several benefits in wireless relay networks as follows.

- *Extension of AP's range*: users near the boundary of an AP's coverage might experience intermittent (dis)connectivity. A mobile wireless router can extend the AP's range by relocating itself near the limit of the AP's communication range and relaying users' traffic.
- *Deployment of wireless relay networks*: for rural areas or outdoor events, multi-hop relay networks are an inexpensive way to provide connectivity, but their optimal placement is still a challenging task. A group of mobile routers can identify a proper position of each router, forming a relay network [32].
- *Adjustment of wireless routers' placement*: many nodes in a relay network need to re-adjust their position to deal with environmental changes (e.g., dynamic obstacles in urban areas), but manually adjusting locations of each pair of nodes incurs significant time and management costs [92]. Equipped with (albeit limited) *local* mobility, mobile routers can cooperatively and automatically adjust their locations to improve connectivity and link-quality.
- *Improved resilience to DoS attacks*: wireless networks are vulnerable to denial-of-service (DoS) attacks such as jamming, and channel hopping provides limited resilience to wide-band jamming. However, because of the spatial locality of such attacks, mobile wireless routers in a relay network can improve resilience to such attacks by physically moving away from the jammed area [115].

Motivated by the above and other likely scenarios, our goals are to (i) design an autonomous robot-based router system that accurately characterizes the quality of wireless link over space and that effectively optimizes a router's position based on the thus-obtained characteristics, and (ii) prototype and evaluate such a mobile router, especially for string-type relay networks, as illustrated in Figure 4.1.

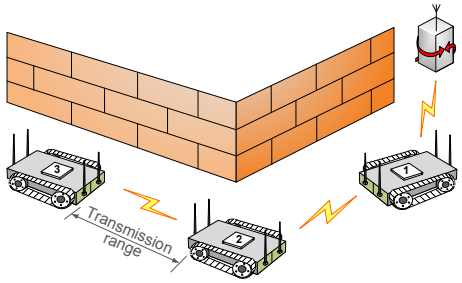


Figure 4.1: A group of mobile routers can cooperatively move and form string-type wireless relay networks by being aware of spatially diverse link-conditions over deployment areas.

4.2.2 Limitations of Existing Approaches

There has been a significant volume of work on link-quality awareness and the use of mobility. We discuss pros and cons of using existing approaches to designing a mobile router.

Link-quality awareness: Mobile routers (relay nodes) must be able to accurately measure wireless link-quality[1] over physical space. There have been numerous link-quality measurement studies and techniques in wireless mesh networks, large-scale WLANs, and wireless sensor networks [23, 27, 57, 70, 116]. Their insights and solutions, however, focus on stationary networks where the physical space of measurements is fixed. The authors of [43] considered the link-quality change resulting from location changes in mobile ad-hoc networks (MANETs). However, their solution only deals with a binary (ON/OFF) connectivity model based on simple metrics such as SNR and distance, and thus, is not suitable for capturing various channel conditions over deployment areas [3].

Node mobility: Mobile routers must be able to exploit their mobility in conjunction with link-quality-awareness. Mobility in wireless networks has been considered from various perspectives. First, flip-type mobility has been proposed for sensor nodes to make a one-time random movement [25]. Second, use of sophisticated robots in hybrid sensor networks has been proposed, and the tradeoff between node density and node mobility has been studied [111]. Third, using mobile sensors against radio jamming attacks has been considered,

[1] In this chapter, we consider the packet-delivery ratio or bandwidth as link-quality parameters of interest.

and movement-decision metrics (i.e., SNR and the amount of traffic) have been proposed [115]. Finally, in MANETs, users' mobility is exploited for mobile devices to improve their end-to-end throughput [45, 67]. However, the mobility used in existing approaches is not closely coupled with link-quality-awareness, reducing the chance of making optimal movements.

Position-awareness: Mobile routers with limited battery and CPU power must be equipped with a light-weight positioning system that does not require any extensive computation or expensive infrastructure support. On the one hand, positioning in an open sky outdoor environment is relatively easy and fast since GPS provides accurate position information. On the other hand, positioning in an indoor environment or an outdoor urban canyon is challenging, and many solutions have been proposed. In the area of robotics, the use of different hardware (sonar, laser, compass, and video camera) has been explored [20, 85], and various assumptions (e.g., known or unknown landmarks) and algorithms (training or search) have also been investigated [109]. However, their hardware and computation costs have to be carefully considered, depending on the underlying applications. In the field of wireless networks, using AP-based landmarks [12], ceiling-mounted sensors and listeners [86], and video cameras [33] have been proposed, but these approaches need installation of a positioning infrastructure or cooperation from the network infrastructure (e.g., APs).

Feasibility: Mobile routers have to be easy and fast to build with commercial components and should be readily available for system evaluation. Depending on their deployment scenarios, the robots may need various capabilities, such as climbing stairs or obstacles, withstanding fire, and carrying extra batteries. However, the robots in this work have only basic driving capabilities (forward/backward/spin), but are adequate for evaluating the proposed design of mobile routers.

4.3 MARS Architecture

We now present the architecture of MARS. We first describe its design rationale and software architecture. Then, we present our current hardware prototype of MARS that is

used for link-quality measurement and system evaluation.

4.3.1 Design Rationale

MARS is designed as a distributed system in which each MARS node autonomously adjusts its position to meet the network requirements (e.g., bandwidth and network coverage) via the following distinct features.

- *Patch-based spatial measurement*: To characterize link-quality over a wide area, MARS divides space into fixed-size squares (or *patches*) and measures the link-quality in selected patches. This divide-and-conquer approach enables MARS to locate subspaces or areas where the quality of links meets the QoS requirements within the entire deployment area.[2]
- *Hierarchical spatial probing*: MARS takes a hierarchical approach to finding locally optimal positions. Unlike exhaustive spatial probing, MARS incrementally probes space as needed. By repeating the loop of spatial measurement and local spatial-probing decision, MARS reduces the probing overhead.
- *Infrastructure-less hybrid positioning*: MARS includes a positioning system for the case where no external positioning system is available. By exploiting the benefits of both dead-reckoning and physical-landmark-based positioning, MARS achieves positioning accuracy at a reasonable training cost.
- *String-type relay networks*: Prototyped with commodity robots and IEEE 802.11 wireless cards, MARS supports the formation of a string-topology relay network with neighboring MARS nodes for the purpose of wireless range extension, as illustrated in Figure 4.1.

[2] In this chapter, we consider link bandwidth (or packet delivery ratio (PDR)) as the QoS parameter, and the two terms are used interchangeably.

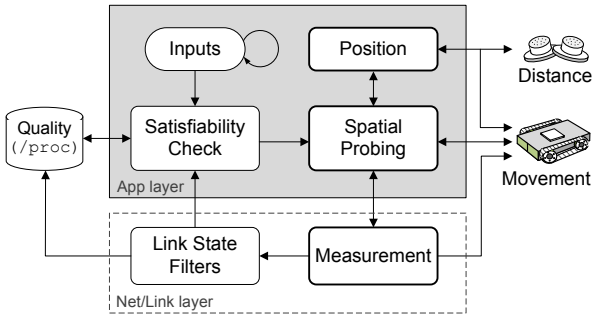


Figure 4.2: *MARS Software architecture.* MARS is designed across the application, network, and link layers in a router, and controls sensors for measuring distance and a robot for movement.

4.3.2 Software Architecture and Operations

Following the above rationale, the software architecture of MARS is designed as shown in Figure 4.2, and operates as follows. Initially, a MARS node receives the bandwidth requirement of link to AP or a neighboring node and then checks if the node's current position meets the requirement, labeled as *Satisfiability Check* in the figure. If the current position does not satisfy the requirement, MARS then decides which direction it has to move-and-measure (*Spatial Probing*) based on previous link-quality information (*Quality*). Next, if further measurements are necessary, MARS moves to a different location (*Movement*) and measures the link-quality at the new location (*Measurement*). Based on the measured link-quality, MARS again checks the bandwidth satisfiability. This procedure is repeated until MARS finds the position that satisfies the bandwidth requirements. Finally, for each movement, MARS maintains the node's position information (*Position*) using the distance information periodically measured by sonar sensors (*Distance*).

We will detail the three core components of MARS: spatial measurement protocol in Section 4.4, spatial probing algorithm in Section 4.5, and positioning system in Section 4.6.

4.3.3 Hardware Prototype

Before detailing the algorithms of MARS, we describe its hardware prototype built for our measurement and evaluation of MARS. Figure 4.3 depicts the hardware prototype of MARS, which consists of a mobile robot, a multi-radio wireless router, and sonar sen-

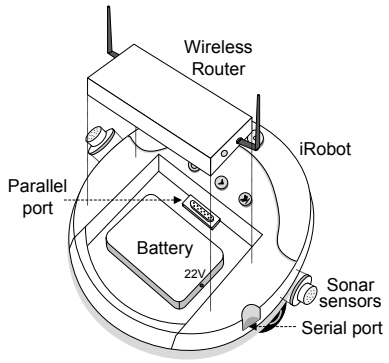


Figure 4.3: *MARS hardware prototype*. A MARS node is prototyped with an iRobot, a wireless router, and sonar sensors.

sors: (i) MARS uses iRobot Create [53] for mobility, which provides a well-defined API for movement control (e.g., a granularity of 1cm movement) and is powerful enough to carry a network node as in [33, 96]; (ii) MARS is equipped with an RB230 wireless router (233Mhz CPU, 128MB memory) [100], and the router is equipped with two IEEE 802.11 miniPCI NICs, each with a 5 dBi omni-directional antenna. In addition, this router includes a serial port for communication with the robot and sonar sensors, and is connected to an external battery that can provide 5 hours of continuous CPU and wireless usage at maximum capacity; (iii) MARS is equipped with a sonar sensor on each side of the robot to measure the distance between the robot and the surrounding obstacles. This sensor is cheap (about $25 apiece) and provides accurate distance information (error of less than 2.5cm) to objects placed at up to 6m in line of sight for the positioning system.

4.4 Measurement Protocol

We first overview challenges in designing a link-quality measurement protocol, and then propose novel point and spatial link-quality measurement techniques tailored for MARS.

4.4.1 Overview

Key challenges in measurements are (C1) how to measure the link bandwidth at a given position/time and (C2) how to characterize the quality of links as a function of physical space. First, to determine if the current position's link satisfies the bandwidth demand,

the measurement protocol has to be able to accurately estimate the link bandwidth at each position. Moreover, the protocol has to capture the link bandwidth as quickly as possible, to reduce probing time, or equivalently node's energy consumption.

Next, MARS has to be able to derive the overall quality of links in a certain space, mainly for intelligent selection of measurement space within a large deployment area. Even though wireless link-quality may change depending on the node's location [93], MARS needs to be able to characterize and differentiate spaces with respect to the quality of links. In addition, this characterization must be accurate even in the presence of adverse environmental factors, such as moving obstacles or short-term interference, which may cause temporal variations in link-quality measurement.

4.4.2 Point Measurement Technique

To meet the accuracy and time constraints, MARS includes a *unicast-based active probing* technique along with a *cross-layer design principle*. There have been numerous techniques for estimating wireless link bandwidth. First, the SNR-based approach has been extensively investigated for the PHY layer based on information theory [93], but the correlation between SNR-based bandwidth estimation and packet-level bandwidth estimation is shown to be weak [3, 97]. Second, the broadcast-based packet-delivery ratio (PDR) measurement technique has been widely used in multi-hop routing metrics such as ETX (Expected Transmission Count [30]) and ETT (Expected Transmission Time [36]). However, due to different underlying communication settings (e.g., modulation schemes) of the broadcast from those of actual data transmission, the technique has limitation in yielding measurement accuracy, even if the broadcast is modified to use various modulation schemes [42, 58]. Next, the packet-pair technique used in [36] is bandwidth-efficient, but sensitive to the short-term fading effect due to its use of a small number of probing packets. By contrast, the measurement technique in MARS uses a set of unicast probing packets to mimic actual data transmissions for accuracy, and evenly spaces the packets throughout a given measurement period (e.g., 1 packet every 50ms for a 2s-period) to minimize the undesirable effect of channel fading.

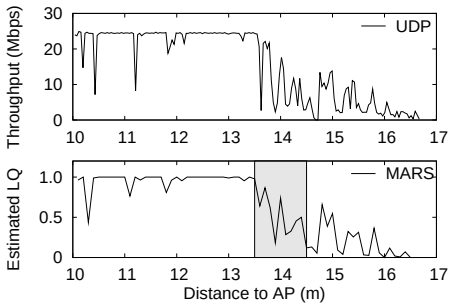


Figure 4.4: *Point measurement accuracy*. Link-quality (PDR) measured by MARS (bottom figure) is accurate enough to capture the actual bandwidth (top figure) of a link over a line of 7m from a remote AP.

Furthermore, as shown in Figure 4.2, the measurement technique in MARS is designed and implemented across the network and link layers to further improve measurement accuracy. After receiving a point measurement request from the spatial probing algorithm, the measurement protocol at the network layer sends a set of unicast probing packets to a neighboring node. At the same time, the protocol at the device driver passively monitors their transmission results based on MAC's feedback. The use of this feedback at the sender side allows for capturing the total number of (re-)transmissions made by MAC for delivering the probing packets, yielding an accurate PDR of the link. Next, to capture link asymmetry, MARS requests the neighboring node to execute the same probing procedure in the opposite direction. Finally, the PDRs of both directions are stored in the $Quality$ database at the MARS node, and the protocol notifies the completion of the point measurement to the spatial probing algorithm.

Our measurement evaluation of the above protocol shows that the active probing technique in MARS achieves higher than 95% accuracy in estimating link bandwidth. Figure 4.4 shows the progressions of actual UDP throughput (upper figure) and PDRs measured via the point measurement technique (lower figure) of a link between MARS and an AP over a straight line of 7m. As shown in the figures, MARS's measurement is indeed close to the actual link bandwidth at the cost of 20 probing packets for a 2s-period at each position. Moreover, from the point measurements, MARS can also identify a network boundary (gray bar in the lower figure) that may marginally satisfy the link bandwidth.

4.4.3 Spatial Measurement Protocol

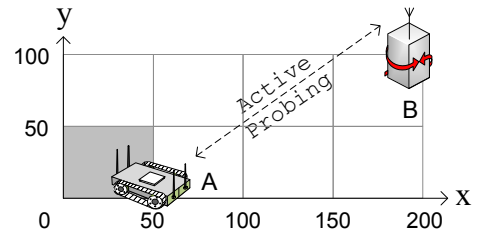
Next, to characterize link-quality as a function of space, MARS essentially uses a divide-and-conquer approach built upon spatial locality in link-quality. There have been numerous techniques proposed to model spectrum propagation over space. First, empirical models such as log-distance path loss model[93] determine the propagation based on mathematical models and empirical off-line measurements in other similar environments. However, the temporal and spatial granularities of such measurements are too coarse (e.g., retail stores during daytime) to predict the propagation in a heterogeneous deployment environment. Next, ray-tracing-based models can predict the propagation of a signal by tracing rays from a transmitter at uniform angular intervals in all directions [103]. However, this model requires the locations and thickness of walls, ceilings, and floors along with information about construction materials. Third, neural-network-based models such as a multiplayer perceptron algorithms have been proposed for cellular networks [78], but they need an extensive training set of terrain information and exhaustive SNR measurements.

By contrast, MARS relies purely on on-line link-quality measurements over unit space without requiring information about physical environments. Specifically, MARS divides space into rectangles or *patches* of fixed size (e.g., 50cm $\times$ 50cm in indoor environments[3]) and measures the quality of link between itself and a fixed neighboring node, while changing its location within the patch. These measurements are then averaged for deriving spatial link-quality as follows:

$$SPDR_i = \frac{1}{n}\sum_{j=1}^{n} PDR_j, \quad PDR_j = \frac{1}{m}\sum_{l=1}^{m} r_l \tag{4.1}$$

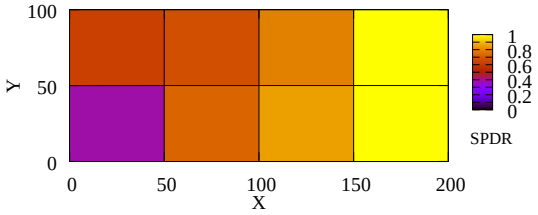
where SPDR$_i$ is the spatial link-quality of patch i, PDR$_j$ is an average PDR of m point measurements at vertex j, and r is the PDR of one point measurement at a given orientation angle. The intuition behind this definition is to exploit spatial locality among the quality of links within a small space [116]. By using a small number of sample measurements within

[3]This size is chosen according to our measurement study, but it changes dynamically, depending on space to probe (outdoor or corridor) and link parameters (transmission power).



x	y	time	neighbor	PDR
0	0	0	B	0.4
0	50	7	B	0.45
50	50	14	B	0.5
50	0	21	B	0.47

(a) Spatial LQ measurement protocol



(b) Spatial LQ measurement

Figure 4.5: *Spatial link-quality measurement*. (a) MARS measures point link-quality at each vertex of a patch. (b) The point measurement results are then used for deriving SPDR of the patch.

the space, SPDR can represent an average bandwidth of links in the space. Furthermore, as we will describe in Section 4.5, the patch-based definition allows the spatial probing algorithm to selectively and incrementally probe an area of interest, as opposed to probing the entire area.

Finally, to reduce temporal and spatial variations in link-quality measurements, MARS not only takes multiple measurements per patch, but also introduces randomization. As shown in Figure 4.5(a), a MARS node conducts point measurement at each vertex m times. In addition, for each point measurement, the node randomly changes its direction so that measurements can reflect as diverse links as possible. Here, m is dynamically adjusted, depending on the variance of point measurements (m=3 by default). MARS also adjusts the number of vertices (n=4 by default) within each patch, if the link-quality variance among the vertices is larger than a given threshold. Figure 4.5(b) shows SPDR measurements in the topology of Figure 4.5(a). As shown in the figure, SPDR normalizes the quality of links for each patch given a variance constraint (0.1 in PDR), and effectively differentiates

spaces with respect to link quality.

4.5 Spatial Probing Algorithm

With the measurement protocol, we present a spatial probing algorithm that guides MARS nodes to find locally optimal positions in a string-type relay network.

4.5.1 Overview

With a group of nodes, the objective of the spatial probing algorithm in MARS is to *cooperatively* form a relay network by efficiently finding a locally optimal node position. However, main challenges are (i) how to coordinate a group of MARS nodes to form a relay network and (ii) how to find an 'interesting' space for each MARS node to probe. First, coordinating a group of relay nodes is challenging because the link-quality between neighboring nodes depends heavily on each other's position. Furthermore, this coordination is not an one-time operation, but a recurring operation due to changes in the physical environment or bandwidth demand.

Next, during the formation of a relay network, each MARS node has to efficiently find an $optimal$ position at the least measurement cost. Finding an optimal node position in a relay network is essentially equivalent to maximizing the physical distance between adjacent relay nodes, while satisfying the bandwidth demands of their links. On the one hand, exhaustive measurements over a target area might be able to provide globally optimal location information, but it might require significant amounts of energy and time. On the other hand, finding a locally optimal position may cause a local maximum, which may result in either poor link-bandwidth or reduced network coverage.

4.5.2 Iterative Network Formation & Adjustment

For deployment and adjustment of a relay network, the spatial probing algorithm uses an iterative approach for both energy-efficiency and reduction of the coordination overhead. Let us consider the following deployment scenario. Starting from a gateway, Sam,

a network administrator, periodically drops a MARS node ($MARS_1, \cdots, MARS_n$) along corridors like a trail of breadcrumbs [107], in a way that neighboring MARS nodes can hear heartbeats of each other. After deploying n nodes in a chain, Sam requests the deployed nodes (e.g., $MARS_i$) to cooperatively optimize their position for meeting the bandwidth (bw) requirement of link to a previous node ($MARS_{i-1}$) or the gateway. One way of coordinating the optimization would be use of a centralized approach in which each node sends its measurement results to the gateway, and the gateway can calculate the best position of each node. However, this approach requires each node to conduct extensive link-quality measurements over the entire local space. Furthermore, since the bandwidth of one link depends on both end-nodes' locations, the number of measurements that each node conducts increases quadratically. For example, assuming that there are m patches that each node needs to explore, the node has to measure the m patches for each patch of the previous node—$O(m^2)$. Next, a distributed approach would help avoid the need for exhaustive measurements and allow adjacent nodes in the chain to locally identify optimal positions. However, due to the nature of the chain topology in a relay network, even if nodes of one intermediate link locally optimizes their position, the nodes might need to re-adjust their positions after their "parent" or upstream nodes close to the gateway optimize their positions.

By considering such measurement overhead and dependency, the spatial probing algorithm in MARS optimizes its location only after the previous node finds its locally optimal position, based on the iterative approach. As explained in Algorithm 3 (1), the first node ($MARS_1$) optimizes its position with a gateway, and then the child nodes optimize their positions in order.[4] Because a parent node's location of a link is fixed, a child node needs to measure only m patches over the parent node, which can be further reduced by incorporating a hierarchical approach (see the next section). In addition, the iterative approach facilitates other complex topologies, such as tree or DAG with a linear increase of complexity. For example, once a string relay network is formed, each intermediate node in the network can create another relay network starting from itself, and apply the same iterative

[4]We assume that during the optimization, each node can maintain link connectivity with neighboring nodes, because the heartbeat is transmitted using low-rate, and thus reliable, broadcasts.

Algorithm 3 Spatial probing in MARS

(1) Main function for formation in MARS$_i$ (bw, $nextloc$)
1: /* bw: *bandwidth demand of link with MARS*$_{i-1}$ */
2: /* $nextloc$: *location of the next node, MARS*$_{i+1}$ */
3: wait until node receives $done$ message from MARS$_{i-1}$;
4: optimize node location by calling the function (2);
5: send $done$ message to MARS$_{i+1}$;

(2) Coarse-grained spatial probing (bw, $nextloc$)
6: face toward $nextloc$;
7: **for** i=0; $i < N_g$; i++ **do** /* N_g *is the number of grids* */
8: move-and-measure corner patches of grid i;
9: **if** SPDR of the patches is greater than bw **then**
10: $g_{id} \leftarrow i$;
11: **end if**
12: **end for**
13: move to the grid g_{id} and call the function (3)

(3) Fine-grained spatial probing (g_{id}, bw)
14: $p_o \leftarrow$ current position;
15: **while** SPDR(p_o) satisfies bw **do**
16: move-and-measure neighboring patches of the patch p_o;
17: $p_m \leftarrow$ the patch with the maximum SPDR among neighbors
18: **if** SPDR(p_m) $> bw$ **then**
19: $p_o \leftarrow$ location of the patch p_m;
20: **end if**
21: **end while**

procedure to build a tree topology.

In addition to the formation of a relay network, MARS also takes the iterative approach to handle link-quality fluctuations in intermediate links. Each MARS node periodically monitors the quality of link to the next (or child) node, and if that link's quality is below the bandwidth requirement, then the child node should start the adjustment procedure. Subsequently, the child nodes in the remainder of the relay network optimize their position iteratively. We show the effectiveness of this optimization approach in Section 4.7.3. Note that this iterative adjustment can be extended for an intermediate node to move and optimize links to both its parent and child nodes in order to avoid the propagation of adjustment requests. This is an interesting, but challenging problem, which is a matter of our future inquiry.

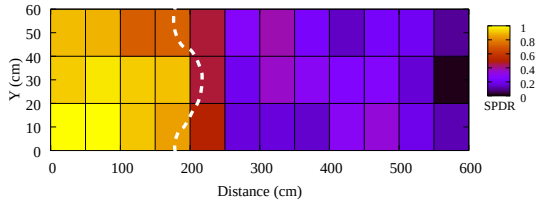


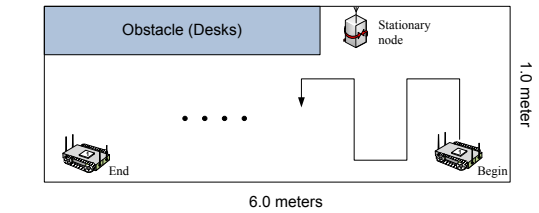
Figure 4.6: *Spatial correlation with distance*. Spatial link-quality is correlated with distance between the two end-nodes of a link. MARS can identify the boundary (dotted line) that satisfies bandwidth demand (SPDR = 0.7).

4.5.3 Hierarchical Position Optimization using Correlation

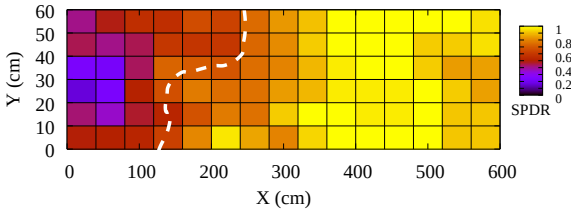
For each link, a child MARS node optimizes its position by finding a patch whose SPDR meets the bandwidth demand, and further among measured candidate patches, the node chooses the farthest one from its parent node to maximize coverage of networks. In fact, there may exist multiple optimal *positions* that meet the bandwidth demand, and a greedy measurement/movement strategy could lead the node to reach one of the farthest positions. However, relying on a point measurement is often erroneous due to spatially diverse link-quality. Moreover, even after its initial deployment, the robot may need to frequently (re)adjust its position to cope with temporal variations in link-quality. Instead, by using SPDR, MARS positions itself in a patch where the majority of positions meet the demand with a certain variance bound.

A main challenge now is how to "efficiently" find such a patch, as opposed to relying on exhaustive search over the entire deployment area. To overcome this challenge, the spatial probing algorithm in MARS exploits characteristics in spatial link-quality measurements. If the measurements can capture correlations of spatial link-quality with stationary factors such as distance, obstacles, or interference source, then the probing algorithm can adaptively adjust the granularity of measurements. To confirm this characteristic, we have conducted two interesting measurement studies as follows. First, we study the correlation of SPDRs with distance. We use an empty room (600cm $\times$ 60cm space) in our lab and an idle IEEE 802.11a frequency to exclude the other effects such as interference, obstacles and moving objects. In addition, we use a reduced transmission power of 5dBm to identify the correlation given the limited room space.[5] Then, we place one stationary node and

[5]Note that the results from these controlled settings are also consistent with those in real-life



(a) Scenario



(b) Effect of an obstacle

Figure 4.7: *Spatial correlation with an obstacle.* Figure 4.7(b) plots MARS's measurement result of SPDR in our office (Figure 4.7(a)). SPDR shows diverse spatial link-quality and correlates with the stationary node and the obstacle. The dotted line also shows MARS's identification of an interesting network boundary that satisfies the bandwidth requirements.

one mobile robot at one corner and measure the spatial link-quality while letting the robot move away from the stationary node. Here, we use an exhaustive spatial probing algorithm, which measures SPDR of every patch in the area. As shown in Figure 4.6, the correlation between spatial link-quality and distance is captured. Furthermore, this correlation helps MARS find locations that satisfy the bandwidth demand (dotted line). Second, we study the correlation of SPDRs with physical obstacles as well as the source of signal. We first place an obstacle, a stationary node, and a mobile node as shown in Figure 4.7(a). While moving toward the end-position, the mobile robot measures and collects SPDR of each patch. Figure 4.7(b) also confirms that the measurements reflect the strong correlation. These confirmed correlations suggest that the robot may take coarse-grained measurements until the robot reaches *areas* whose SPDRs are close to meeting the bandwidth requirements (correlation with distance). On the other hand, in such interesting *areas*, the robot may need to take fine-grained measurements to optimize its position because of the irregular link-quality distribution resulting from obstacles (correlation with obstacle).

Based on the above observations, our spatial probing algorithm takes a hierarchical ap-

settings, as we will show in Section 4.7.2.

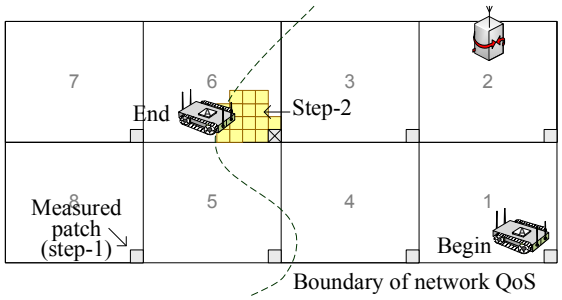


Figure 4.8: *Two-step spatial probing procedure of MARS*: MARS first finds the best among 8 grids. Then, within the grid, it identifies a locally optimal patch

.

proach. As explained in Section 4.4.3, existing spectrum propagation models either suffer from unmatched physical parameters or require comprehensive information about physical environments to predict optimal node positions. Exhaustive spatial probing requires excessive time and system resources–an area of 5m$\times$5m with a 50cm$\times$50cm unit requires 100 measurements. Instead, MARS takes a two-step hierarchical procedure as explained in Algorithm 3 (2) and (3). In the first step, it divides the probing space using a grid large enough to identify the correlation with distance,[6] and then measures a subset of patches inside each grid (coarse-grained measurement). Among them, MARS identifies patches beyond which spatial link-quality does not meet the bandwidth requirements. In the second step, within the grid including the identified patches, MARS uses fine-grained measurements to find a locally optimal location.

Let us consider an example of optimizing one link from an AP. As shown in Figure 4.8, a MARS node in grid 1 needs to find a location far away from the AP in grid 2. MARS first measures the spatial link-quality of a corner patch (50cm$\times$50cm) in each grid (5m$\times$5m), and then identifies grid 6 as the farthest grid that contains a bandwidth-satisfying patch, shown with a cross in the figure. Next, within the identified grid, MARS uses Newton's method, which recursively selects the best neighboring patch until the node reaches a locally optimal position. Note that MARS uses Newton's method for its simplicity, but other optimization methods such as second moments or extrapolation can also be used for fine-grained measurements/movements. Nevertheless, using the hierarchical approach, MARS

[6]The grid size is determined based on the wireless technology used and the environment. We will show one measurement-based grid size for indoors 802.11a in Section 4.7.2.

significantly reduces the probing space, as shown in the example. Our evaluation results in Section 4.7.2 also confirm the benefits of the two-step procedure against the exhaustive spatial probing.

4.6 Positioning System

We present the final component of MARS, a positioning system, that provides the location information of a node.

4.6.1 Overview

The function of a positioning system is to continously maintain the accurate location information of a node for both derivation of spatial link-quality and relocation to previous measurement areas. Although it is flexible enough to adopt any positioning system, MARS uses an *infrastructure-less* hybrid positioning system, especially for indoor environments. The system is designed for using dead-reckoning (DR) combined with landmark-based positioning. Although the system deliberately adopts well-known positioning techniques from the robotics, the main purpose of this section is to share our experience in building an inexpensive positioning system tailored for a mobile router, while completing the design and evaluation of MARS.

4.6.2 Hybrid Positioning Algorithms

The positioning system in MARS consists of (i) continuous location tracking or dead-reckoning and (ii) periodic landmark-based position measurement. First, DR positioning has been widely used in many navigation systems due mainly to its simplicity. Adopting DR, the positioning system in MARS simply maintains the location information of a robot by constantly updating the robot's position. Using the robot's previous position (x) and previous movement information (δ), the system can easily estimate the robot's current position ($x + \delta$).

However, this technique accumulates errors, due to unexpected obstacles, floor condi-

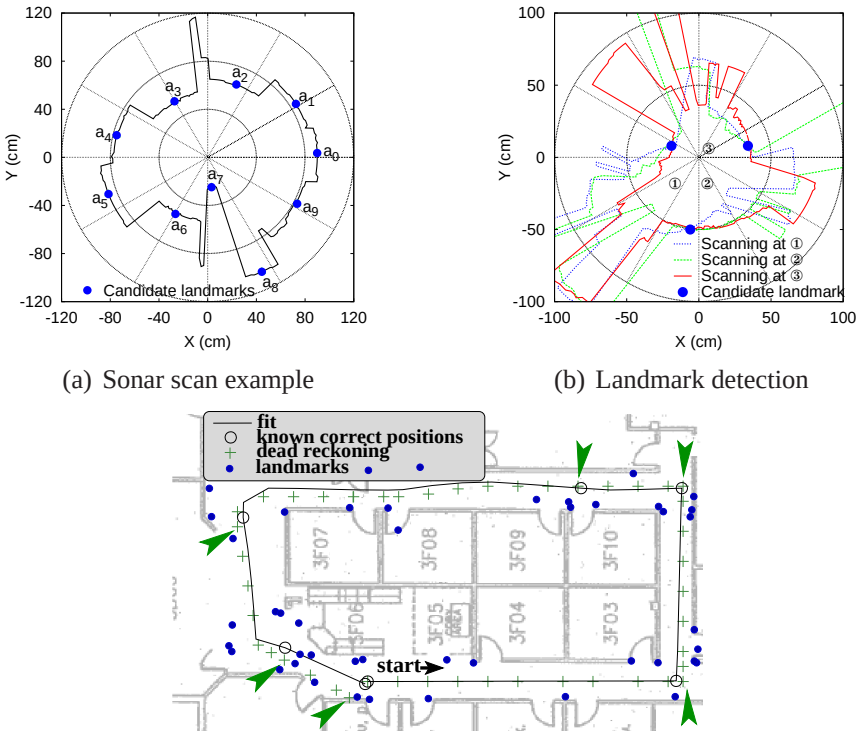
tions, or physical inertia, which are difficult to avoid. For example, assume that MARS requests a 90^o rotation followed by a movement of 2m but the robot physically rotates 93^o instead, while still believing it has rotated 90^o. After a forward movement of 2m its physical position is 10cm away from its believed-to-be position. These small errors accumulate over time and may render the robot unable to follow its planned trajectory. While use of a compass would greatly improve the performance of DR which is more sensitive to angular errors, we found that stray magnetic fields (HVAC, power cables, metallic structures) render all compasses unusable.

One method to avoid accumulation of positioning errors is to periodically measure the robot's true position and update the DR-based position. Numerous position measurement techniques have been proposed, but many require infrastructure support such as sensors and access points [12, 33] or incur extensive computation overhead for processing image/training data [20, 85]. Instead, MARS exploits naturally occurring landmarks in the environment. Briefly, given positions (x_i, y_i) of at least three landmarks and distances d_i to each landmark from the current position, one can derive the current position (x, y) by solving the following equation:

$$(x - x_i)^2 + (y - y_i)^2 = d_i^2. \tag{4.2}$$

However, the main challenges in using this technique are how to identify the landmarks (x_i, y_i) and how to accurately derive the position in the presence of measurement uncertainties of d_i. First, to sense landmarks near the robot, MARS uses sonar scanning. Spinning around 360^o, the robot collects the information of distances to its surrounding obstacles. Figure 4.9(a) shows one scan result from the robot located at (0, 0). Because the sonar has a 45^o wide beam shape, the resulting polar plot has regions of constant depth (RCDs) only for some of the objects in the environment.

Next, using the above scanning, the positioning system needs to derive the robot's current position. However, as shown in Figure 4.9(a), the scan result includes many candidate landmarks (measured as arcs $a_0, \ldots, a_9$), and the robot has to determine which arcs in-



(a) Sonar scan example (b) Landmark detection

(c) Landmark collection

Figure 4.9: *Landmark-based positioning*: (a) A sonar scanning primitive identifies landmarks around the robot's position. (b) MARS determines landmarks that are visible from at least three different locations. (c) MARS discovers landmarks using a semi-automated procedure, which is based on DR and manual measurements of a few positions.

deed represent landmarks and how these arcs are associated with known landmarks.[7] To solve this association problem, MARS uses a matching algorithm similar to the one in [66]. Briefly, given a set of arcs and known landmarks that are likely to be visible from the robot's current believed-to-be position, MARS first generates a set of feasible matchings between arcs and landmarks. Then, each feasible matching is evaluated and considered only if from the estimated robot's position, landmarks are actually sensed at measured distances in the scanning results. Finally, if there are multiple valid matchings, then MARS chooses the matching that minimizes the residual error, defined as follows:

[7]We assume the position information of landmarks has already been collected, and the next section will discuss how to collect the position information.

$$\frac{1}{n}\sum_{i=1}^{n}|(x-x_i)^2+(y-y_i)^2-d_i^2| \tag{4.3}$$

where n is the number of landmarks, (x, y) the estimated position of a robot, (x_i, y_i) the position of a landmark i, and d_i distance between (x, y) and (x_i, y_i).

4.6.3 Landmark Collection Procedure

To build the landmark information, the positioning system includes a semi-automated landmark collection procedure. This procedure consists of DR-based collection of RCDs over deployment areas, followed by offline processing that extracts landmarks from a large set of collected RCDs. Initially, a robot navigates deployment areas (corridors), and periodically stops and measures RCDs via scanning. These RCDs and the robot's scanning position information are then used to calculate the positions of potential landmarks. However, for this calculation, accurate information on scanning points is essential, and the robot obtains it in the following semi-automated way. The robot records the scanning positions using DR during the navigation, and a few positions are manually measured. Then, the manual position measurements and the DR-based positions are fed into a trajectory fitting optimization technique, described in Appendix C for generating true scanning points. Finally, based on the true scanning points and collected RCDs, the positioning system can obtain accurate positions of candidate landmarks.

Figure 4.9(c) depicts the result of the landmark-collection procedure on the corridor of an office building. The robot starts at the circle marked "start", drives in a counter-clockwise loop, and periodically stops and takes a sonar scan to collect RCDs. At the same time, the robot's DR-based positions are recorded (denoted by crosses). As expected, DR-based position becomes erroneous toward the end of the drive, where the robot is 60cm away from its DR-based position. On the other hand, using a few manual position measurements (denoted by circles), the robot's actual trajectory (the solid line) and scanning positions (omitted) are also collected.

Next, based on the information of RCDs and scanning points, the positioning system identifies "good" landmarks that have robust visibility. Good landmarks should be visible from different places, and we found that corners, door frames, or even cracks in the walls are good landmarks mainly because they are fixed and reflective to sonars. MARS only includes landmarks that are visible from at least three scanning points. Figure 4.9(b) shows the detection from the scanning results at three different positions denoted as 1, 2, 3. Good landmarks like the top two dots are heavily intersected (high visibility). On the other hand, the candidate landmark at the lowest spot is not good, since the intersection is weak–it is in fact a flat wall. Using this technique and RCD collections, MARS effectively collects landmarks as shown in Figure 4.9(c) that are used later for positioning during the spatial probing.

4.7 Performance Evaluation

We now present the evaluation results of the current implementation of MARS. We first describe an experimental setup, and then present key experimental evaluation results. Finally, we discuss some of the remaining practical issues associated with MARS.

4.7.1 Experimental Setup

We extensively evaluated MARS in a challenging indoor environment consisting mainly of office rooms and corridors (Figure 4.10). This environment includes floor-to-ceiling concrete walls and wooden doors, thus providing natural multi-path fading effects on the radio signal. In addition, IEEE 802.11a is used for wireless links since this standard provides high

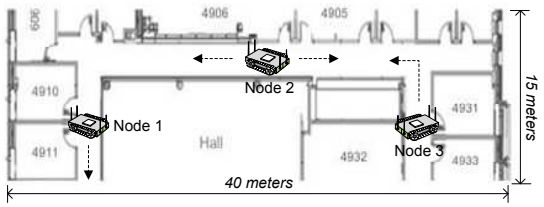


Figure 4.10: *Topology for corridor experimentation*. We evaluate the spatial probing algorithm of MARS in our Department Building.

data-rates and many idle channels. Throughout the entire experimentation a fixed data-rate is used to exclude the effects of rate-adaptation algorithms in NICs, and each radio is tuned to a medium transmission power of 10 dBm to allow multi-hop relays in a limited space. Finally, we prototype and use three MARS nodes for our experimentation, which are sufficient to demonstrate MARS' potential benefits for wireless relay networks. By performing the same experiment recursively, one can realize experimentation on an arbitrary number of hops.

4.7.2 Reducing Space to Probe

We first studied the effectiveness of the spatial probing algorithm of MARS in reducing space to probe. We place node 1 in one corridor shown in Figure 4.10 and let node 2 find the position farthest away from node 1 given QoS constraints. We first collect spatial link-quality using exhaustive probing–visit every patch and measure SPDR–over 15m $\times$ 1.2m on the corridor. Then, we run the spatial probing algorithm explained in Section 4.5.3, 20 times with the same settings. We measure the final position of node 2, the number of point measurements, and the number of patches visited.

Figure 4.11 shows the spatial link-quality measurements on the corridor using the exhaustive probing.[8] We measure the link-quality on every patch of size 50cm $\times$ 30cm. As shown in the figure, the spatial link-quality shows correlation with several obstacles (doors, walls, etc.) along the corridor as well as with distance to the stationary node. In addition, this link-quality shows several interesting boundaries denoted by white lines, or areas (e.g., the one labeled with SPDR=0.85) that satisfy link's QoS demand. However, even though this exhaustive probing provides a comprehensive SPDR map, it is very expensive in terms of energy and time to build. For example, the above spatial probing for constructing the entire map requires more than 450 point measurements.

The spatial probing algorithm in MARS reduces this overhead while maintaining reasonable accuracy in finding a locally optimal position. Figure 4.12 shows the distribution of MARS's final positions from 20 runs for each QoS demand (SPDR=0.85 or 0.45). Starting

[8]Note that we ran the same experiment more than 5 times over 2 days and saw similar results.

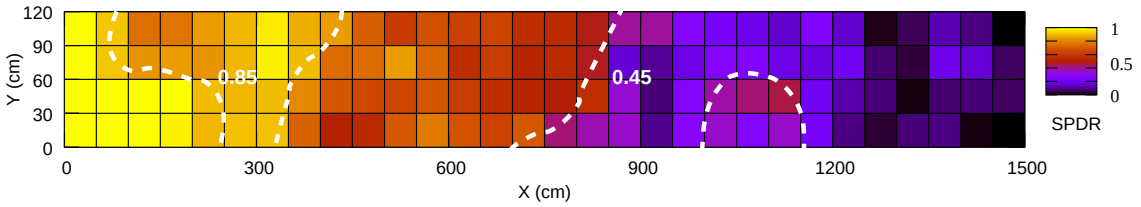


Figure 4.11: *Spatial link-quality measurement on a corridor*. Spatial link-quality in a real-life environment also shows correlation with distance and many obstacles. Areas with blue shades and white dot lines are interesting places for MARS to be positioned.

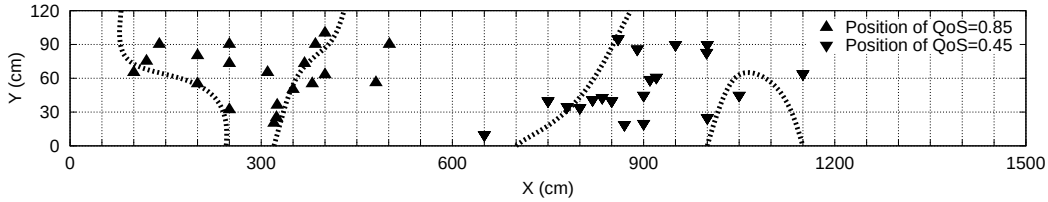


Figure 4.12: *Accuracy of spatial probing*. The spatial probing finds locally optimal positions with 86% accuracy for given QoS requests, while reducing the measurement overhead by two-thirds, compared to exhaustive probing.

from (0, 0), the robot is guided by the spatial probing algorithm to find the farthest position that satisfies the given SPDR. As shown in the figure, MARS's probing algorithm effectively determines a locally optimal position; 86% of the final locations are located within the area of required SPDR boundary, whereas the remaining 14% deviates from the boundary by, on average, 54 cm. Some of runs yielded local maxima–still satisfying the required QoS, but without reaching the farthest position. MARS could avoid this local maximum by using other optimization techniques, such as extrapolation, in fine-grained measurements, and we will explore these in future. On the other hand, thanks to its hierarchical approach, MARS reduces the number of measurements, on average, by two-thirds over the exhaustive probing, as shown in Table 4.1. For example, $\text{MARS}_{0.45}$ reduces the total number of measurements (N_m) from 465 to 150. Moreover, an interesting feature is that the increase in number of measurements from $\text{MARS}_{0.85}$ to $\text{MARS}_{0.45}$ is only 19%, although the navigation space of the former is twice larger than that of the latter, indicating the scalability of the spatial probing algorithm.

Throughout our experimentation, the spatial probing algorithm is set to use the 2.0m $\times$ 0.6m size of a grid. This value is determined based on our off-line measurement study,

Table 4.1: Efficiency benefit of the spatial probing in MARS. MARS reduces the measurement effort by two-thirds over the exhaustive spatial probing, while finding a locally optimal position. The number in a parenthesis is variance.

	Exhaustive	$\text{MARS}_{0.45}$	$\text{MARS}_{0.85}$	Benefits[§]
N_m *	465 (0)	150 (32)	126 (43)	67.3%
N_p ¶	120 (0)	24 (6)	18 (5)	80.0%

§ Overhead reduction of $\text{MARS}_{0.45}$ over exhaustive probing.

* Total number of measurements.

¶ Total number of patches visited.

which helps identify the degree of link-quality attenuation in corridor or indoor environments. As increasing the grid size by 1m, we measure the difference between two SDPRs in both ends of the grid, and repeat this measurement until the difference becomes greater than the minimum threshold (10%). Under current experiment settings, the use of a 2.0m $\times$ 0.6m grid provides 10–100% more efficiency than the use of other's. This efficiency results from the trade-off in the two-step procedure of spatial probing. For example, the larger the size of grid, the faster the algorithm can find a boundary of interest. However, it is likely that the algorithm needs more fine-grained measurements within a large-size grid.

4.7.3 Optimizing Multi-Hop Links

We also studied the effectiveness of optimizing multi-hop wireless links in the presence of changes in link conditions. As discussed in Section 4.5.2, to adapt to changing link conditions or QoS requirements, MARS iteratively and cooperatively adjusts its position to maintain the required QoS. To evaluate the performance of this adjustment, we place three nodes as shown in Figure 4.10 along different corridors and let them form multi-hop relay links (node 1↔node 2↔node 3). Next, we move node 1 to south so that the quality of the link between node 1 and 2 is degraded, and we let node 2 and 3 maintain QoS demand (SPDR=0.85). We ran the same experiment 10 times and measured the final positions of node 2 and 3 after their adjustment.

Figure 4.13 shows the effectiveness of our adjustment in coping with link condition changes. The unshaded part of the figure shows the robots' final positions (denoted as tri-

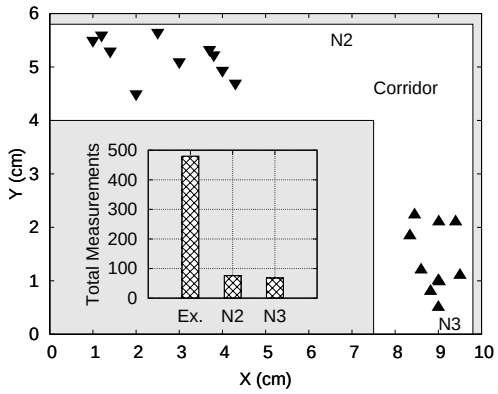


Figure 4.13: *The multi-hop optimization result.* In MARS, relay nodes optimize each position one-by-one when the link conditions changed. Triangles denote adjusted nodes' positions.

angles) on the corridor. Starting from their original positions, denoted as N2 and N3, node 2 first starts its spatial probing to adjust its position. Next, once node 2 finds a location that satisfies the QoS requirements (0.85), the node informs node 3 of its adjustment through a broadcast-based message handshake. Then, node 3 starts adjusting its own position with respect to the new location of node 2 in order to maintain the required QoS. As shown in the figure, each robot successfully senses its direction (e.g., east for node 2) without relying on movement information of node 1 and moves in the direction that node 1 moved. In addition, this iterative adjustment significantly reduces the average measurement overhead. Compared to the exhaustive probing where node 2 and 3 have to measure SPDR of every patch in the corridor, the spatial probing in MARS selectively measures SPDR over the fixed previous node. As shown in the inset graph, the iterative adjustment in MARS reduces the total number of measurements ($N_2 + N_3$) by 72% compared to the exhaustive probing-based adjustment ($Ex.$).

4.7.4 Maintaining Positioning Accuracy

We now turn to the quantitative evaluation of MARS's positioning system. We first set up one large square space (240 cm $\times$ 240cm) with four artificial landmarks in our lab. In the square, we let the robot collect positions of the four landmarks through the landmark collection technique described in Section 4.6.3. As expected, the robot properly

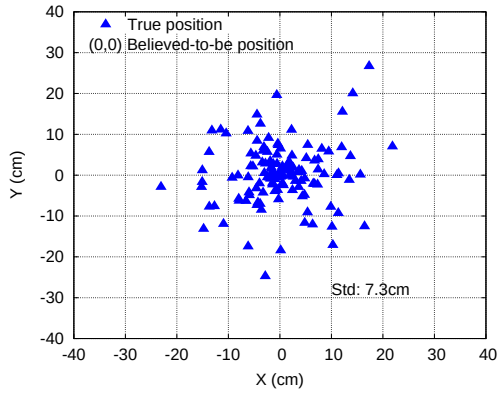


Figure 4.14: *Accuracy in correcting location error*. The positioning system in MARS accurately corrects the error in its location information, on average, within 7.3cm error.

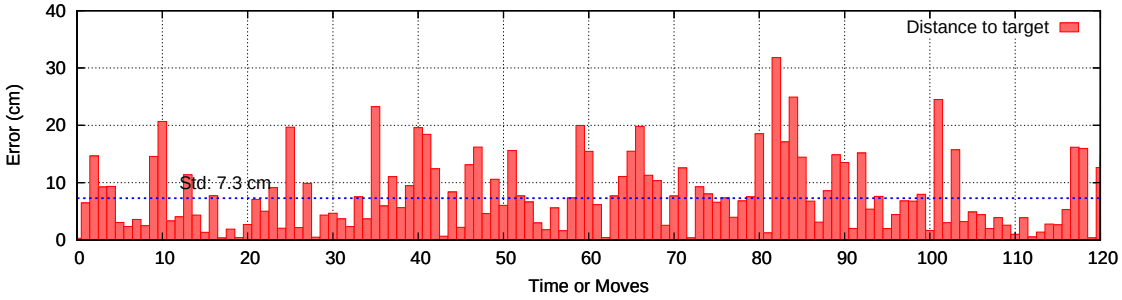


Figure 4.15: *Histogram of location error from 120 movements requests*.

detects all the landmarks with error of at most 2cm from their true positions, and this measured landmark information is used for the robot's positioning during the following random walks. At each point on the random walk, the robot updates its position (i.e., believed-to-be position) based on its previous position and movement information. At the same time, the robot takes a sonar scan (as in Figure 4.9(a)) and derives its current true position using the scanned result. Next, the robot derives the position error by comparing the believed-to-be position and the true position from the scan. Finally, the robot updates its position information with the true position and drives itself toward the next random point.

Figures 4.14 and 4.15 show the robot's true positions relative to believed-to-be positions (0, 0) and the derived location errors for 120 random walks, respectively. As shown in these figures, the MARS's positioning system keeps the average location error less than 7.3cm, thanks to landmark-based position measurements. In addition, the distributions of errors is found to be independent Gaussians on x and y with an almost diagonal covariance matrix.

This error is due mostly to the drifts caused by the heading error, but it is small enough for the robot to correct the error using the landmarks and continue its walk without unbounded growth of error.

4.7.5 Discussion

So far, we have shown the effectiveness of MARS based on our experimental results. Now, we discuss some of the remaining practical issues associated with MARS. First, *energy-efficiency* is an important issue. MARS relies on batteries to operate both a wireless router and a robot, and hence, needs to save energy to prolong the network lifetime. In Section 4.7.2, the MARS's spatial probing algorithm reduces the number of measurements by two-thirds, compared to exhaustive probing, which can be translated into energy-savings. However, MARS can solve the energy-efficiency issue by carrying extra batteries or by automatically recharing its battery with the help of an alternating node.

Next, in a large-scale relay network, MARS's iterative adjustment strategy might cause a *scalability* problem, compared to a "parallel" adjustment strategy. However, as we illustrated in Section 4.5.2, moving two nodes at the same time to optimize their positions rather increases the total number of measurements ($O(n^2)$), compared to the MARS's iterative adjustment ($O(n)$). On the other hand, there is case where an intermediate node can optimize links to both its parent and child nodes, thus avoiding the propagation of iterative adjustments to child nodes. This is interesting because this allows MARS to support complex topologies such as trees and meshes. However, such an optimization is proven to be NP-complete, and studying heuristic optimizations is part of our future work.

4.8 Summary & Future Directions

In this chapter, we have presented MARS—a mobile wireless router that is aware of spatial diversity in wireless link-quality. Using extensive experimental evaluation, we have demonstrated the feasibility and practicality of MARS in maintaining a target QoS and in forming a multi-hop relay network. The benefits of MARS stem from its unique view

on mobility: it targets occasional use of mobility to optimize communication in a mostly static network, as opposed to MANETs which aim to maintain communication in a mostly mobile network.

While the current prototype targets 802.11-based indoor applications, the idea of enhancing communication through mobility can be further developed in many directions.

- *Various scales of spatial probing*: Using the current prototype, spatial probing requires different resolutions in different physical environments. Although MARS relies on off-line measurements to determine the scale, a further study is needed on how to use simple measurements and channel models to determine the scale.
- *Fully automatic landmark collection*: The current landmark collection in MARS requires a few manual position measurements. This may be time-consuming and not scalable in large networks. We are planning to investigate ways to fully automate this procedure [109].
- *Use of directional antennas*: Directionality is another way of reducing interference and enhancing link quality, which can benefit from the link optimization procedures used in MARS. Jointly considering both *position* and *orientation* allows MARS to enable better link/topology creation, or to directionally track specific users for improved throughput.
- *Flexible deployment scenarios*: Even in cases when routers are deployed by humans [107], MARS can support the deployment scenario via iterative adjustments around each drop point. This is especially useful for inherently hazardous scenarios such as rescue or military applications, but a robotic platform that is more capable than the iRobot is necessary for such applications.
- *Reconfigurable antennas*: While in this work we experimented with the mobility of the entire node, we have found from experimentation that even small (comparable to carrier wavelength) movements can greatly affect the quality of the signal. Mobility may be exploited even in fixed access points at the antenna level, on a much smaller scale. For example, studies on MIMO have shown that inter-element distance can be a useful control for enhancing performance in sparse multi-path situations (outdoors)[102].

The MARS demonstration videos are available in a public website of `http://kabru.eecs.umich.edu/bin/view/Main/SMART`.

CHAPTER 5

Sybot: A Robot-Based Spectrum Site-Survey System

5.1 Introduction

To deal with the increasing demand for pervasive wireless connectivity, a large number of access points (APs) or relay routers are being deployed, often redundantly, at many locations within homes, offices, and campuses [38, 48]. Despite its expanded coverage, such deployment trend makes it difficult for network engineers to manage the performance and the spectrum usage of wireless networks, due primarily to heterogeneous deployment environments and time-varying spectrum-propagation characteristics [5, 88, 92].

To cope with such "chaotic" wireless spectrum environments, spectrum site-surveys have been used widely for numerous network applications and services, thanks to their ability to provide comprehensive information on spatial spectrum characteristics. For example, radio signal propagation from each AP is used for initial deployment and assessment of wireless networks [92, 99]. Accurate spectrum site-survey results are essential for accurate signal-based localization systems [12, 119]. Spatial spectrum footprints are useful to pinpoint problematic areas for network troubleshooting [115].

Although numerous spectrum site-survey techniques have been proposed, they still have several critical limitations in challenging indoor environments. First, commercial site-survey tools (e.g., [4, 72, 80]) can provide the collection of Signal-to-Noise Ratio (SNR) information by having a human navigate through the network deployment area. However, such tools require expensive and time-consuming human labor whenever there is a change

to physical (e.g., new furniture) or radio (e.g., new AP) environments. Moreover, a human's navigation is likely to generate errors or biases in measurement results. Next, a hybrid approach that uses both measurements and an empirical propagation model, has been proposed (e.g., [99]) for outdoor wireless networks, but this approach has limitations in providing fine-grained survey results—necessary for indoor localization systems [12]. Finally, a sensor-based approach [119] can reduce manual measurement efforts, but it requires the deployment of a large number of sensors, or provides limited survey accuracy due to fixed sensor locations.

To remove these limitations, we present a novel robot-based (thus *automated*) spectrum survey system, called *Sybot*, that provides spatial spectrum-condition information, especially for large indoor wireless networks, with the following salient features. First, Sybot is nothing but a wireless router mounted on an inexpensive mobile robot [33, 53, 95]. Sybot intelligently makes use of robot-based mobility to conduct spectrum site-surveys, thus automating labor-intensive manual measurements. Furthermore, by physically controlling the robot's movement, Sybot can navigate through target areas and conduct spectrum surveys in as fine-grained a manner as necessary.

Next, Sybot takes a *layered* approach to achieving both accuracy and efficiency in spectrum surveys. The layered approach allows Sybot to adaptively use three complementary probing techniques—*complete*, *selective*, and *diagnostic*—for capturing both repeatable and time-varying spectrum-condition information, depending on network usage or physical-environment changes. Briefly, when the network is not in use or lightly used (e.g., night-time), Sybot performs the complete probing for collecting information on a baseline or repeatable network condition over the entire deployment area. When the network is in active use (e.g., day-time), Sybot periodically triggers the selective probing through which the system can capture the time-variance in spectrum condition by measuring only the subset of the entire measurement space. When the network experiences abnormal spectrum conditions in local areas, Sybot uses the diagnostic probing to efficiently identify such areas and update a spectrum-condition map of only those areas, as opposed to an exhaustive survey.

We have implemented the above software components of Sybot in Linux OS running

on a wireless router mounted on an iRobot [53]. We have deployed 12 APs in our Department building, and conducted an extensive measurement study over more than 10,000 measurement points for a period of four weeks. Our experimentation results show that Sybot generates a repeatable spectrum condition map that reflects physical characteristics such as moving or stationary obstacles. Next, the selective probing in Sybot reduces the measurement space by more 56% based on the spectrum condition map. Finally, the diagnostic probing effectively identifies abnormal spectrum conditions and maintains the spectrum map to be up-to-date with 50% less measurement efforts than a navigation-based exhaustive probing.

Our analysis of the measurement data confirms that Sybot's design goals are met as follows. First, Sybot can adaptively determines the granularity of survey parameters such as time interval, unit measurement size, or total measurement space, depending on a specific site (room or corridor), AP locations (close or away), or unexpected events (obstacles or interferers). Second, Sybot can build and estimate a site-specific profile on the trade-off between accuracy and efficiency in spectrum survey. For example, Sybot can reduce the measurement overhead by more than 65% at a 3.5 dBm standard deviation in the survey results (see Section 5.5.3.3).

The rest of this chapter is organized as follows. Section 5.2 describes the motivation of this work. Section 5.3 presents the software architecture and algorithms of Sybot. Section 5.4 describes the implementation of Sybot. Section 5.5 presents our experimental evaluation of the Sybot prototype. Section 5.6 draws conclusions and discusses some of the remaining issues.

5.2 Motivation

We first argue for the need of an automated spectrum-survey system and then discuss the limitations of existing approaches.

5.2.1 Why Spectrum Site-Survey?

Due to exponentially-growing demands for wireless services and applications, both network administrators and mobile end-users have to deal with a large number of access points (APs), often with overlapped coverage, installed at many locations [38]. Despite their coverage benefits, the increasing number of network nodes face challenging coordination and performance problems due mainly to site-specific (hence *heterogeneous*) spectrum propagation from each AP.

To mitigate these problems associated with the spectrum heterogeneity, spectrum site-surveys have been used in many network applications and services as can be seen in the following use-cases.

- *Deployment & assessment of wireless networks*: Spectrum site-survey results help engineers determine the placement of network nodes [92]. Furthermore, when networks are incrementally expanded (as is often the case), the locations of newly-added nodes can be easily estimated. Even after the deployment of network nodes, a site-survey is necessary to assess network performance [99].
- *Identification of sources of interference*: Spectrum surveys of deployed or neighboring networks allow network operators to identify interference areas. Many wireless devices, such as cordless phones, may cause interference in certain areas, and measuring and using their spatial footprints are a common way to locate the interfering transmitters [115].
- *Supporting indoor localization systems*: Spectrum site-survey results or spectrum-propagation maps can improve the accuracy of location estimation by providing comprehensive RF signal signatures [12, 62, 118]. By comparing current signal strengths from multiple APs with those in maps, mobile users can accurately obtain their location information.
- *Forecasting connectivity of mobile users*: Spectrum-survey maps can help mobile users select the best AP among those within their range [81]. Although mobile devices can build a connectivity profile of visiting areas, spectrum-survey maps can instantly provide the profile even for new or changing physical sites.

Motivated by the above and other potential use-cases for spectrum site-surveys, our goal is to develop a novel spectrum site-survey system (equipped with techniques and tools) that autonomously monitors the spatial characteristics of radio propagation in challenging indoor wireless environments.

5.2.2 Limitations of Existing Approaches

There has been a significant volume of work on characterizing spectrum propagation. We discuss pros and cons of using existing approaches to build a spectrum-survey system.

Accuracy and repeatability: A spectrum site-survey system must collect information on spectrum-propagation characteristics that not only reflects a snapshot of actual network conditions, but also shows repeatable conditions over time. Many empirical radio propagation models [93] have been used for calculating the propagation path-loss. However, these models are based on extensive measurements in cellular network environments and have limitations in capturing site-specific spectrum conditions, especially in dynamic and heterogeneous indoor wireless environments.

Next, ray-tracing techniques [103] and neural network models [78] have been proposed to calculate path-loss. Ray-tracing techniques can accurately predict the propagation of a signal by tracing rays from a transmitter at uniform angular intervals in all directions. However, this model requires information about the locations, thickness, and construction materials of walls, ceilings, and floors. The neural network models, such as a multiplayer perceptron algorithm, have been proposed for cellular networks, but they need an extensive training set of terrain information and exhaustive SNR measurements.

Third, navigation-based on-line measurement tools [4, 72, 99] have been proposed. These tools help network engineers collect spectrum-propagation information by traversing a path within the network deployment area. Even though these tools allow for capturing a snapshot of each navigation, these manual measurement results can often be erroneous because of its navigation direction, measurement times, and interference. Furthermore, the results become obsolete as physical settings change.

Efficiency and autonomy: A site-survey system has to incur minimum time overhead

and human involvement to meet various requirements from applications, including initial network deployment/assessment and RF-based localization systems. First, measurement-based surveys with portable measurement tools [4, 72] are most popular, but they require network engineers to navigate through the deployment areas multiple times whenever the spectrum conditions change.

Next, the use of empirical models may reduce measurement overheads (e.g., [99]), while maintaining the accuracy of a site-survey. Even though their models are suitable for open environments in outdoor wireless networks, they provide limited information to such applications as indoor localization systems that require unique signal footprints every 1 meter.

Adaptation and awareness: A site-survey system must be able to dynamically adjust the granularity of measurements by being aware of site-specific spectrum characteristics. The navigation-based measurements rely on samples that are collected in a fixed unit space or time interval (e.g., [92]). Even though such an approach can provide uniform measurement results, the approach cannot capture spatially heterogeneous spectrum conditions. For example, if the variance of spectrum conditions with respect to an AP in the room is larger than one in the corridor, then the uniform spectrum survey is likely to have more measurement errors in the room than one in the corridor.

5.3 The Sybot Architecture

This section details the architecture of Sybot. First, we outline its design rationale and overall operations. Next, we discuss spectrum-survey metrics of interest and then detail site-survey techniques. Finally, we analyze the complexity of Sybot.

5.3.1 Overview of Sybot

Sybot is a mobile spectrum-survey system that controls the underlying mobility and collects the spatial spectrum conditions of IEEE 802.11-based wireless networks via the following salient features.

- *Robot-based automation*: Sybot is mounted on an inexpensive (<$100) mobile robot for its mobility. This robot-based mobility[1] allows Sybot to automate the collection of spectrum-condition information, hence reducing the human labor significantly.
- *Layered approach*: Sybot decomposes a survey task into three distinct but complementary layers, and includes a specialized probing technique for each layer. By selectively triggering each layer in different time scales, Sybot efficiently collects both comprehensive and time-varying spectrum condition information.
- *Use of spatio-temporal variance*: Sybot measures and computes spatial variance in spectrum condition over time, and further extracts spatial locality in spectrum condition. These spatial characteristics are then used to identify spatial network performance, such as interference areas and to minimize the measurement effort.
- *Flexible design*: Sybot is designed to be easily deployable on existing wireless-enabled robots as used in [41, 52, 54]. Such robots are becoming popular in our daily lives [10, 40], and Sybot can be installed as a service module (in the application and device-driver layers) without requiring any specialized hardware.

Algorithm 4 describes the overall operation of Sybot, which consists of the following three periods: (1) measurement period (t_m) during which Sybot navigates in the unit space (or $grid$[2]) determined by a current probing technique (p_{cur}) and measures spectrum condition with respect to each AP; (2) navigation period (t_n) during which Sybot derives spectrum condition over the grid and then chooses the measurement granularity and the next grid to be measured. Sybot recursively repeats the measurement-then-navigation (for $t_m + t_n$) until it covers the entire measurement space under p_{cur}; and (3) scheduling period (t_s) after completing measurements, during which Sybot constructs a spectrum map over all deployment areas, based on the measurement results, and determine the next probing technique and schedule time for it.

[1] In this chapter, we assume that a robot has basic driving capabilities (forward/backward/spin), which are sufficient for navigating through indoor environments.

[2] For example, a unit grid can be 20in×20in, in which Sybot incrementally moves and measures spectrum condition.

Algorithm 4 Sybot operations for t_i-th survey

(1) During the measurement period, t_m
1: $\mathbb{L}\leftarrow$ list of APs visible from current grid, g_{cur};
2: **for** j=1 to n **do**
3: collect selected survey metrics to every $a \in \mathbb{L}$;
4: randomize a robot's direction within g_{cur};
5: **end for**
6: derive spectrum conditions of g_{cur} using the measurements;

(2) During the navigation period, t_n
7: v = variance in measurements on g_{cur};
8: $g_{next}\leftarrow$ determine the next grid using v and p_{cur};
9: **if** g_{next} == NULL **then** */* i-th survey is complete */*
10: move a robot to a start-point; enter a scheduling period;
11: **else** */* continue to explore */*
12: move a robot to g_{next}; enter the measurement period;
13: **end if**

(3) During the scheduling period, t_s
14: $M(i)\leftarrow$update a spectrum-condition map;
15: c_i = variance of $M(i)$ from previous maps $\mathbb{B}$;
16: **if** $c_i \leq \alpha$ **then** */* spectrum condition is stable */*
17: p_{next} = COMPLETE; t_{next} = T_{day};
18: **else if** $\alpha < c_i \leq \beta$ **then** */* need periodic probing*/*
19: p_{next} = SELECTIVE; t_{next} = T_{hour};
20: **else if** $c_i > \beta$ **then** */* abnormal spectrum usage exists */*
21: p_{next} = DIAGNOSTIC; t_{next} = T_{min};
22: **end if**
23: schedule the next survey t_{i+1} with p_{next} at t_{next};

5.3.2 Metrics of Interest

Sybot focuses on characterizing the radio signal propagation as spectrum-survey metrics. Specifically, to obtain the signal propagation characteristic for each unit grid i, Sybot measures a received signal strength (RSS) (γ) m times and uses their mean $\overline{\gamma}_i$ and standard deviation σ_i as metrics, which are defined as follows:

$$\overline{\gamma}_i = \frac{1}{m}\sum_{j=1}^{m}\gamma_i(j) \quad \sigma_i = \Big(\frac{1}{m}\sum_{j=1}^{m}(\gamma_i(j)-\overline{\gamma}_i)^2\Big)^{1/2}, \tag{5.1}$$

where $\gamma_i(j)$ is the j_{th} RSS in grid i. Sybot collects information on signal strength by

passively monitoring periodic beacon messages from the AP, which are transmitted every 100 ms [63].[3] These measurement results for an area A at time t are used to construct a spectrum condition-map $M_A(t)$ for both γ and σ each—$M(\gamma, t)$ and $M(\sigma, t)$.

In summary, Sybot is to collect the above metrics of to generate and maintain an accurate spectrum map for a target wireless network with a minimum number of measurements (or minimal human involvement). Even though Sybot is flexible enough to survey other metrics (e.g., packet-delivery ratio), Sybot relies mainly on RSS-based metrics, not only because the RSS is a fundamental parameter to represent network performance but also because many applications such as localization systems use such metrics.

5.3.3 Layered Approach

As described earlier, Sybot adopts a *layered approach* via three spectrum probing techniques—complete, selective, diagnostic—for efficient and accurate spectrum surveys. A layered approach allows systems to decompose one complex task into several sub-tasks to improve their performance [28], whereas existing spectrum survey systems rely mostly on a single measurement technique (e.g., trajectory-based scanning), which is labor intensive because engineers have to repeatedly navigate same areas to detect environmental changes. Sybot intelligently solve this problem by decomposing a spectrum survey task into three sub-tasks not only to reduce measurement efforts, but also to cope with the unpredictable spatio-temporal variations in spectrum condition.

Figure 5.1 describes Sybot's layered approach. Briefly, when wireless networks are idle (e.g., the night time), Sybot uses the *complete* probing that collects a baseline spectrum condition, which can capture physical changes in the environment (e.g., reorganization of furniture) on large-time scales, such as days (T_{day}). When networks are actively used and interfered with by the environment (e.g., moving obstacles or co-existing network activities during the daytime), Sybot uses the *selective* probing to capture temporal variance in spectrum condition on a scale of hours (T_{hour}). When networks experience a large deviation in spectrum conditions in local areas, due to events such as severe interference or new

[3] Sybot focuses on downlinks.

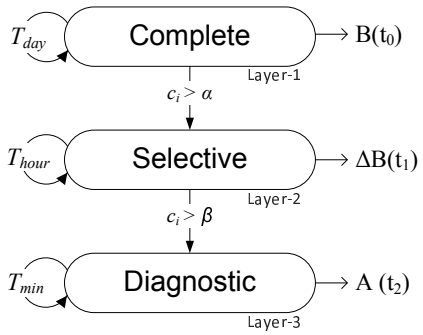


Figure 5.1: *Layered approach*: Sybot triggers different probing techniques, depending on spatial variance (c_i) in spectrum condition. α and β are thresholds to meet system requirements.

obstacles, Sybot uses the *diagnostic* probing to locate such areas and to quickly update a spectrum map over a short period of time (T_{min}).

In what follows, we will detail the above three probing techniques and their rationale.

Complete Probing

The *complete* probing is designed to obtain a baseline radio propagation characteristic specific to each AP's physical site. Numerous radio propagation models [93] and computational methods [78, 103] have been proposed. However, they are mostly suited for outdoor environments or require expensive human labor to tune parameters for each site or even different corridors.[4] Next, measurement-based approaches from static APs [88], sensors [27], or mobile users [1] help detect performance problems in certain areas, but they do not provide fine-grained spectrum condition information at every location in the entire network coverage area.

The complete probing automatically performs comprehensive spectrum surveys with a coarse-grained periodic timer (once every T_{day}) to collect repeatable spectrum-condition information governed mainly by physical environments. However, to incorporate this complete probing into Sybot, there are several challenges to be met as follows.

- *Building a comprehensive map*: The complete probing has to provide a spatially-thorough spectrum map. The approach in [99] estimates the coverage or boundary

[4] We will show this characteristic in Section 5.5.3.

map of outdoor networks with a small number of measurements, but the boundary map is not enough for such applications as indoor localization systems that rely on fine-grained spectrum condition information (e.g., every foot). The complete probing in Sybot virtually divides network deployment areas into small grids and measures the survey metrics of each grid accessible by a robot.

- *Selection of grid size*: The grid size is an important factor that determines the accuracy and efficiency of site survey. Using one grid size might not capture accurate spectrum condition of areas with large variances or might waste time to measure the spectrum condition of open areas with small variances. The complete probing adaptively determines the grid size based on the degree of heterogeneity in spectrum characteristics in a given measurement area. From a measured spectrum map, if the difference in measured signal strengths neighboring grids is less than a predefined threshold, then Sybot linearly increases the grid size for the space and uses the size for the next survey.
- *Eliminating temporal variance*: One-time measurement result of the complete probing can contain temporal variance, due to unexpected events in certain areas (e.g., moving people), and Sybot has to remove the variance for constructing a baseline spectrum map. Sybot maintains a series of spectrum probing maps $\mathbb{M}(\gamma) = [M(\gamma, t-n+1), \ldots, M(\gamma, t)]$ and utilize n recent spectrum maps to generate the baseline spectrum map $\mathbb{B}(\gamma) = \mathsf{E}[\mathbb{M}(\gamma)]$.

Figure 5.2 illustrates the complete probing. From the 'start' point in Figure 5.2(a), Sybot navigates through the measurement areas and measures spectrum condition in each unit grid 20in$\times$20in, denoted as dots. After completing the measurement of every grid, Sybot accumulates current measurements with the previous spectrum maps and generates a new spectrum map, shown in Figure 5.2(b). Finally, for each corridor or room, if the variances of most grids are less than a threshold (1dBm), Sybot increases the grid size to 40in$\times$ 40in for the space for the next survey.

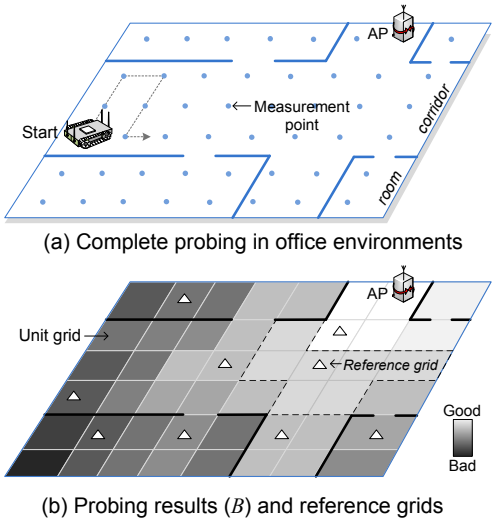


Figure 5.2: *Example of complete and selective probing*: (a) The complete probing progressively measures RSS over the target areas; (b) the complete probing result is then used to determine reference points to capture temporal variance in spectrum condition.

Selective Probing

Due to changes in spectrum usage and condition over time (e.g., in the order of hours) [48], the spectrum map needs to be updated, and Sybot uses the selective probing technique to capture such dynamics. Sensor-based network monitoring has been proposed to measure such dynamics, especially for diagnosing network performance [27] or maintaining accuracy in localization systems [119]. However, they require the deployment of a large number of sensors (e.g., 8 sensors in 30m $\times$ 15m), or they have to painstakingly determine or adjust optimal locations of sensors over time.

The selective probing makes use of robot-based mobility and previous spectrum maps to efficiently maintain up-to-date spectrum condition information. The selective probing measures only a small set of reference points $\mathbb{R}$ and estimates spectrum condition over the entire network areas. However, to implement this idea in Sybot, there are several issues to be addressed as follows.

- *Finding spatially-correlated grids*: Sybot has to find a group of grids that are spatially-correlated in spectrum condition under heterogeneous physical environments. Sybot exploits a complete probing history ($\mathbb{M}$) and characterizes the site-specific spatial cor-

relation among neighboring grids. Specifically, using the baseline spectrum map ($\mathbb{B}$), Sybot finds a block (b) of neighboring grids whose RSS (γ)'s are close to grid i's within a given tolerance (π).

- *Determining the smallest set*: Because the block b of each grid can have different sizes, multiple combinations of grids can become $\mathbb{R}$. Sybot has to determine one combination whose size is minimum. Finding a globally optimal $\mathbb{R}$ becomes an NP-hard problem—$O(2^n)$, where n is the total number of grids. Instead, Sybot uses a heuristic approach with which it iteratively includes the grid with the largest boundary set first in the reference-point set. This algorithm provides reasonable performance in minimizing the set size (see Section 5.5.3.3) with $O(nlogn + nm)$ complexity, where n is the total number of grids and m the average block size.
- *Controlling accuracy*: There exists a fundamental trade-off between efficiency and accuracy of measurements, so the selective probing must have a knob for controlling the accuracy, depending on the network requirements. Sybot uses π as the control knob. The lower π value, the higher accuracy Sybot can achieve at the cost of more measurements, and vice versa. This trade-off profile can be built and used based on $\mathbb{B}$, as we show in Figure 5.10.

Let's consider the example in Figure 5.2. Using a spectrum map (shown in (b)) and a tolerable system error of 2 dBm, Sybot determines a set of reference points, each denoted as a triangle. Then, the selective probing measures spectrum condition only for 9 grids. Finally, Sybot updates the spectrum condition of correlated grids with the measurements at their reference points.

Diagnostic Probing

When wireless networks experience local environmental changes such as appearance of new wireless interference sources or obstacles, Sybot uses the diagnostic probing to identify such areas and quickly update the spectrum-condition map of those areas. AP- or sensor-based network monitoring solutions [27, 80] can indirectly detect changes in spectrum condition. However, they still need human involvement to estimate the boundary

of problematic areas that need to be surveyed, or simply require engineers to conduct fine-grained spectrum survey over the entire coverage areas.

The diagnostic probing in Sybot identifies abnormal spectrum changes in local areas and estimates the spatial boundary of the changes using the baseline spectrum-condition map from the complete probing. In addition, by quickly measuring the spectrum condition of the identified areas, Sybot updates the spectrum map. To implement this probing technique, there are, however, several challenges to overcome:

- *Detecting abnormal changes*: Sybot must be able to determine the existence of drastic changes in spectrum condition over certain areas, and the diagnostic probing makes use of both complete and selective probing results for making decisions. Whenever the selective probing is completed, Sybot calculates the difference between its current measurement and the baseline measurement ($diff_i$=$|\gamma_i - \overline{\gamma}_i|$) of each reference grid. If the difference $diff_i$ is greater than a predefined threshold (e.g., an integer multiple of the grid's standard deviation (σ_i), then Sybot immediately triggers the diagnostic probing on the suspicious grid.
- *Speculating measurement areas*: On detecting deviations on the suspicious grids, the diagnostic probing in Sybot has to estimate the spatial boundary of the deviation. By alternating speculative navigation and spectrum measurements, Sybot not only identifies the boundary, but also updates a spectrum map. Specifically, for each suspicious grid i, Sybot progressively navigates and measures spectrum conditions of the grids within the grid i's block until their neighboring grids do not show large deviations. Because the grids within the block are likely to experience the same spatial deviation, Sybot updates their spectrum-condition information.
- *Exploiting external network monitoring information*: Sybot has to exploit network information on spectrum-condition changes. Network-monitoring infrastructures [80] can provide information on network performance degradation at specific APs. Upon receiving such information, Sybot triggers the selective probing over the APs' coverage areas, and initiate the diagnostic probing to update the areas' spectrum map.

Suppose that a new obstacle is placed at one location, as shown in Figure 5.3 (a). Sybot

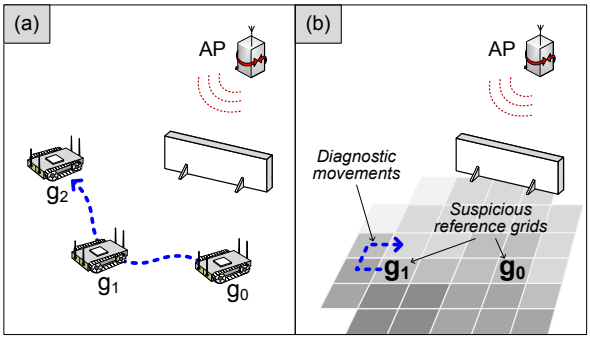


Figure 5.3: *The diagnostic probing upon deployment of a new obstacle*: (a) Sybot periodically performs the selective probing of the reference grids (g_i) to the AP. (b) The diagnostic probing identifies suspicious grids that experience large deviations in spectrum condition, and measures their spectrum condition.

periodically performs the selective probing with respect to the AP. Once it finishes the survey, Sybot finds that reference grids g_0 and g_1 have large deviations from their baseline conditions ($\mathbb{B}$). Next, Sybot starts the complete probing from g_1 to its neighboring grids in the block of g_1 and applies the same method to g_0. Finally, Sybot updates the spectrum map with the measured grids, as shown in Figure 5.3 (b).

5.3.4 Complexity

The algorithms of Sybot incurs only polynomial-time complexity. In the compete and diagnostic probing, Sybot needs to calculate and update the metrics of m measurements within each grid, and its computational complexity is $O(nm)$, where n is the total number of grids. In the diagnostic probing, as explained in Section 5.3.3, Sybot needs to calculate the set of reference grids and incurs $O(nlogn + nm)$ computational complexity per AP, where n is the total number of grids and m the average block size ($\leq n$).

Next, Sybot is scalable in both complexity and energy efficiency with a large number of APs. In the computational complexity $O(nlogn + nm)$, since n is constant[5] per AP, the total computational complexity also grows polynomially with the number of APs. Since the Sybot's energy consumption is mainly governed by the number of grids to be measured and the number is less than or equal to n, Sybot's energy consumption linearly increases

[5]The maximum n is essentially bounded by a maximum transmission range of an AP, which is constant.

with the number of APs.

5.4 System Implementation

Using the hardware prototype that we have built in the previous work (see Figure 4.3), we have implemented Sybot in Linux. Figure 5.4 shows the Sybot's software architecture running in the router, and consists of (1) a mobility control module at the application layer and (2) a spectrum monitoring module at the link layer.

Mobility control module

This module is responsible for controlling a robot's movement and managing the measurement results. This module is implemented in the application layer and is composed of the following components. First, a graphic user interface (GUI) receives/sends survey requirements/reports from/to network engineers. Based on the requirements, GUI initially schedules a spectrum survey. Then, the *scheduler* adaptively triggers complete, selective, or diagnostic probing using the algorithms in Section 5.3.3. During a survey, the *mobility controller* physically moves a robot to a target location and triggers the monitoring module to take measurements.

Upon completion of measurements by the monitoring module, the mobility control module updates a spectrum-condition *map* and schedules the next probing technique, time, and areas. We have also implemented a robot's positioning system using techniques in [66]. The system provides high accuracy ($<$10 cm error), but we omit the details due to the page limit.

Spectrum monitoring module

This module is responsible for measuring spectrum-survey metrics within a target space. Specifically, the module is implemented in an open MADWiFi device driver [68] and is composed of two components: spectrum monitor and filters. When the monitoring module receives a measurement request from the mobility controller (via a socket), the spectrum

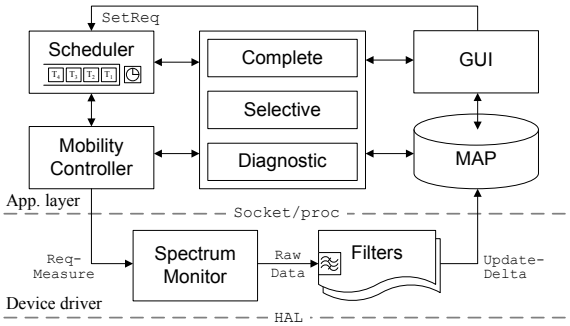


Figure 5.4: *Software architecture of Sybot*: The Sybot software design includes (1) a mobility control module in the application layer and (2) a spectrum monitoring module in the device driver.

monitor starts collecting information on SNR and PDR to APs. Through a hardware abstraction layer (HAL) that Atheros-based chipset [9] provides, the monitor can acquire the above information available in the MAC protocol.

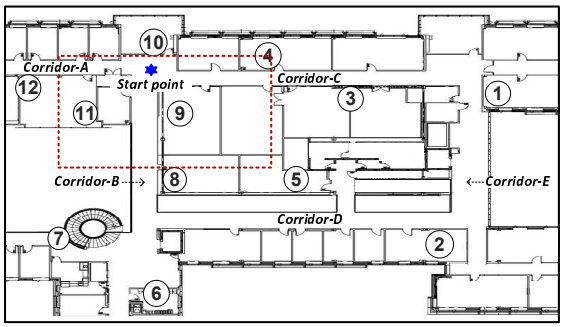
These raw data are then processed through the filters to calculate survey metrics over the target space. Finally, the calculated information is reflected into MAP through a `/proc` interface.

5.5 Performance Evaluation

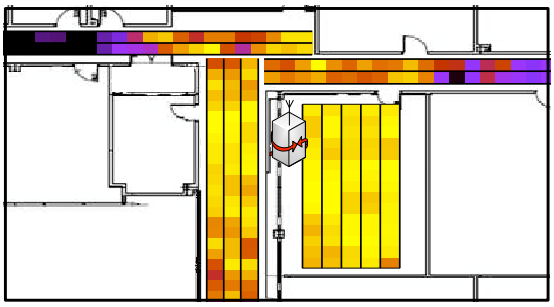
We have evaluated Sybot through extensive experiments and measurement analysis.

5.5.1 Testbed Setup

To evaluate Sybot in an indoor environment, we have deployed 12 IEEE 802.11-based APs with the topology shown in Figure 5.5(a). Each AP is deliberately placed in the ceiling or shelves to cover the entire floor. All APs are equipped with an omni-directional antenna and are operated at the IEEE 802.11a frequencies. Each AP is equipped with an Atheros-based miniPCI NIC and is tuned to use heterogeneous transmission powers of 3–10 dBm so that every location can be covered by 3–4 APs, given limited space.



(a) Topology



(b) Spectrum map

Figure 5.5: *Testbed topology (210 × 110 ft) and a spectrum-condition map*: (a) 12 APs are deployed on the fourth floor of our Department building. (b) Example of a spectrum map constructed by Sybot for AP-9 over three corridors (dotted box in Figure 5.5(a)).

5.5.2 Experiment Methodology

In the testbed, we conducted extensive spectrum surveys using Sybot. Starting from 'start-point' in Figure 5.5(a), Sybot navigates through corridors A, B, C, D, and E and performs the complete, selective, and diagnostic probing with respect to each AP. We ran experiments during the night time when all corridors are accessible.

During the spectrum surveys, we used and tested various time-scales and experimental settings. First, for long-term spectrum measurements, we ran Sybot over selected APs three times a day for 11 days. Next, for short-term measurements, we ran Sybot 5–10 times a day for every AP in our testbed. For each run, Sybot measures 2 to 3 corridors per AP and generates a spectrum map per AP. Finally, we used a 20in×30in rectangle as the minimum grid size to generate a high-resolution spectrum map and analysis. We say that the size of grid i is x if its area is x times larger than that of the minimum-sized grid.

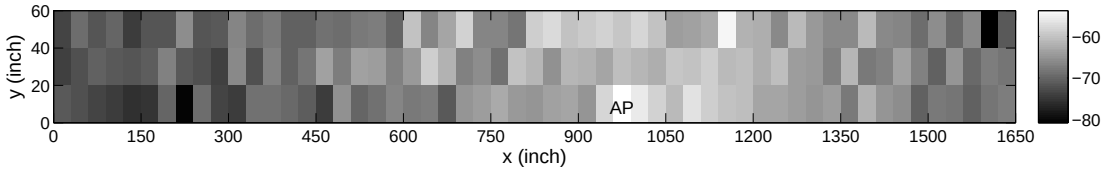


Figure 5.6: *The complete probing results over a long-term period*: The figure shows the average RSS $\overline{\gamma}$ of every grid over AP-3, measured via the complete probing, and reflects real radio propagation over corridors.

5.5.3 Experimental Results and Analysis

5.5.3.1 Repeatability

We first studied the repeatability of Sybot's complete probing. Sybot periodically performs a comprehensive spectrum survey and constructs a baseline spectrum map specific to the surrounding physical environment. To evaluate the repeatability of the complete probing, we randomly selected several APs and analyzed a set of spectrum maps that Sybot built over corridors using the complete probing. Finally, we plot the baseline spectrum map that consists of the average $(\overline{\gamma})$ and standard deviation σ of measured RSSs for each grid.

Figures 5.6 and 5.7 show the constructed baseline spectrum map of $\overline{\gamma}$ and the histogram of σ, respectively. As shown in the figures, the baseline spectrum map accurately represents interesting signal propagation characteristics as follows. First, in Figure 5.6, $\overline{\gamma}$ gracefully diminishes as Sybot moves away from the AP located at (1000, 0), and shows the spatial RSS pattern (or gradient) over distance. This pattern is indeed repeatable in that the RSSs are stable (small σ) over the measured corridors. Figure 5.7 shows the distribution of σ for 5 APs over 4 corridors (894 grids). As shown in the figure, more than 87% grids show a small standard deviation ($<$ 4dBm). We observed that some areas with high σ (> 6 dBm) actually experiences physical changes (e.g., trash cans or doors), which is also captured in the map.

This baseline spectrum map enables Sybot to perform other probing techniques, further improving the performance of spectrum survey. For example, the selective probing can reduce the space to be measured by skipping areas with similar $\overline{\gamma}$. Furthermore, the diagnostic probing can detect abnormal spectrum patterns from the baseline or normal spectrum-condition information.

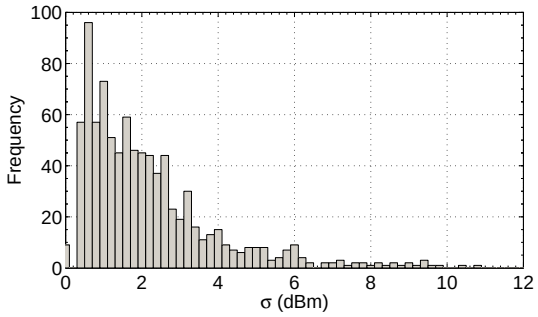


Figure 5.7: *Histogram of the standard deviation σ based on the complete probing results*: More than 87% grids among 894 measured grids show less than 4dBm deviations.

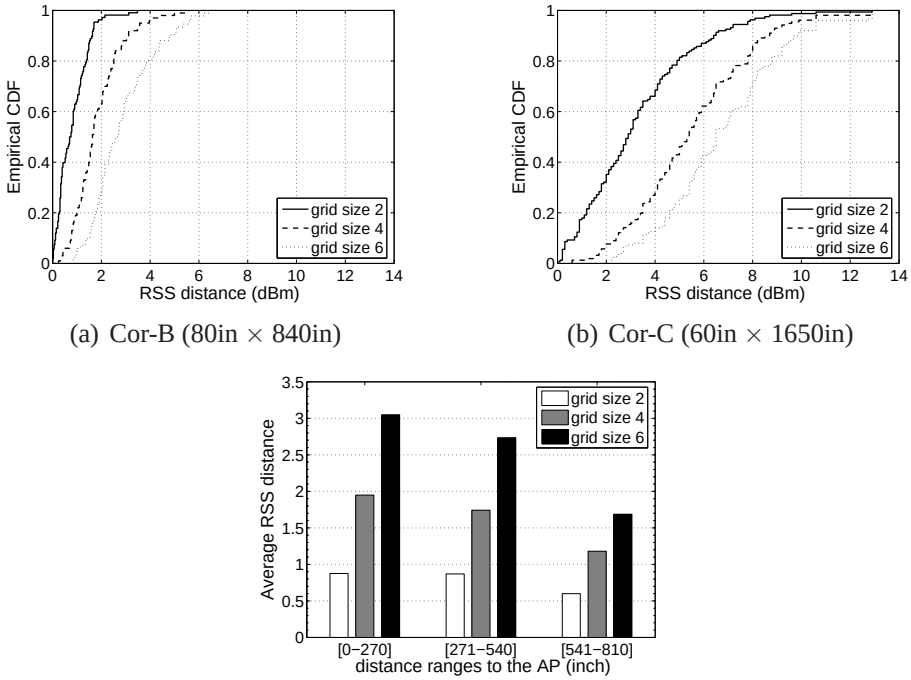
5.5.3.2 Impact of a grid size

Next, we study the impact of a unit grid size on measurement accuracy and efficiency. While a small (fine) grid size provides a high-accuracy spectrum condition map, it incurs significant time and resource overheads. Furthermore, determining the optimal grid size is also challenging due to the spatial heterogeneity in spectrum conditions. To address these issues, we analyzed spectrum maps of different corridors (Cor-B and Cor-C), while varying the grid size. While increasing the grid size in multiples of the minimum size (i.e., 20in$\times$30in), we analyzed how much error the grid size introduces in spectrum surveys. For this analysis, we use the metric, called *RSS distance*, to quantify errors, and the error on grid i, $\gamma_{dist}(i)$ is defined as:

$$\gamma_{dist}(i) = \max_{k \in i}\{\gamma_i(k)\} - \min_{k \in i}\{\gamma_i(k)\}, \tag{5.2}$$

where $\gamma_i(k)$ is the measured RSS at point k within grid i. Given a set of $\gamma_i(k)$ measurements in grid i, the RSS distance calculates difference between the maximum and the minimum RSS values, and thus represents the degree of *heterogeneity* in RSS within the grid. Intuitively, the smaller γ_{dist}, the smaller deviation or error in the set of $\gamma_i(k)$.

We first compare the impact of grid size at different physical sites (corridors, offices). Figures 5.8(a) and 5.8(b) show the empirical cumulative distributions (CDFs) of γ_{dist} for three different grids sizes (i.e., size 2, 4, 6) over Cor-B and Cor-C. As shown in the figures, CDFs grow faster with small-sized grids, indicating better measurement accuracy.



(a) Cor-B (80in $\times$ 840in)

(b) Cor-C (60in $\times$ 1650in)

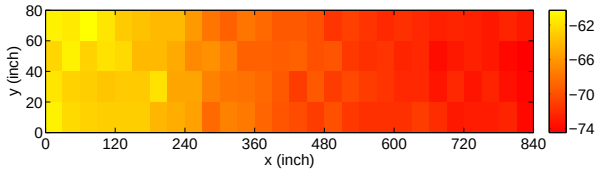
(c) Distance (Cor-B)

Figure 5.8: *Impact of grid size on the achievable accuracy of complete probing*: (a-b) The accuracy is measured in terms of the RSS distance under various grid sizes, i.e., 2, 4 and 6, at two different corridors. The figures indicate that the grid size must be adaptively chosen for different measurement spaces. (c) The average RSS distance decreases as the distance from the transmitter (AP) increases.

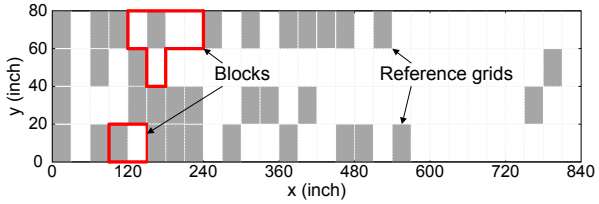
This confirms our expectation that a larger grid size introduces a greater measurement error. In addition, for each corridor, different grid sizes introduce different impacts on the measurement error.

Next, we compare the impact of grid size with different distances from an AP. Figure 5.8(c) shows the average RSS distance of three grid sizes at three different distance zones from AP-10. Given a grid size, the average RSS distance is shown to decrease as the distance from the AP increases. This is because that RSS changes are more dynamic in a close proximity of the AP than the areas far away from it.

Therefore, the grid size for the complete probing should be carefully selected, depending on the physical site and distance to an AP. Using the complete probing results, Sybot can build a profile that estimates the impact of each grid size on site-specific spectrum characteristics. Furthermore, because of this non-uniformity of the characteristics, Sybot can apply the selective and diagnostic probing techniques to improve efficiency and accuracy.



(a) Spectrum map $\mathbb{B}(\gamma)$



(b) Reference grids

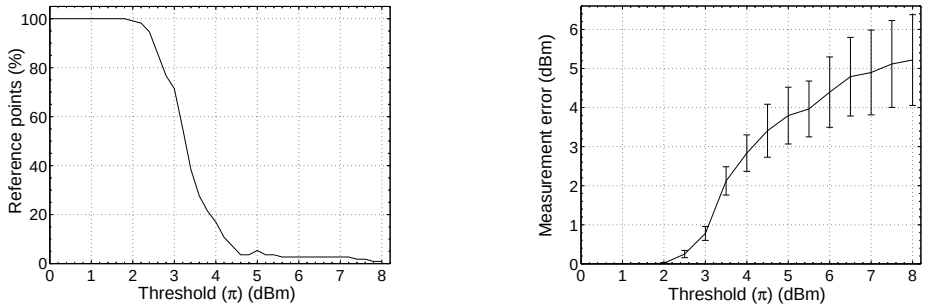
Figure 5.9: *Selection of reference points in selective probing (Cor-B)*: Reference points are optimally chosen so as to cover the measurement with a minimum number of blocks.

5.5.3.3 Reducing the space to measure

Now, we evaluate the effectiveness of Sybot's selective probing in reducing the measurement space during a spectrum survey. As discussed in the previous experiment, the spectrum condition is very heterogeneous over space. This spatially-heterogeneous spectrum conditions can be accurately obtained through the complete probing with the minimum grid size. Extracting these spectrum characteristics can be very useful in several cases, such as capturing the temporal variance of the areas or saving the survey resource/time by making only a small number of measurements. However, such an exhaustive spectrum survey requires excessive overheads, thus degrading the spectrum survey efficiency.

The selective probing reduces the overheads by identifying areas with similar spectrum conditions and merging them together as a unit measurement block. Therefore, the main questions that the selective probing has to answer are: (1) how dense or large the blocks are, (2) how to make a trade-off between efficiency and accuracy, and (3) how much of benefit the selective probing can achieve over the complete probing.

First, to see the blocks formed by the selective probing, we ran Sybot over Cor-A with the tolerance threshold π = 3.5dBm, and plot the spectrum map constructed using the results of the complete probing, and the reference grids chosen by the selective probing in Figure 5.9. Figure 5.9(a) shows clear and comprehensive spectrum propagation characteristics



(a) Efficiency (b) Accuracy

Figure 5.10: *Reducing the measurement space and the resultant tradeoff*: The selective probing minimizes the measurement efforts by reducing the reference points (top figure), at the cost of measurement accuracy (bottom figure).

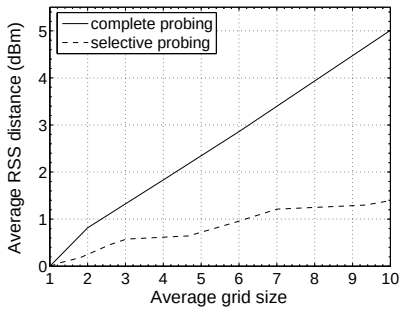


Figure 5.11: *Performance comparison of complete vs. selective probing*: The selective probing reduces the measurement space by up to 72 % with grid size of 10.

over distance. One can also observe that the spectrum conditions in close proximity of the AP, located at (0, 40), are diverse, whereas those far away from the AP is almost monotonic. This spectrum heterogeneity is exploited in selecting the reference grids by the selective probing (Figure 5.9(b)). The map shows that the reference grids are densely distributed near the AP, because of large spatial and temporal variations in RSSs, while they are sparsely placed where the signal is out of reach.

Next, we study the tradeoff between efficiency and accuracy of the selective probing. If the tolerance threshold (π) is increased in determining a block, then Sybot reduces the number of measurements (or reference grids) at the cost of measurement accuracy. To show this tradeoff, we ran the selective probing for Cor-B with the spectrum map constructed based on the results of the complete probing and derives how much Sybot can reduce the measurement space. Figure 5.10(a) shows that as the threshold π (in dBm) increases, the selective probing becomes more *aggressive* in merging grids, thus reducing the number of reference

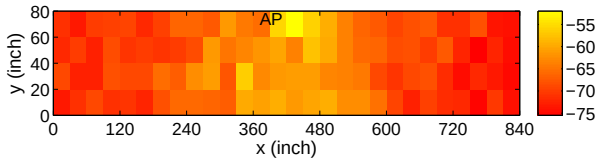
grids. For example, when $\pi = 3.5$ (dBm), the number of reference grids can be reduced by 70%, compared to the complete probing results. Figure 5.10(b) shows, on the other hand, that the measurement errors increase as the threshold π increases. This is because large threshold π allows the selective probing to merge less similar grids, thus degrading the measurement accuracy. Therefore, Figure 5.10 shows a clear tradeoff between the measurement effort and accuracy, which can be used as a guideline in spectrum measurement planning.

Third, to study the advantage of the selective over the complete probing, we compare the average RSS distance (γ_{dist}) achieved by the complete and selective probing on the measurement data from Cor-B. Figure 5.11 shows γ_{dist} as a function of average grid size. The figure indicates that the average RSS distance increases almost linearly with the complete probing, while the distance remains below 1.5 dBm under the selective probing. This advantage comes from its ability to incorporate site-specific spectrum conditions in determining the measurement grids.

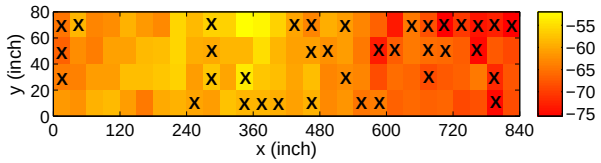
5.5.3.4 Diagnosing abnormal spectrum condition

We now study the effectiveness of Sybot in detecting abnormal changes in local spectrum conditions. When a network faces such changes in local areas, the diagnostic probing identifies the boundary of such areas and incrementally updates their spectrum conditions only, as opposed to the entire coverage area of local APs. By using both the selective probing results (for detection) and the baseline spectrum map (for range estimation), the diagnostic probing can efficiently maintain an up-to-date spectrum map. To evaluate its efficiency, we placed one obstacle in the middle of Cor-C (480, 80) next to the AP (400, 80). Then, we ran the complete probing without the obstacle to obtain a baseline spectrum map and ran both complete and diagnostic probing with the obstacle.

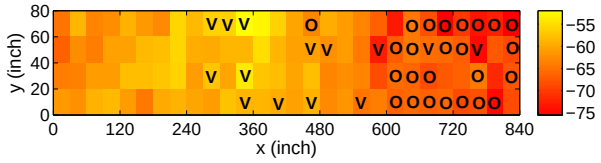
Figure 5.12 shows the above three measurement results and demonstrates the improvement of the diagnostic probing. The probing result without the obstacle (Figure 5.12(a)) appears as regular radio propagation from the AP. However, the complete probing result with the obstacle shows clear effects of the obstacle and includes a large deviation in spectrum condition in the right side of the corridor. Grids with 'X' show large deviations in



(a) Spectrum map $\mathbb{B}(\gamma)$



(b) Areas with abnormal conditions



(c) Diagnosis and update results

Figure 5.12: *Example of the diagnostic probing with an obstacle*: (a-b) The appearance of an obstacle (next to AP) causes abnormal changes in spectrum conditions (denoted as X in Figure 5.12(b)); (c) The diagnostic probing identifies the boundaries of areas with abnormal changes with fewer measurements.

their spectrum condition (i.e., for each grid, $diff_i$=$|\gamma_i - \overline{\gamma}_i| > 2.5 \times \sigma_i$), due to the obstacle, and their boundaries must be identified. Because the diagnostic probing uses the selective probing results, Sybot can quickly find the subset of reference grids that experience the large deviations (denoted as 'V' in Figure 5.12(c)). Then, Sybot incrementally updates the reference grids in the subset ('O'). As shown in the figure, the diagnostic probing successfully estimates the problem areas, while reducing its survey space by 56%, compared to the exhaustive probing.

5.6 Conclusion

We first make concluding remarks and then discuss some of the remaining issues associated with Sybot.

5.6.1 Concluding Remarks

This chapter presented Sybot, a robot-based spectrum site-survey system, that autonomously provides an accurate spectrum-condition map for wireless networks. In addition to its automation in a labor-intensive spectrum survey, Sybot significantly reduces measurement overheads by adaptively using three complementary probing techniques. Furthermore, Sybot also provides important control knobs for determining a trade-off between accuracy and efficiency in spectrum surveys. Our experimental evaluation results show that Sybot reduces the measurement effort (e.g., the number of measurements) by more than 56%, compared to a traditional exhaustive survey. Moreover, our in-depth analysis of the measurement data has yielded several important guidelines for adjusting important survey parameters.

5.6.2 Remaining Issues

Mobility limitations: Although the current Sybot prototype has basic mobility, a robot can have more sophisticated mobility, such as climbing stairs or obstacles, depending on target environments, and Sybot can be served as a general framework to support them as well.

Use of multiple robots: Multiple Sybot nodes can cooperate to reduce the spectrum-survey time. For example, Sybot nodes on different floors of a building can collaborate to construct a building-wide three-dimensional spectrum map. This is part of our future work.

Antenna height: Sybot is equipped with a omni-directional antenna of 10-inch height from the floor. Although its measurement results demonstrate the effectiveness of Sybot, we plan to study the effects of the height or the types of antenna on the survey results.

CHAPTER 6

Conclusions and Future Work

This thesis focused on improving QoS support and the autonomous management capability of multi-hop wireless networks. Despite the promises of fast, easy, and inexpensive wireless access to the Internet, multi-hop networks often provide poor QoS and unexpected management costs, due mainly to time-varying and heterogeneous wireless link-conditions. This thesis has discussed the importance of link-quality awareness on QoS and management in multi-hop wireless networks. Then, the thesis has presented novel *monitoring* and *adaptation* techniques, through which the networks can satisfy their initial promises.

In what follows, the primary research contributions of the thesis are summarized, and future extensions/directions are discussed.

6.1 Primary Contributions

This thesis makes the following contributions toward QoS-adaptive self-managing multi-hop wireless networks.

- **Accurate link-quality monitoring**: Due to the use of an open wireless medium, wireless link-quality fluctuates with time. This link-quality fluctuation greatly affects the performance and the design of multi-hop wireless networks. To mitigate the fluctuation problem, this thesis has presented an efficient and accurate link-quality monitor, called EAR. Designed as a distributed monitoring system, EAR is equipped with hybrid measurement techniques that exploit existing data traffic in measurement

and yield accurate link-quality information. In this work, EAR has been implemented in Linux and evaluated in an indoor wireless mesh network testbed. The evaluation results of the implementation show that EAR improves measurement accuracy up to 12 times, over existing measurement techniques, and further helps routing protocols achieve improved QoS (e.g., end-to-end throughput improvement more than 100%).

- **Localized network reconfiguration**: Because of time-varying environmental factors (e.g., interference) on wireless link-conditions, multi-hop networks often experience performance degradation or QoS failures. To recover from such failures, multi-hop wireless networks have to be able to reconfigure network settings with minimum QoS disruption. To this end, this thesis has proposed a localized network reconfiguration algorithm, called LEGO. Designed for a multi-radio WMN, LEGO makes use of multi-channel and multi-radio diversities for recovering from channel-related link failures. By cooperatively reconfiguring channel assignment or radio association among local mesh routers, LEGO enables a multi-radio WMN to continue meeting user QoS demands. LEGO has been implemented and deployed in a multi-radio WMN testbed at the University of Michigan. The evaluation results show that LEGO outperforms network-level reconfiguration schemes in channel efficiency ($>$86%) as well as in QoS-satisfiability.

- **Dynamic (re-)formation of wireless relay networks**: Network nodes need to be (re-)located to satisfy QoS requirements under heterogeneous and changing physical environments. In particular, real-time deployment of network nodes with QoS constraints requires significant management efforts to manually tune node positions. This thesis has proposed a mobile autonomous router system, called MARS. Using robot-based mobility, MARS includes a spatial link-quality monitoring protocol and a spatial probing algorithm that allow a wireless router to selectively and progressively measure-and-navigate target areas for finding its optimal positions. In addition, MARS enables a group of nodes to cooperatively form a multi-hop relay network or to adjust their placements in response to the changes in QoS requirements or spectrum conditions. MARS has been prototyped and implemented using

open-source software and off-the-shelf products. Our evaluation results on the prototype show that MARS successfully identifies locally optimal positions that satisfy the QoS requirements. At the same time, it reduces measurement overhead by two thirds, compared to the exhaustive measurement strategy.

- **Spectrum site-survey automation**: Even after the deployment of wireless networks, network operators need to periodically monitor spatial spectrum usage over entire deployment areas, which incur labor-intensive management costs. This thesis has proposed a spectrum survey robot, called Sybot. Using a mobile robot, Sybot automates labor-intensive site-survey and further characterizes various spectrum-survey parameters that determine accuracy and efficiency in spectrum survey. Sybot is equipped with three spectrum probing techniques that allow for the collection of repeatable spatial spectrum-condition maps as its spatiotemporal variance by measuring only a subset of entire measurement space. Sybot has been implemented and evaluated with 12 APs in challenging indoor environments. Its evaluation results show that Sybot autonomously generates repeatable spectrum maps and further reduces survey overheads more than 65%, compared to traditional spectrum survey methods.

6.2 Future Work

This thesis work on improving the QoS and management cost of multi-hop wireless networks can be extended further as follows.

- **QoS gateway for real-time applications**: In Chapter 2, EAR has been proposed to provide accurate link-quality information, and this information can be used for various network applications and services. As new real-time applications (e.g., VoIP) are being introduced, WMNs need a controller that determines the admission of new real-time traffic (e.g., voice call, video sessions) to provision or maintain network QoS. In future, we plan to develop such a QoS controller or QoS gateway that makes use of EAR's link-quality information and manages network resources, especially for emerging real-time applications.

- **Channel reconfiguration over dynamic spectrum access networks**: In Chapter 3, LEGO has been proposed to dynamically reconfigure network settings in multi-radio WMNs. With the recent advance in dynamic spectrum access (DSA) [39] and cognitive radio [46] technologies, WMNs can adopt them to improve network capacity. However, DSA-based networks inherently face dynamic spectrum availability, and requires efficient and real-time channel reconfiguration and coordination systems for QoS support. In future, we would like to extend LEGO to cope with dynamic spectrum availability, especially in cognitive-radio-based WMNs.

- **Supporting complex topologies using mobile wireless routers**: In Chapter 4, MARS has been proposed to form a string-type wireless relay network. In addition to the string topology, supporting complex topologies such as tree and mesh could be interesting, but is challenging due mainly to its complexity. In future, we would like to develop heuristic spatial probing algorithms that allow MARS to optimize node positions, with respect to multiple neighboring nodes, to dynamically (re-)form various and complex network topologies.

- **Spatial spectrum aggregation**: In Chapter 5, Sybot has been proposed for automatic spectrum site-survey, and the survey results can be used for improving spectrum utilization. Because today's wireless networks are being redundantly deployed in many places, Sybot can aggregate such multiple network connectivities to improve QoS. By using spectrum survey results, Sybot can easily identify locations that are covered by multiple APs. In future, we would like to design and build a spectrum aggregation system that opportunistically uses one or multiple APs to achieve aggregated bandwidth as well as improve spectrum utilization.

APPENDICES

APPENDIX A

Link Capacity Estimation

Although there are a considerable number of analytical models for link capacity [13, 21, 22, 114], LEGO uses a simple and realistic model based on both MAC behavior and actual measurement results similar to the derivation in [64]. Link capacity (C) can be derived by estimating the expected packet transmission latency (t_l), which consists of back-off time (t_b) and actual transmission time (t_s). First, the initial value of t_b is determined based on a uniformly-chosen random value in [0, CWMin]. On average, t_b is $\frac{CWMin \times slotTime}{2}$. However, because the window size exponentially increases every time the transmission attempt fails, t_b^i becomes $2^i \times \frac{CWMin \times slotTime}{2}$ on i consecutive failures.

Next, the packet-transmission time t_s includes inter-frame time (i.e., SIFS, DIFS) and RTS/CTS, DATA/ACK frame transmission time. While inter-frame times and control/ACK frames consume a fixed amount of time, a data frame takes different amounts of time due to different transmission rates (c). Thus, t_s can be represented as $\frac{RTS+CTS+ACK}{2.0 \times 10^6} + \frac{DATA}{c} + 3SIFS + DIFS$.

Given t_b, t_s, and the measured data-delivery ratio d, the expected latency or t_l for a packet transmission can be derived as:

$$t_l = \sum_{i=0}^{n_r} (1-d)^i \times d \times \{t_b^i + (i+1) \times t_s\}, \quad C = \frac{DATA}{t_l} \tag{A.1}$$

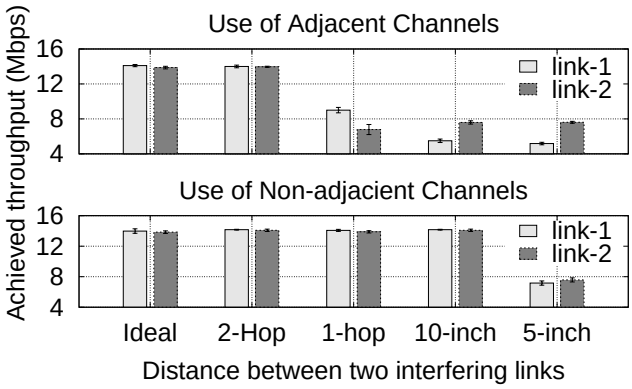
where n_r is a retry limit at the MAC layer, and d and c are obtained from LEGO's monitoring protocol.

APPENDIX B

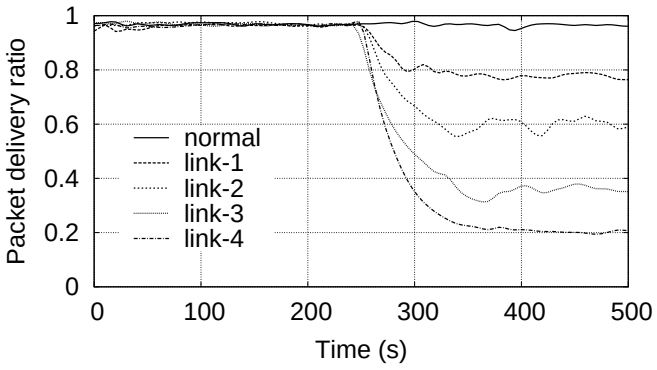
Interference Generation Method

To generate interference on a target channel, we use one laptop that generates one UDP flow and whose radio is tuned to the channel adjacent to the target channel in IEEE 802.11a. Even though IEEE 802.11a uses the OFDM modulation scheme in PHY for orthogonal channels, adjacent channels interfere with each other, especially when they are used within a certain range. To study this effect on network performance, we measure the achieved throughput of two different flows while increasing the distance between the two flows' transmitters. Fig.B.1(a) shows that if two adjacent channels (e.g., 5.18 and 5.2 Ghz) are used within one hop, their throughput degrades up to 75%. On the other hand, the use of non-adjacent channels (e.g., 5.18 and 5.22 Ghz) does not cause such channel interference.

Using this fact, we generate various levels of interference on network performance. One interference node generates garbage traffic at different rates in the channel adjacent to a target channel within one-hop range. While generating the interference, we measure the packet-delivery ratio of one UDP flow on the selected target channel. As shown in Fig. B.1(b), different degrees of channel-related failures (i.e., link-quality degradation), and this scheme is used for injecting channel faults throughout the evaluation of LEGO.



(a) Effect of adjacent channel



(b) Interference levels

Figure B.1: *Interference Generation*

APPENDIX C

The trajectory fitting

Using a few "good" points that are manually measured, the entire trajectory of the robot can be fitted so that the trajectory satisfies two goals: (i) the trajectory passes through good points and (ii) the trajectory is close to the length/angle of each leg from the odometry as much as possible. This trajectory fit is obtained using the following optimization:

$$\min [\sum_{i=1}^{n} d_{err}(i, i+1) + K \sum_{i=2}^{n-1} a_{err}(i-1, i, i+1)] \tag{C.1}$$

where $d_{err}(i, i+1)$ is the absolute difference between the length of leg i obtained from the odometry and the length imposed by the fit; $a_{err}(i-1, i, i+1)$ is the absolute error in angle at measurement stop i, namely, the angle between segments $(i-1, i)$ and $(i, i+1)$; K is a constant that modifies the relative weight of preserving angles versus preserving distances from the original drive. The procedure is implemented using the function *fmins* in octave-forge.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] A. Adya, P. Bahl, R. Chandra, and L. Qiu. Architecture and techniques for diagnosing faults in IEEE 802.11 infrastructure networks. In *Proceedings of ACM MobiCom*, Philadelphia, PA, September 2004.

[2] A. Adya, P. Bahl, J. Padhye, A. Wolman, and L. Zhou. A multi-radio unification protocol for IEEE 802.11 wireless networks. In *Proceedings of IEEE BroadNets*, San Jose, CA, October 2004.

[3] D. Aguayo, J. Bicket, S. Biswas, G. Judd, and R. Morris. Link-level measurements from an 802.11b mesh network. In *Proceedings of ACM SIGCOMM*, Portland, OR, August 2004.

[4] AirMagnet, Inc. http://www.airmagnet.com, 2002.

[5] A. Akella, G. Judd, S. Seshan, and P. Steenkiste. Self-management in chaotic wireless deployments. In *Proceedings of ACM MobiCom*, Cologne, Germany, September 2005.

[6] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *IEEE Communication Magazine*, pages 102–114, 2002.

[7] I. Akyildiz, X. Wang, and W. Wang. Wireless mesh networks: A survey. *Computer Networks*, pages 445–487, 2005.

[8] M. Alicherry, R. Bhatia, and L. Li. Joint channel assignment and routing for throughput optimization in multi-radio wireless mesh networks. In *Proceedings of ACM MobiCom*, Cologne, Germany, August 2005.

[9] Atheros Communications. http://www.atheros.com.

[10] Aware Home Research Initiative at Georgia Institute of Technology. http://awarehome.imtc.gatech.edu, 2000.

[11] P. Bahl, R. Chandra, and J. Dunagan. SSCH: Slotted seeded channel hopping for capacity improvement in IEEE 802.11 ad-hoc wireless networks. In *Proceedings of ACM MobiCom*, Philadelphia, PA, September 2004.

[12] Paramvir Bahl and Venkata N. Padmanabhan. Radar: An in-building RF-based user location and tracking system. In *INFOCOM (2)*, pages 775–784, 2000.

[13] G. Bianchi. Performance analysis of the IEEE 802.11 distributed coordination function. *IEEE JSAC*, 18(3), March 2000.

[14] S. Biswas and R. Morris. Opportunistic routing in multi-hop wireless networks. In *Proceedings of the Second Workshop on Hot Topics in Networks (HotNets-II)*, Cambridge, MA, November 2003.

[15] S. Biswas and R. Morris. ExOR: Opprotunistic multi-hop routing for wireless networks. In *Proceedings of ACM SIGCOMM*, Philadelphia, PA, August 2005.

[16] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin. Resource reservation protocol (rsvp). Internet Request for Comments 2205 (rfc2205.txt), September 1997.

[17] A. Brzezinski, G. Zussman, and E. Modiano. Enabling distributed throughput maximization in wireless mesh networks- a partitioning approach. In *Proceedings of ACM MobiCom*, Los Angeles, CA, September 2006.

[18] M. Buddhikot, P. Kolodzy, S. Miller, K. Ryan, and J. Evans. DIMSUMnet: New directions in wireless networking using coordinated dynamic spectrum access. In *Proceedings of IEEE Symposium on a World of Wireless Mobile and Multimedia Networks*, Naxos, Italy, June 2005.

[19] J. Camp, J. Robinson, C. Steger, and E. Knightly. Measurement driven deployment of a two-tier urban mesh access network. In *Proceedings of ACM MobiSys*, Uppsala, Sweden, June 2006.

[20] J. Campbell, R. Sukthankar, I. Nourbakhsh, and A. Pahwa. A robust visual odometry and precipice detection system using consumer-grade monocular vision. In *Proceedings of IEEE International Conference on Robotics and Automation*, Barcelona, Spain, April 2005.

[21] M. M. Carvalho and J. J. Garcia-Luna-Aceves. Delay analysis of IEEE 802.11 in single-hop networks. In *Proceedings of IEEE ICNP*, Atlanta, GA, November 2003.

[22] M. M. Carvalho and J. J. Garcia-Luna-Aceves. A scalable model for channel access protocols in multihop ad hoc networks. In *Proceedings of ACM MobiCom*, Philadelphia, PA, September 2004.

[23] A. Cerpa, J. L. Wong, M. Potkonjak, and D. Estrin. Temporal properties of low power wireless links: Modeling and implications on multi-hop routing. In *Proceedings of ACM MobiHoc*, Urbana-Champaign, IL, May 2005.

[24] Ranveer Chandra, Lili Qiu, Kamal Jain, and Mohammad Mahdian. Optimizing the placement of integration points in multi-hop wireless networks. In *Proceedings of IEEE ICNP*, Berlin, Germany, October 2004.

[25] S. Chellappan, X. Bai, B. Ma, and D. Xuan. Sensor networks deployment using flip-based sensors. In *Proceedings of IEEE MASS*, Washington, DC, November 2005.

[26] S. Chen and K. Nahrstedt. Distributed quality-of-service routing in ad hoc networks. *IEEE JSAC*, 17(8):1488–1505, 1999.

[27] Y.-C. Cheng, J. Bellardo, P. Benko, A. C. Snoeren, G. M. Voelker, and S. Savage. Jigsaw: Solving the puzzle of enterprise 802.11 analysis. In *Proceedings of ACM SIGCOMM*, Pisa, Italy, September 2006.

[28] M. Chiang, S. H. Low, A. R. Calderbank, and J. C. Doyle. Layering as optimization decomposition: A mathematical theory of network architectures. *IEEE*, 95(2), January 2007.

[29] D. D. Clark, C. Partridge, J. C. Ramming, and J. T. Wroclawski. A knowledge plane for the internet. In *Proceedings of ACM SigComm*, Karlsruhe, Germany, August 2003.

[30] D. S.J. De Couto, D. Aguayo, J. Bicket, and R. Morris. A high-throughput path metric for multi-hop wireless routing. In *Proceedings of ACM MobiCom*, San Diego, CA, September 2003.

[31] D. S.J. De Couto, D. Aguayo, B. A. Chambers, and R. Morris. Performance of multi-hop wireless networks: Shortest path is not enough. In *Proceedings of the First Workshop on Hot Topics in Networks (HotNets-I)*, Princeton, New Jersey, October 2002.

[32] DARPA/IPTO. LANdroids: Autonomous, expendable, threat detection and locating system. http://www.darpa.mil/ipto.

[33] P. De, R. Krishnan, A. Raniwala, K. Tatavarthi, N. A. Syed, J. Modi, and T. Chiueh. Mint-m: An autonomous mobile wireless experimentation platform. In *Proceedings of ACM MobiSys*, Uppsala, Sweden, June 2006.

[34] C. Dovrolis, P. Ramanathan, and D. Moore. What do packet dispersion techniques measure? In *Proceedings of IEEE InfoCom*, Anchorage, AK, April 2001.

[35] A. B. Downey. Using pathchar to estimate Internet link characteristics. In *Proceedings of ACM SIGCOMM*, Cambridge, MA, September 1999.

[36] R. Draves, J. Padhye, and B. Zill. Routing in multi-radio, multi-hop wireless mesh networks. In *Proceedings of ACM MobiCom*, Philadelphia, PA, September 2004.

[37] R. Draves, J. Padhye, and B. Zill. Routing in multi-radio, multi-hop wireless mesh networks. In *Proceedings of ACM MobiCom*, Philadelphia, PA, September 2004.

[38] J. Eriksson, S. Agarwal, P. Bahl, and J. Padhye. Feasibility study of mesh networks for all-wireless offices. In *Proceedings of ACM MobiSys*, Uppsala, Sweden, June 2006.

[39] Federal Communications Commission (FCC). Facilitating opportunities for flexible, efficient, and reliable spectrum use employing cognitive radio technologies. In *FCC Document ET Docket No. 03-108*, December 2003.

[40] Bill Gates. A robot in every home. *Scientific American*, pages 58–65, January 2007.

[41] Gecko Systems. http://www.geckosystems.com, 2008.

[42] D. Giustiniano and G. Bianchi. Are 802.11 link quality broadcast measurements always reliable? In *Proceedings of ACM CoNEXT*, Lisboa, Portugal, December 2006.

[43] T. Goff, N. B. Abu-Ghazaleh, D. S. Phatak, and R. Kahvecioglu. Preemptive routing in ad hoc networks. In *Proceedings of ACM MobiCom*, pages 43–52, 2001.

[44] J. L. Gross and J. Yellen. Graph theory and its applications, 2nd edition. Chapman & Hall/CRC, 2006.

[45] M. Grossglauser and D. N. C. Tse. Mobility increases the capacity of adhoc wireless networks. *IEEE Transactions on Information Theory*, 10(4), August 2002.

[46] S. Haykin. Cognitive radio: Brain-empowered wireless communications. *IEEE Journal on Selected Areas in Communications(JSAC)*, 23(2), February 2005.

[47] C. Hedrick. Routing information protocol. Internet Request for Comments 1058 (rfc1058.txt), June 1988.

[48] T. Henderson, D. Kotz, and I. Abyzov. The changing usage of a mature campus-wide wireless network. In *Proceedings of ACM MobiCom*, Philadelphia, PA, September 2004.

[49] C. C. Ho, K. N. Ramachandran, K. C. Almeroth, and E. M. Belding-Royer. A scalable framework for wireless network monitoring. In *Proceedings of ACM WMASH*, Philadelphia, PA, October 2004.

[50] IEEE 802.11, Wireless LAN medium access control (MAC) and physical layer (PHY) specifications. Standard, IEEE, August 1999.

[51] Iperf. Network measurement tool. http://dast.nlanr.net/Projects/Iperf/.

[52] iRobot ConnectR, Virtual Visiting Robot. http://www.irobot.com/sp.cfm?pageid=338, 2008.

[53] iRobot Corp. http://www.irobot.com.

[54] iRobot Roomba, Cleaning Robot. http://www.irobot.com, 2008.

[55] A. Jain, D. Qiao, and K. G. Shin. RT-WLAN: A soft real-time extension to the orinoco linux device driver. In *Proceedings of IEEE PIMRC*, Beijing, China, September 2003.

[56] D. B. Johnson and D. A. Maltz. Dynamic source routing in ad hoc wireless networks. In *the Book of Mobile Computing*, volume 353. Kluwer Academic Publishers, 1996.

[57] K.-H. Kim and K. G. Shin. On accurate measurement of link quality in multi-hop wireless mesh networks. In *Proceedings of ACM MobiCom*, Los Angeles, CA, September 2006.

[58] K.-H. Kim and K. G. Shin. Accurate and asymmetry-aware measurement of link quality in wireless mesh networks. *To appear in IEEE/ACM Transactions on Networking*, 2009.

[59] M. Kodialam and T. Nandagopal. Characterizing the capacity region in multi-radio multi-channel wireless mesh networks. In *Proceedings of ACM MobiCom*, Cologne, Germany, August 2005.

[60] D. Kotz, C. Newport, R. .S. Gray, J. Liu, Y. Yuan, and C. Elliott. Experimental evaluation of wireless simulation assumptions. Technical Report TR2004-507, Dept. of Computer Science, Dartmouth College, June 2004.

[61] P. Kyasanur and N. Vaidya. Capacity of multi-channel wireless networks: Impact of number of channels and interfaces. In *Proceedings of ACM MobiCom*, Cologne, Germany, August 2005.

[62] Andrew M. Ladd, Kostas E. Bekris, Algis Rudys, Lydia E. Kavraki, and Dan S. Wallach. Robotics-based location sensing using wireless Ethernet. *Wireless Networks*, 11(1–2):189–204, January 2005.

[63] H. Lee, S. Kim, O. Lee, S. Choi, and S. J. Lee. Available bandwidth-based association in ieee 802.11 wireless lans. In *Proceedings of ACM MSWiM*, Vancouver, Canada, October 2008.

[64] S. Lee, S. Banerjee, and B. Bhattacharjee. The case for a multi-hop wireless local area network. In *Proceedings of IEEE InfoCom*, Hong Kong, March 2004.

[65] S. Lee, B. Bhattacharjee, and S. Banerjee. Efficient geographic routing in multihop wireless networks. In *Proceedings of ACM MobiHoc*, Urbana-Champaign, IL, May 2005.

[66] J. J. Leonard and H. F. Durrant-Whyte. Mobile robot localization by tracking geometric beacons. *IEEE Transactions on Robotics and Automation*, pages 376–382, June 1991.

[67] Q. Li and D. Rus. Sending messages to mobile users in disconnected ad-hoc wireless networks. In *Proceedings of ACM MobiCom*, Boston, MA, August 2000.

[68] MADWiFi. http://www.madwifi.org.

[69] B. M. Maggs. Asymmetric wireless networks. http://www-2.cs.cmu.edu/ bmm/ wireless.html.

[70] R. Mahajan, M. Rodrig, D. Wetherall, and J Zahorjan. Analyzing the mac-level behavior of wireless networks in the wild. In *Proceedings of ACM SIGCOMM*, Pisa, Italy, September 2006.

[71] M. J. Marcus. Real time spectrum markets and interruptible spectrum: New concepts of spectrum use enabled by cognitive radio. In *Proceedings of IEEE Symposium on New Frontiers in Dynamic Spectrum Access Networks*, Baltimore, MD, November 2005.

[72] MetaGeek, LLC. http://www.metageek.com.

[73] MIT Roofnet. http://www.pdos.lcs.mit.edu/roofnet.

[74] Motorola Inc. Mesh Broadband. http://www.motorola.com/mesh.

[75] J. Moy. OSPF version 2. Internet Request for Comments 2328 (rfc2328.txt), April 1998.

[76] MuniWireless LLC. http://www.muniwireless.com, 2008.

[77] S. Nelakuditi, S. Lee, Y. Yu, J. Wang, Z. Zhong, G.H. Lu, and Z.L. Zhang. Blacklist-aided forwarding in static multihop wireless networks. In *Proceedings of IEEE SECON*, Santa Clara, CA, September 2005.

[78] A. Nesovic, N. Neskovic, and D. Paunovic. Macrocell electric field strength prediction model based upon artificial neural networks. *IEEE JSAC*, 40(6), August 2002.

[79] Netfilter. http://www.netfilter.org.

[80] Newbury Networks. http://www.newburynetworks.com, 2007.

[81] A. J. Nicholson and B. D. Noble. Breadcrumbs: forecasting mobile connectivity. In *Proceedings of the ACM MobiCom*, San Francisco, CA, September 2008.

[82] *ns-2* Network Simulator. http://www.isi.edu/nsnam/ns.

[83] C. Perkins, E. Belding-Royer, and S. Das. Ad-hoc on-demand distance vector routing. Internet Request for Comments 3561 (rfc3561.txt), July 2003.

[84] C. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In *ACM SIGCOMM*, pages 234–244, London, UK, September 1994.

[85] D. M. Pierce. Map learning with uninterpreted sensors and effectors. *Doctoral dissertation, Department of Computer Sciences, The University of Texas at Austin*, 1995.

[86] Nissanka Bodhi Priyantha, Anit Chakraborty, and Hari Balakrishnan. The cricket location-support system. In *ACM MOBICOM*, Boston, MA, August 2000.

[87] Linux-WLAN Project. http://www.linux-wlan.com/.

[88] L. Qiu, P. Bahl, A. Rao, and L. Zhou. Troubleshooting multi-hop wireless networks. In *Proceedings of ACM SigMetrics (extended abstract)*, Alberta, Canada, June 2005.

[89] K. Ramanchandran, E. Belding-Royer, and M. Buddhikot. Interference-aware channel assignment in multi-radio wireless mesh networks. In *Proceedings of IEEE InfoCom*, Barcelona, Spain, April 2006.

[90] A. Raniwala and T. Chiueh. Architecture and algorithms for an IEEE 802.11-based multi-channel wireless mesh network. In *Proceedings of IEEE InfoCom*, Miami, FL, March 2005.

[91] A. Raniwala and T. Chiueh. Architecture and algorithms for an IEEE 802.11-based multi-channel wireless mesh network. In *Proceedings of IEEE InfoCom*, Miami, FL, March 2005.

[92] A. Raniwala and T.-C. Chiueh. Deployment issues in enterprise wireless lans. Technical Report TR-145, Technical Report, Experimental Computer Systems Lab, State University of New Work, September 2003.

[93] T. S. Rappaport. Wireless communications: Principles and practice. Second Edition, Pearson Education, December 2001.

[94] T. S. Rappaport. Wireless Communications: Principles and Practice. Prentice Hall, 2002.

[95] J. Reich, V. Misra, and D. Rubenstein. Roomba madnet: a mobile ad-hoc delay tolerant network testbed. *SIGMOBILE Mob. Comput. Commun. Rev.*, 12(1):68–70, 2008.

[96] J. Reich, V. Misra, and D. Rubenstein. Spreadable connected autonomic networks. Technical report, CUCS-016-08, Columbia University, 2008.

[97] C. Reis, R. Mahajan, M. Rodrig, D. Wetherall, and J. Zahorjan. Measurement-based models of delivery and interference in static wireless networks. In *Proceedings of ACM SIGCOMM*, Pisa, Italy, September 2006.

[98] J. Robinson and E. Knightly. A performance study of deployment factors in wireless mesh networks. In *Proceedings of IEEE InfoCom*, Anchorage, AL, May 2007.

[99] J. Robinson, R. Swaminathan, and E. W. Knightly. Assessment of Urban-Scale Wireless Networks with a Small Number of Measurements. In *Proceedings of ACM MobiCom*, San Francisco, CA, September 2008.

[100] Router Board. http://www.routerboard.com.

[101] B. Sadeghi, V. Kanodia, A. Sabharwal, and E. Knightly. Opportunistic media access for multirate ad hoc networks. In *Proceedings of ACM MobiCom*, Atlanta, GA, September 2002.

[102] Akbar M. Sayeed and Vasanthan Raghavan. Maximizing MIMO capacity in sparse multipath with reconfigurable antenna arrays. *IEEE Journal on Special Topics in Signal Processing*, 2007.

[103] K. R. Schaubach, N. J. Davis, and T. Rappaport. A ray tracing method for predicting path loss and delay spread in microcellular environments. In *Proceedings of IEEE Vehicular Technology Conference*, May 1992.

[104] S. Seshan, M. Stemm, and R. H. Katz. SPAND: Shared passive network performance discovery. In *Proceedings of USENIX Symposium on Internet Technologies and Systems*, Monterey, CA, December 1997.

[105] R. Sivakumar, P. Sinha, and V. Bharghavan. CEDAR: Core extraction distributed ad hoc routing. *IEEE JSAC*, 17(8):1454–65, 1999.

[106] Soekris Engineering. http://www.soekris.com.

[107] M. R. Souryal, J. Geissbueheler, L. E. Miller, and N. Moayeri. Real-time deployment of multihop relays for range extension. In *Proceedings of ACM MobiSys*, San Juan, Puerto Rico, June 2007.

[108] A. P. Subramanian, H. Gupta, S. R. Das, and J. Cao. Minimum interference channel assignment in multi-radio wireless mesh networks. *IEEE Transaction on Mobile Computing*, December 2008.

[109] S. Thrun. Robotics mapping: A survey. Technical report, Technical Report CMU-CS-02-111, School of Computer Science, Canegie Mellon University, February 2002.

[110] VisiWave, AZO Technologies, Inc. http://www.visiwave.com, 2003.

[111] W. Wang, V. Srinivasan, and K. C. Chua. Trade-offs between mobility and density for coverage in wireless sensor networks. In *Proceedings of ACM MobiCom*, Montreal, CA, September 2007.

[112] C. Williamson. Internet traffic measurement. *IEEE Internet Computing*, 05(6):70–74, November 2001.

[113] S. H. Y. Wong, H. Yang, S. Lu, and V. Bharghavan. Robust rate adaptation for 802.11 wireless networks. In *Proceedings of ACM MobiCom*, Los Angeles, CA, September 2006.

[114] H. Wu, X. Wang, Y. Liu, Q. Zhang, and Z.-L. Zhang. SoftMAC: Layer 2.5 mac for VoIP spport in multi-hop wireless networks. In *Proceedings of IEEE Secon*, Santa Clara, CA, September 2005.

[115] W. Xu, T. Wood, W. Trappe, and Y. Zhang. Channel surfing and spatial retreats: Defenses against wireless denial of service. In *Proceedings of ACM WiSe*, Philadelphia, PA, October 2004.

[116] Y. Xu and W.-C. Lee. Exploring spatial correlation for link quality estimation in wireless sensor networks. In *Proceedings of IEEE PerCom*, Pisa, Italy, March 2006.

[117] Q. Xue and A. Ganz. Ad hoc QoS on-demand routing (AQOR) in mobile ad hoc networks. *ACM Journal of Parallel and Distributed Computing*, 63(2):154–165, 2003.

[118] J. Yin, Q. Yang, and L. Ni. Adaptive temporal radio maps for indoor location estimation. In *Proceedings of the IEEE PerCom*, Kauai, Hawaii, March 2005.

[119] J. Yin, Q. Yang, and L. M. Ni. Learning adaptive temporal radio maps for signal-strength-based location estimation. *IEEE Transactions on Mobile Computing*, July 2008.

[120] C. Yu, K. G. Shin, and L. Song. Link-layer salvaging for making routing progress in mobile ad hoc networks. In *Proceedings of ACM MobiHoc*, Urbana-Champaign, IL, May 2005.

[121] C. Yu, K. G. Shin, and H. Y. Youn. Power-stepped protocol: Enhancing spatial utilization in a clustered mobile ad hoc networks. *IEEE JSAC*, pages 1322–1334, September 2004.

[122] Y. Yuan, S. H. Wong, S. Lu, and W. Arbuagh. ROMER: Resilient opportunistic mesh routing for wireless mesh networks. In *Proceedings of IEEE WiMesh*, Santa Clara, CA, September 2005.

[123] J. Zhao, H. Zheng, and G.-H. Yang. Distributed coordination in dynamic spectrum allocation networks. In *Proceedings of IEEE Symposium on New Frontiers in Dynamic Spectrum Access Networks*, Baltimore, MD, November 2005.