Stony Brook University

Cross-layer Design for Interference Mitigation and Mobility Support in Wireless Access Networks

A Dissertation Presented
by
Vishnu Navda
to
The Graduate School
in Partial fulfillment of the
Requirements
for the Degree of

Doctor of Philosophy
in
Computer Science
Stony Brook University

December 2007

Stony Brook University

The Graduate School

Vishnu Navda

We, the dissertation committee for the above candidate for
the Doctor of Philosophy degree,
hereby recommend acceptance of this dissertation.

Dr. Samir R. Das
Advisor
Associate Professor of Computer Science

Dr. Tzi-cker Chiueh
Professor of Computer Science

Dr. Himanshu Gupta
Assistant Professor of Computer Science

Dr. Samrat Ganguly
Research Staff Member
NEC Labs America at Princeton

This dissertation is accepted by the Graduate School.

Lawrence Martin
Dean of the Graduate School

Abstract of the Dissertation

Cross-layer Design for Interference Mitigation and Mobility Support in Wireless Access Networks

by
Vishnu Navda

Doctor of Philosophy
in
Computer Science

Stony Brook University

2007

Over the past few years WiFi technology has emerged as the most widely used wireless technology for providing network access to end-users over single hop networks (WLANs) and multiple hop networks (Mesh Networks). Although WiFi-based access networks have numerous benefits such as use of inexpensive commodity hardware, operation in unlicensed spectrum and ease of deployment, they suffer from performance and reliability related problems that arise mainly due to the presence of wireless interference. The problems are exacerbated when there are mobile clients in the network. In this regard, we focus on two important issues: mitigating interference and supporting client mobility. We design novel cross layer solutions that utilize information available at the link layer and the network layer to effectively address these issues. Specifically, we contribute in four specific problem areas related to interference and mobility.

First, interference and fading can cause sudden drop in wireless link quality, which can persist for hundreds of milliseconds. But, existing routing protocols for mesh networks are incapable of reacting quickly to such transient link quality fluctuations. To address this issue, we design a lightweight opportunistic protocol called **Deflect** that operates underneath the routing layer, and protects end-to-end routes from short-term link quality degradation. Deflect achieves this using a novel passive link quality measurement technique along with a low overhead path switching

protocol. Second, in dense WLAN deployments interference between neighboring access-points operating on the same channel can be significant due to limited number of non-conflicting channels in WiFi networks, and this results in degradation of achievable network throughput. In this regard, we design a dynamic transmit power control technique called **Contour** that improves spatial reuse while avoiding link asymmetry problem associated with the use of multiple transmit power levels in the network. The Contour framework uses of a centralized slotted packet scheduling framework enforced on top of the existing IEEE 802.11 CSMA/CA protocol to achieve fine grained transmit power control and packet scheduling across all access-points in the network. Third, a malicious user can create interference and jam all communication on a channel. To counter such an extreme form of interference, we design a **Channel Hopping** protocol that improves the resilience of WiFi networks to jamming attacks. The hopping parameters are optimized to maximize the achievable throughput in presence of a smart jammer, who is aware of the use of a hopping protocol. Finally, client mobility causes temporary disruption in connectivity. This is particularly problematic in mesh networks. We design a mobility management architecture called **iMesh** that tracks the client's location using link layer feedback information and quickly updates the routes on the mesh network for mobile clients.

We design and implement system prototypes of our proposed solutions on commodity off-the-shelf hardware to show the feasibility of deploying these solutions. Further, through extensive experimentation on realistic wireless testbed deployments, we demonstrate the effectiveness of these systems in addressing the aforementioned issues.

To my grandmother Girijamma

Contents

List of Figures

List of Tables

Acknowledgments

I am greatly indebted to my adviser Professor Samir Das for his efforts in making this dissertation possible. The key ideas and concepts in this thesis came out of innumerable long and lively meetings we had over the course of last 5 years. His persistent efforts helped me hone my presentation and writing skills. He gave me enough freedom to think independently and come up with new ideas, while also providing constructive criticism. I thank him for being understanding and helpful when I was working remotely from Princeton. I also thank his wife June for often inviting us to get-togethers, and building a long lasting friendly relationship.

I am fortunate to have worked with many Professors at Stony Brook who are experts in different areas of research. Professor Himanshu Gupta, my co-adviser early on in my PhD., helped me improve my algorithmic and theoretical background. It was very exciting to work with Professor Tzi-cker Chiueh during my Masters. He inspired me to pursue further research, and also helped me to improve my systems background.

I spent two memorable years at NEC Labs in Princeton. I learned a lot from my manager Samrat Ganguly. He is a very good friend and a great mentor. I will cherish those long conversations we had outside the office building discussing research problems and everything else under the sun. It was inspiring to work with Ravi Kokku. His problem solving skills and clarity of thought were amazing, and I picked up quite a bit from him. I also thank Dan Rubenstein for giving me an opportunity to work with him. I enjoyed our discussions together.

There are many friends and colleagues at Stony Brook who have helped me in different ways. I take great pleasure to acknowledge their efforts and mention at least a few of them here. Anand Kashyap has been a great lab buddy ever since I started my PhD. We have worked on so many projects, traveled to several conferences, and spent many sleepless nights working on paper deadlines. Salil Gokhale, my lab-mate and a very close friend, helped me get acquainted to wireless systems area. We had a lot of fun setting up the lab network from scratch. Lap Chung Lam helped me build my systems skills and also motivated me to pursue PhD.

My collaborators Anand Kashyap, Vishal Chowdhary, Anand Prabhu, Kannan Dhanasekaran, Andreas Timm-Giel, Aniruddha Bohra, Dan Rubenstein, Ravi Kokku, Samrat Ganguly and Dragos Niculescu have contributed in different capacities to my success.

I enjoyed the company of my lab-mates Zongheng Zzhou, Shweta Jain, Bin Tang, Ritesh Maheshwari, Pralhad Deshpande and Swapnil Patil. Thank you all for keeping a lively atmosphere in the lab.

I enjoyed the company of my friends Akshay Athalye and Alok Tongaonkar. I could always count on them for any help at anytime of the day.

I like to thank Amma, Appa and Veena for their support and encouragement. Their constant queries regarding my thesis kept me focused on completing it soon. Finishing the last 10% can be especially time consuming. But, I was lucky to have Smitha Udupa enter my life, and provide the impetus to finish this thesis on time.

Chapter 1

Introduction

1.1 Motivation

The IEEE 802.11 standard complaint radios, also popularly known as WiFi were first introduced ten years ago. Since their inception, several newer variants of the protocol were introduced, among which 802.11a, 802.11b, 802.11g and more recently 802.11n are the most popular and widely available radios. There are several reasons for WiFi technology to become very popular. Firstly, WiFi radios operate in the ISM frequency bands that requires no licensing fees for usage. In addition, the hardware and operational cost of WiFi is very low. Secondly, WiFi technology provides flexible operational modes to cater to different deployment scenarios. In deployments where wired backbone network is already existing, WiFi based base-stations or access-points can be connected to the wired backbone network to provide wireless coverage. Any end-user with a compatible WiFi equipped device can connect to the nearby access-point to get high throughput network connectivity over a single hop wireless link. Since the connectivity range of each AP is typically few hundred meters in outdoors and few tens of meters in indoors, multiple WiFi APs are used to blanket a large region with WiFi coverage. WLAN deployments are commonly used in enterprise networks [39] campus-wide networks [3] and hot-spot networks [11]. WiFi radios can be deployed as multi-hop wireless networks or Mesh Networks that are composed of WiFi routers, which relay pakets over multiple hops. Some of these routers are connected to the wired backbone, which form the gateways. WiFi access-points are connected to the mesh network routers to provide end-users connectivity. Mesh networking enables extenstion of wireless coverage without the need for additional wiring. Some of the examples of mesh deployments are city-wide networks such as Google WiFi in San Francisco [4], Roofnet in Boston [25], and Houston's urban network [29]. In this thesis, we use

the term Wireless Access Networks or WANs to refer to WiFi networks in general, which can be either WLANs or Mesh Networks.

Although WiFi technology is gaining grounds, achieving reliability and robustness in dynamic wireless environments and meeting the growing bandwidth requirements is still a significant challenge. Wireless links in a WAN experience dynamic and fluctuating conditions over a range of time scales due to several external conditions, each of which result in failures and service disruptions within the network. Such disruptions are unacceptable for deployments in which clients require service guarantees like bandwidth, loss, delay and connectivity.

We discuss two classes of external conditions, which result in link dynamics at different range of time-scales. We also discuss the research challenges that exist for handling these dynamics.

Interference: A wireless link can experience interference from other wireless transmissions occurring in its surroundings as well as from external noise present on the same frequency spectrum. Often, interference from such sources are unintentional, but sometimes it can be deliberately generated by a malicious user intending to block all communication on a particular wireless channel. Interference can degrade the quality of a link, thereby causing packet losses [74] that severely impact the perceived quality of applications on the end user, especially those applications that are delay and loss sensitive. In addition, interference among the nodes in the network also causes increased contention among these nodes for channel access, thereby limiting the total capacity of the network.

To counter the link quality variations, the 802.11 MAC layer is designed to automatically lower the link layer bitrate. Such an adaptation at the MAC layer is not sufficient because of the following reasons. Although packet losses can decrease, but the link capacity also reduces. Moreover, when the link quality is very poor, even using the lowest bitrate can still incur significant packet losses. When the link quality stays poor for a long duration of time, the routing protocol can detect such failures and switch to an alternative path. For practical reasons, the existing routing protocols for wireless mesh networks, such as DSDV [91], OLSR [32], AODV [90], DSR [51], LQSR [37], SrcRR [35], respond conservatively to sudden degradations in link quality, thus are not sensitive to short time-scale degradation of link qualities. Thus, adaptations at both these layers are insufficient to handle short term link fluctuations caused due to interference.

To mitigate interference and increase the capacity of the network, there are two major directions: channel allocation and spatial reuse through transmit power control. Although there are several solutions in the literature that use the first approach, the spatial reuse dimension has not been exploited in the context of 802.11-based WANs for increasing the network capacity. The main problem with using uncoordinated transmit power control is formation of asymmetric links in the network. An asymmetric link is formed between two nodes whenever one of the nodes is able

to sense the other node's transmission while the converse is not true. The 802.11 CSMA/CA based MAC relies on symmetric carrier sensing property for ensuring a fair access to the channel. Consequently, asymmetric links can cause unfair allocation of network resources with the possibility of long-term starvation of certain clients.

IEEE 802.11 based protocols are designed for deployment in cooperative environments, and hence do not include mechanisms to protect from such jamming attacks. Such attacks are becoming increasingly feasible due to the availability of programmable radios with most of the functionality implemented in software. It is now very easy to modify the software of a wireless radio in order modify the standard CSMA/CA protocol for access to the channel. This enables a user to unfairly get access to the channel for any amount of time, thus starving all communications on that channel.

Client Mobility: The client to AP link is dynamically changing as the client moves from one cell to another. A new link is created between the client and the new AP, while the link between the old AP and the client no longer exists, The point of attachment of the client to the WAN changes as a result of client mobility. This gives rise to a mobility management problem – how to deliver frames destined to a client when its point of attachment to the WAN has changed. The handoff latency is the duration of time when the client connectivity is disrupted. Many applications require that the handoff delay be as small as possible with minimum or no loss of packets.

1.2 Research Issues and Contributions

In this thesis, we designed and implemented several novel cross-layer solutions that involve the medium access control layer and the network layer for adapting to the above mentioned link dynamics. All solutions proposed in this thesis have been evaluated on real 802.11-based wireless testbed deployments with prototype implementation.

Interference-aware Fast Path Adaptation in Mesh Networks: To mitigate small time-scale fluctuations we designed a mechanism that enables fast adaptation of the end-to-end path by temporarily circumventing the poor links. To be effective, such an adaptation mechanism must act at a short time scale so that it is able to a) *quickly detect the degradation of the link quality*, b) *switch to a path circumventing the poor link*, and c) *react quickly with negligible control overhead.* We present a lightweight mechanism, called Deflect. Deflect achieves this goal by (a) a passive, *zero-overhead*, measurement driven approach for monitoring link qualities and (b) switching to alternate routes only in the local neighborhood by very low-overhead signaling.

Slotted Symmetric Transmit Power Control Framework in WLANs: We devised a transmit power control mechanism that is able to increase spatial reuse without sacrificing fairness; we propose a novel *slotted symmetric* power control framework. The main idea underlying this framework is to have all access points operate at the same power level at any given instant of time and have all access points follow a sequence of power levels synchronously. Time is divided into slots, and in each slot all access points operate at same power level, thereby avoiding any link asymmetry. Through hopping over different power levels across successive slots, all access points are able to serve the clients at their corresponding minimum power level requirements.

Channel Hopping Protocol for Jammer Resilience in WLANs: We analysed how to protect WLANs from jamming attacks by having the legitimate transmission hop among channels to hide the transmission from the jammer. Using a combination of mathematical analysis and prototype experimentation, we explore how much throughput can be maintained in comparison to the maintainable throughput in a cooperative, jam-free environment. Our mathematical analysis allows us to extrapolate the throughput that can be maintained when we relax certain constraints, such as the number of orthogonal channels and the number of channels simultaneously used for both legitimate communication and for jamming.

Client-Aware Routing Protocol for Seamless Mobility in Mesh Networks: In order to provide seamless networking services to the mobile clients, we designed a link-state based routing protocol that tracks the movement of clients using feedback from the MAC layer and updates the routes on the backbone network to quickly divert the client traffic to the currently associated cell. Handoff latency is optimized as a result of the triggered route update. In addition, packet losses during handoff is eliminated by smartly buffering the packets at the old AP and forwarding them to the client via the new AP after the handoff is complete.

1.3 Outline

In the following chapters, we describe details of the aforementioned research problems, and our proposed solutions along with extensive theoritical and experimental results. We start by describing the impact of interference on routing protocols for mesh networks and describe our fast path adaptaion mechanism in Chapter 2. Next in Chapter 3, we look at the problem of co-channel interference in dense WLAN deployments and describe our transmit power control solution. In Chapter 4 we analyze the problem of jamming in WiFi networks and present our channel hopping protocol. Following this, in Chapter 5 we describe the client mobility support problem and describe our mobility management architecture. Finally we conclude this dissertation in Chapter 6 and summarize our contributions.

Chapter 2

Interference-aware Fast Path Adaptation in Mesh Networks

2.1 Introduction

In this chapter we investigate the issue of supporting real-time applications such as streaming/interactive voice and video for wireless mesh networks. Examples include providing VoIP coverage in enterprises, and supporting streaming video from surveillance cameras to monitoring locations in communities. To support these applications, a mesh network must provide a steady level of quality of service guarantees. However, the wireless link carrying the streaming data can undergo sudden quality degradation due to interference from other wireless transmitters and external noise sources. This may cause frequent bursts of frame losses [74] resulting in a severe impact on the perceived quality of real-time applications.

For practical reasons, existing routing protocols for wireless ad hoc or mesh networks, such as DSDV [91], OLSR [32], AODV [90], DSR [51], LQSR [37], SrcRR [35], respond conservatively to sudden degradations in link quality. The limitation in the sensitivity of the routing protocols to link quality is difficult to overcome due to the following reasons. All link quality metrics used in routing, such as ETX [35, 37] require active probing,[1] but frequent probing results in high overhead and can interfere with the actual traffic. Further, a time window of 5 to 10 seconds [37] is required to gather enough probe samples to allow reasonable confidence in the statistics and avoid load sensitivity. Finally, frequent updates of routing metric across the network also results in high control overhead. Additional latency is also incurred in propagating routing updates. The resulting time scale of recovery from poor links can cause significant interruptions in streaming applications – a 10

[1] ETX (expected transmission count) measures the expected number of transmissions needed to send a unicast packet over an 802.11 link. Since in 802.11, unicast packets use a link-layer acknowledgment, ETX is measured as the inverse of the product of delivery ratios of probe packets in both directions for the link.

second loss of a link can result in a loss of 250 frames from a typical video stream.

Our goal in this work is to design a mechanism that enables fast adaptation of the end-to-end path by temporarily circumventing the poor links. To be effective, such an adaptation mechanism must act at a short time scale so that it is able to a) *quickly detect the degradation of the link quality*, b) *switch to a path circumventing the poor link*, and c) *react quickly with negligible control overhead.*

We present a lightweight mechanism, called **Deflect** for fast path adaptation. Deflect is decoupled from the end-to-end route construction and can complement many existing routing protocols. Deflect's goal is to protect the end-to-end route from short-term link quality degradation or outage before the end-to-end route recovery takes over. Deflect achieves this goal by (a) a passive, *zero-overhead*, measurement driven approach for monitoring link qualities and (b) switching to alternate routes only in the local neighborhood by very low-overhead signaling. Both these strategies enable rapid path adaptation in the order of hundreds of milliseconds in commodity 802.11-based deployments. Deflect preserves the end-to-end number of hops, preventing sudden changes in end-to-end delays. Earlier protocols such as Divert [74] and ExOR [26] have been proposed that exploit link or path diversity similar to Deflect, but Divert is limited to single-hop WLANs, while ExOR is applicable only for bulk-transfer data and not real-time traffic.

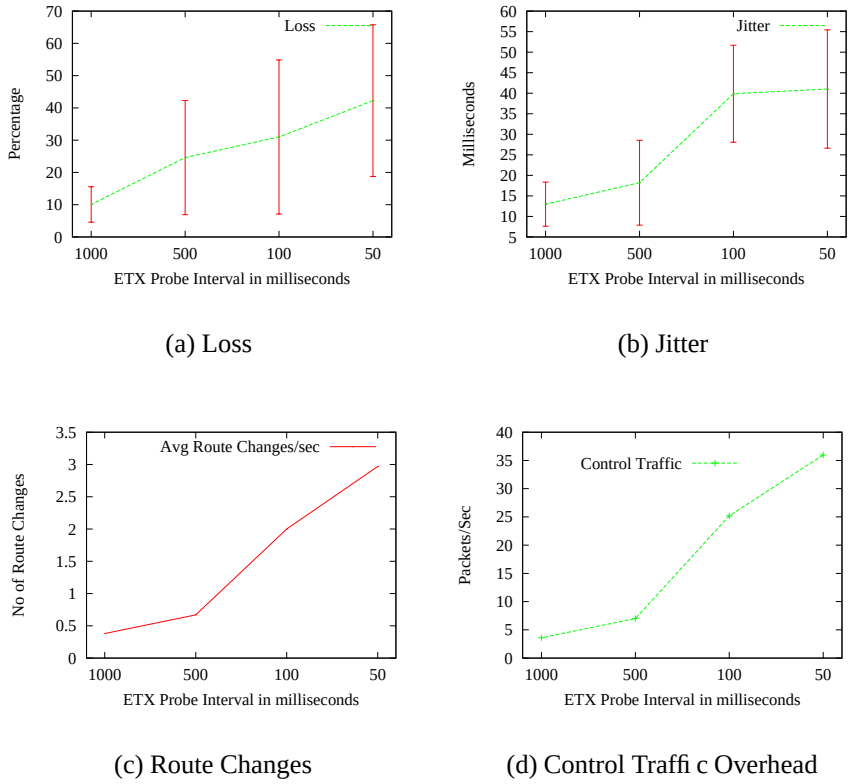
In the rest of the chapter we design and evaluate Deflect and demonstrate its performance in an indoor 802.11-based mesh network. First, we describe the Deflect protocol, which is based on the novel concept of *cooperative relay* nodes. A cooperative relay node silently monitor links in its vicinity and, if and when appropriate, volunteer to bypass them in a fashion transparent to the routing protocol. We argue that such relays will be plentiful in a dense mesh network. We then describe the detailed design of Deflect particularly with regard to how link qualities are monitored and estimated, and how paths are switched. Finally, we present evaluations to analyze the advantages of using Deflect.

2.2 Motivation and Overview

In this section, we first present a set of experiments demonstrating the inadequacy of routing protocols using a probing-based metric like ETX to recover from short-term link quality fluctuations. Then, we present an overview of the design of Deflect and then do a study to determine the frequency with which Deflect may be applicable.

2.2.1 Limitations of routing protocols

To evaluate the responsiveness of routing protocols and routing metrics to adapt to bursty background traffic, we set up a series of experiments in our 20 node indoor



(a) Loss (b) Jitter

(c) Route Changes (d) Control Traffi c Overhead

Figure 2.1: Effect of ETX probe interval on loss, jitter and route changes and route control traffic overhead.

mesh testbed that uses 802.11b radios. The actual details of the testbed architecture is not relevant here. It is described later in Section 2.4. These measurements are done on a section of testbed with 5 nodes. Two nodes, say S and D, act as traffic source and destination respectively, with three 2-hop paths available between them, via three other nodes, say A, B, C. We run a link-state protocol OLSR [32] for routing and use ETX [35] as the path metric. By default, ETX uses a 10 sec time window with probes at the rate of 1 probe/sec.

We also set up background traffic to deliberately cause link quality fluctuations. This traffic is generated by node A in the form of 10 sec long bursts of back-to-back broadcast frames alternating with 10 sec silence periods. This causes channel contention at nodes B and C. By default, the ETX metric used for routing is calculated by using the statistics of the last 10 ETX probe packets, where the probes are sent at the rate of 1 probe/sec. This measurement window controls the responsiveness of ETX with respect to changing conditions. Similar parameter setting has been used in literature in the past [37].

Figure 2.1 depicts various performance metrics obtained during a 100 sec run

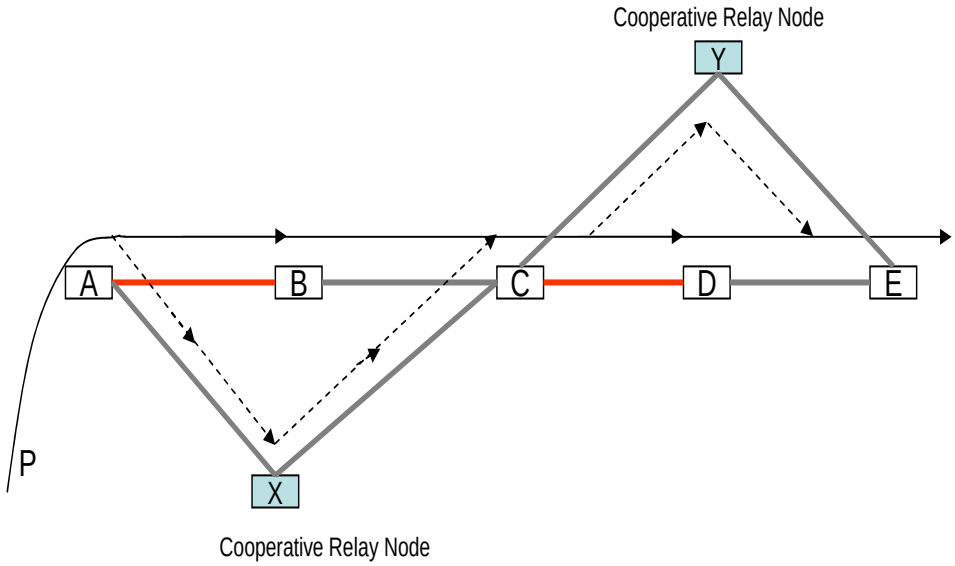


Figure 2.2: Path adaptation overview.

of 1000 Kbps CBR traffic from S to D, averaged over several such runs. With the default probing frequency, the CBR flow incurs about 10% packet losses due to the interference traffic. Such losses are due to the lack of responsiveness in the metric calculation as mentioned before – route changes lag interference changes by several seconds. However, increasing probing frequency (500ms, 100ms and 50ms) does not improve the performance. Packet losses actually increase with probe frequency as probes themselves interfere with data traffic (Figure 2.1(a)). For the same reason, the delay jitter also increases (Figure 2.1(b)). As far as routing is concerned, route change frequency increases (Figure 2.1(c)) with probe frequency, as now routing becomes more reactive, increasing control overhead as well (Figure 2.1(d)). Higher control overhead is another reason for increasing loss and jitter.

Above experiments show that existing routing protocols cannot adapt quickly to handle transient losses. There is a need for fine grained path adaptation to support realtime media over mesh networks. We conclude that any active probing method is not suitable to handle short term variations due to self-interference, control overhead and route instability.

2.2.2 Basic Idea of Deflect

The basic idea in Deflect is that a node can volunteer to replace a neighboring relay node on a routing path if it determines – via passive measurements based on snooping – that the overall delivery rate will improve. Consider Figure 2.2 for an illustration. Assume that $A - B - C - D - E$ is a route computed by the routing

protocol based on long term observation of the ETX metric [35] of each link of the network. Although alternative routes such as $A - X - C - Y - E$ exist, let us assume that they have a higher cost as determined by the metric. Deflect provides a low overhead mechanism for fast switching to the alternate route depending upon the condition of links AB, BC, CD, and DE. The advantage of Deflect comes with the limitation that in the interest of quick response, it can adapt the path only using nodes in the neighborhood, and cannot do any end-to-end rerouting. Reconstruction of an overall better end-to-end path considering complete topology of the mesh network is still the responsibility of the routing protocol. But this activity incurs substantial overhead and can only be done over a longer time-scale.

In Deflect, a node such as X is called a *cooperative relay* for a node such as B that is the original relay node on a path. X monitors the performance of the AB and BC links (in terms of their delivery ratios) by overhearing the packets transmitted from A and from B. Now, if X determines, based on this estimation, that the product of the delivery ratios of AB and BC is worse than that of AX and XC, it signals node A so that A temporarily flips the next hop route through X. At this time, B can start playing the role of X. Similar relay switching can happen at any hop in an end-to-end path, for example, again at Y. Clearly, the scheme depends on the existence of such cooperative relay nodes X and Y. We show in Section 2.2.3 that in a dense, large-scale deployment network such cooperative relays are expected to be quite common. The scheme also depends on the accuracy of estimation of the link qualities for AB, BC, AX and XC. The estimation procedure is described in Section 2.3.1.

Note again that Deflect keeps the overhead low by restricting route changes only in the radio neighborhood of the original route computed by the routing protocol. It depends on available route diversity in this neighborhood. It does not change the end-to-end path length in number of hops, thus avoiding any sudden change in the delay and major packet reordering problems. It does increase packet reordering; but – as our performance evaluation later shows – this is minor enough that can be easily handled by a small reorder buffer.

2.2.3 Existence of Cooperative Relays

The success of Deflect obviously depends on the existence of cooperative relays. In this section, we show that a large number of such cooperative relays do exist in dense networks. Specifically, we argue that for a given *pair* of two-hop paths (say $A - B - C$ and $A - X - C$) with common end points (A and C), the relay nodes (B and X) hear each other with high probability. This can be understood by using an idealized unit-disk graph model.

Assume that if two nodes are within radio communication range unity, they have a perfect link between them. Otherwise, they do not have a link. If the distance between the end points of a two-hop path is r, then $1 < r < 2$. The potential relay

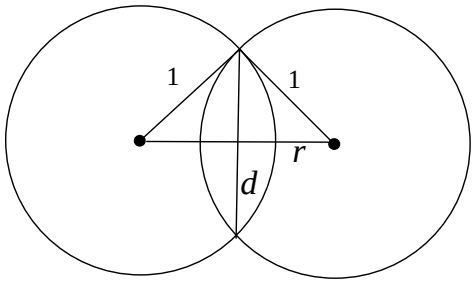


Figure 2.3: Unit disk graph model illustrating probability of existence of two relay nodes within communication range. The relay nodes must be within the intersection region of the two circles.

nodes must lie within the intersection of the unit circles centered at the end points. The maximum distance between any two points selected in this region is given by $d = \sqrt{4 - r^2}$, as shown in Figure 2.3.

We do a Monte Carlo simulation to determine the probability that the distance between any two relay nodes is less than unity. Figure 2.4 shows the likelihood of the distance between two relay nodes being less than 1 with increasing r. Averaging over r gives the probability that any two relay nodes for common endpoints can hear each other. This probability comes to 0.97, which is quite high. It can be argued that even though the two relay nodes (B and X) can hear each other, they may be so close that interference on one node will effect the other node in the same manner. So, we also consider two cases when the distance between relay nodes is atleast 1/4 and 1/2. The average probability in such cases is 0.59 and 0.26 respectively. This analysis shows that whenever there is a pair of two-hop paths with common endpoints, there is a high probability that one of the relay nodes will act as a cooperative relay for the other.

However, what is the guarantee that there is a pair (i.e., an alternate path) to start with? We argue that such redundancy is quite likely. In a recent study [20] on uncoordinated deployment of 802.11 access points in six urban areas in the US, the authors found that the median number of neighbors of an access point is about 4–7 (see Figure 1 in [20]). If we overlay a mesh network on these access points, such high degrees indicate good possibilities for such redundancies.

We obtained some numbers for existence of cooperative relays in two existing testbeds – the mesh testbed at NEC Labs, shown in Figure 2.5, and MIT Roofnet project [72]. The density of mesh testbed at NEC Labs is greater than in Roofnet. We computed the number of cooperative relays for all possible paths of certain lengths in the networks. A path consists of links, where each link has a delivery ratio of atleast 50%. Then the path is broken into consecutive pairs of links, and we determine the number of possible cooperative relay for each such subpath. For

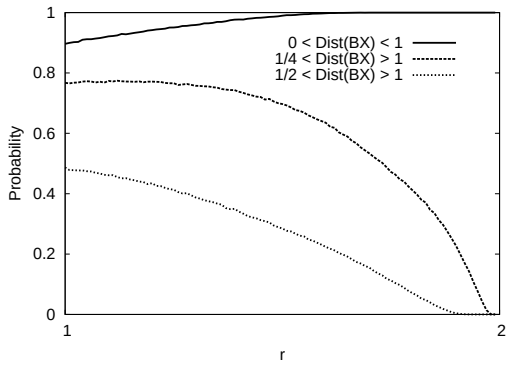


Figure 2.4: Probability that 2 relay nodes for a pair of 2-hop paths with common end points can hear each other. Probability of the two nodes being atleast 1/4 and 1/2 of the transmission range apart is also plotted. Dist(BX) is the distance between the relay nodes B and X.

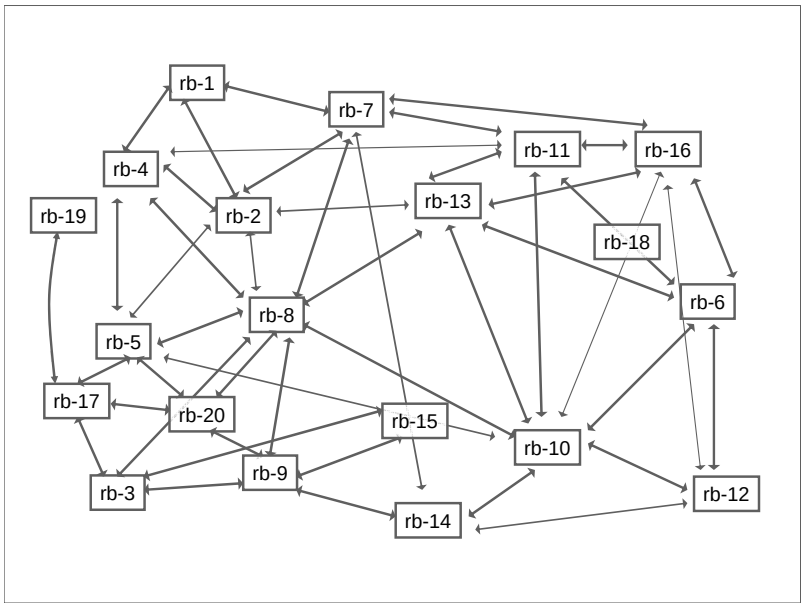


Figure 2.5: Indoor mesh network testbed. Only links with delivery ratio $> 50\%$ are shown.

every such link pair (A-B-C), we consider X as a cooperative relay, if it satisfies the following two conditions –

- $|d_{AB} \times d_{BC} - d_{AX} \times d_{XC}| < 0.2$, where d_{AB} is the delivery ratio of the link AB. This indicates that the difference between the delivery ratios along the two paths is not significant enough to choose one over another, and
- $1 < ETX_{BX} < 6$, indicating that a link exists between B and X, albeit a

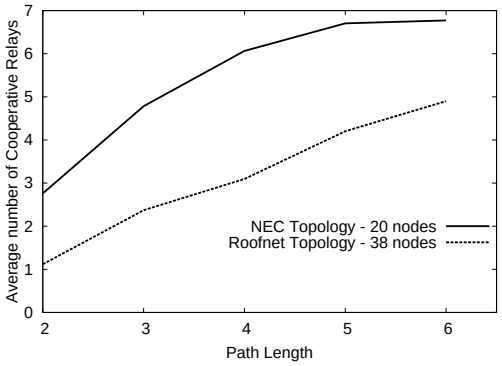


Figure 2.6: Number of cooperative relays as the path length increases. The topologies of NEC Labs testbed and MIT Roofnet are considered.

weak one.

Figure 2.6 shows the average number of cooperative relays as the path length increases for the two networks considered. All possible paths in the network are considered. The result shows that enough cooperative relays exist in networks for Deflect to be effective.

2.3 Deflect Design

This section is devoted to the complete design of Deflect including its testbed implementation. We start with the procedure for link quality estimation. The link quality is indicated by the delivery ratio on the link.

2.3.1 Estimating Link Quality via Passive Monitoring

For ease of discussion, we will refer to Figure 2.2, particularly the sub-topology involving nodes A, B, C, X. Assume that $A - B - C$ is a part of the path computed by routing and X is the cooperative relay node. Node X snoops on the medium and estimates the delivery ratios of all links AB, BC, AX and XC following the procedure we describe here. In 802.11, links must work bi-directionally and thus, the bi-directional delivery ratios (delivery ratio for unicast traffic) must be estimated. Keep this in mind when reading the following procedures. We will discuss this issue further at the end of this subsection.

The estimation is dependent on the use of acknowledgments and sequence numbers in 802.11. In the 802.11 link layer, all unicast frames are acknowledged. If an acknowledgment is not received by the sender, the frame is retransmitted after a suitable backoff. This continues up to a retry limit (4 in our experimental setup) when the frame is dropped. Each frame generated in the link layer carries a 12 bit

sequence number, between 0 and 4095. For each new frame, the sequence number is incremented modulo 4096. Retransmitted frames carry the same sequence number as the original frame. A bit in the MAC-layer header distinguishes between the original and retransmitted frames.

In the discussion that follows, the notation d_{AB} is used to denote the actual delivery ratio for link AB and d_{AB}^X is used to denote the delivery ratio for link AB as estimated by X. The notation $t(AB)$ denotes the number of MAC-layer data frames transmitted on the link AB in a unit time interval. $t^u(AB)$ denotes only frames with unique sequence numbers. The notation $t(AB, X)$ denotes the number of frames transmitted on link AB as heard by node X. Thus, $t(AB, X) \leq t(AB)$.
Estimating link AB: The delivery ratio of link AB can be estimated by simply snooping all frames on link AB at node X and computing the fraction that carries a unique sequence number. Note that

$$d_{AB} = \frac{t^u(AB)}{t(AB)},$$

and

$$d_{AB}^X = \frac{t^u(AB, X)}{t(AB, X)}.$$

If losses on links AX and AB are completely independent,

$$\frac{t^u(AB, X)}{t^u(AB)} = \frac{t(AB, X)}{t(AB)}.$$

Then,

$$d_{AB}^X = d_{AB}.$$

Estimating link AX: The delivery ratio on link AX can be estimated by counting how many frames transmitted by A (on link AB) are heard by X. Thus, no actual traffic on the link AX is needed. However, this is a harder problem as frames transmitted on link AB could contain one or more retransmitted frames for each originally transmitted frame. All retransmitted frames are identical. Thus, the number of retransmitted frames on AB cannot be accurately estimated via snooping. Snooped acknowledgment frames from B cannot be gainfully used either, since X has no guarantee that the sender A actually receives the acknowledgment. Because of these limitations, we simply estimate d_{AX} as

$$d_{AX}^X = \frac{t^u(AB, X)}{t^u(AB)}.$$

Now, $t^u(AB)$ is unknown at X and must again be estimated. This is estimated by adding to $t^u(AB, X)$ the number of "holes" in the sequence numbers among the set of packets used to compute $t^u(AB, X)$.

Estimating link BC: This procedure is similar to link AB. Note that here we exploit the ability of X to overhear B via link BX.
Estimating link XC: The delivery ratio from C to X in one direction is used as an estimator for link XC. The technique here changes depending on whether C transmits any frames (for any node). If true, the link CX can be estimated exactly as in link AX. Otherwise, the procedure is slightly more complex. Essentially, it involves estimating what fraction of ACK frame transmissions from C to B are heard by X. ACK frames do not carry sequence numbers. They also do not carry source MAC address. Thus, node X must correlate each overheard data frame to each overheard ACK frame. If X hears the data frame on link BC, but misses the corresponding ACK frame and does not hear a following data frame retransmission, X concludes that it has lost the ACK frame. d_{CX} is then estimated as follows:

$$d_{CX}^{X} = \frac{\text{\# ACK frames heard}}{\text{\# ACK frames heard + \# ACK frames lost}}.$$

Note that in this method d_{CX}^{X} underestimates d_{CX}.

Recall that in 802.11 links are bi-directional, and thus delivery ratios must be estimated in a bi-directional sense. The above estimation procedure estimates the delivery ratios at node X. According to the procedure, d_{AB} and d_{BC} are estimated in the bi-directional sense; but d_{AX} and d_{CX} are estimated only in one direction (viz., AX and CX). In the absence of any further information, it is simply assumed that d_{AX}^{X} and d_{CX}^{X} are bi-directional estimators. It is possible to improve this estimation further, by observing how the bi-directional ETX metric differs from unidirectional ETX metrics on these links. This will require availability of ETX metric information to the Deflect sublayer. In our experience, however, the simplest mechanism works quite well in practice.

Finally, path $A-B-C$ should be switched to $A-X-C$ if $d_{AB}d_{BC} < d_{AX}d_{XC}$ where all delivery ratios are computed in the bi-directional sense.

2.3.2 Path Adaptation Heuristic

In the previous subsection, we have described the general strategy for estimating link qualities by cooperative relay nodes. In general, more than one cooperative relay nodes are available for a two-hop path, depending on the network topology and link qualities. Any node satisfying the topological conditions discussed in Section 2.2.3 can offer to be a cooperative relay for a two-hop path. A node can also offer to be a relay for multiple paths. It is possible for any node X to verify whether it can act as a cooperative relay for a two hop path $A-B-C$ by simply overhearing transmissions on links AB and BC, and thereby ascertaining actual occurrence of packet forwarding on path $A-B-C$. Note that this can be determined just via snooping, even without access to any routing layer information. This is important as one of our design goal is to implement the mechanism below, and transparent to,

the routing layer.

Once a node (say, X) determines that it is a potential cooperative relay, it continues to evaluate the link qualities as detailed in Section 2.3.1. The delivery ratios are estimated in *time windows* of length H. H defines the path switching time granularity and must be chosen carefully. Too small a value will make the statistics, and hence switching decision unreliable. On the other hand, too large a value will tend to average out short term link fluctuations. In general, H should be much less than ETX metric measurement intervals, and thus H should not exceed a few seconds. The lower limit on H depends on the traffic on the path $A - B - C$, since the rate at which samples can be collected to estimate the link quality will be proportional to traffic. This means that with slower traffic (in packets/sec) H must be larger. However, effectiveness of the path adaptation is best observed with faster traffic where a number of consecutive packets can be lost due to burst errors. Thus, our interest here is in supporting streaming/interactive media traffic such as VoIP (20ms inter-packet time in one direction), where H can be kept reasonably small (a few hundreds of milliseconds).

A second parameter of importance is a *sensitivity* parameter. This is indicated by a threshold T, by which the estimated delivery ratio on the alternate path ($d_{AX}d_{XC}$) must exceed the estimated delivery ratio on the primary path ($d_{AB}d_{BC}$), before the path is switched. Note that the use of parameters H and T are similar to their use in the Divert [74]. As suggested in [74], it is also possible to adapt the values of H and T. However, in this work, we limit ourselves to statically chosen values. Adapting the values with traffic and link behaviors is a topic of further research.

2.3.3 Path Switching

The path switching protocol is straightforward. All potential cooperative relays (i.e., nodes like X in Fig 2.2) passively monitor the link qualities continuously on the two-hop paths it is able to bypass. The procedure outlined in section 2.3.1 is used to determine the estimated delivery ratios. If the estimated delivery ratio on the alternate path exceeds that in the primary path by a threshold T in a time window H, the cooperative relay transmits a *signaling message* to the source of the two-hop path (i.e., node A) notifying the availability of a better path. The message contains the ID of the cooperative relay (X) and the ID of the nodes in the path it can help bypass (i.e., $A - B - C$). The source node at this point simply switches the next hop to the cooperative relay (X). After a path is switched, the role of the original relay node (e.g., B) and the cooperative relay (e.g., X) is reversed. This happens automatically, and not through explicit messaging. Thus, the path can be switched back to the original later when the link qualities change.

Deflect sublayer is implemented beneath the routing or packet forwarding layer. It uses a simple mechanism to keep track of multiple paths that may multiplex at the cooperative relay and have the same prior hop. Again referring to Figure 2.2,

the relay node X maintains a list of destination IP addresses of packets that it heard along the subpath $A - B - C$ during the last measurement interval. When X takes over the relaying and informs A about the alternate path $A - X - C$, it also sends A this list. Now assume that A chooses X as the new relay node. For every outgoing packet whose destination MAC address is B, the Deflect sublayer at node A checks whether the packet's destination IP address exists within the list sent by X. If it does exist, it changes the destination MAC address of that packet from B to X. This check ensures that only those packets that were previously routed via path $A-B-C$ are now re-routed via $A-X-C$. The Deflect sublayer of X also checks on every packet received from A to verify whether the destination IP address is in the list it just constructed. If so, it transmits the packet to node C without sending the packet up to the routing or packet forwarding layer. Any other packet (that is not "deflected") received from A must be targeted to a different destination, and is sent up to the routing or packet forwarding layer.

Note that when a path is switched, it is possible for Deflect to notify routing. However, our design goal in this work is to operate Deflect *underneath and transparent to routing*. This way, long term and end-to-end path changes are generated by the routing protocol and short term and local changes are generated by Deflect. At this time, we do not explore any further tighter coupling between these two protocol layers. Also note that Deflect cannot introduce any routing loops not present in the routing protocol. This is because it simply replaces a two-hop sub-path by another with the same end points. Deflect is dependent on the assumption that nodes are cooperating to improve overall performance, much like routing protocols in ad hoc or mesh networks. There are indeed security implications of path switching, similar to any routing protocols. We expect that security issues can be addressed via authentication of signaling messages from X. However, we do not explore this issue here.

2.3.4 Implementation

We have implemented Deflect on Linux using *Click* modular router [57]. The latest `madwifi` device driver [64] is used in our testbed that works in 802.11/a/b/g interfaces with the Atheros chipset. It allows creation of an additional raw *virtual* device (`ath0raw`) for each wireless interface that allows reception of all 802.11 frames (control, management, data) as if in monitor mode, while the main interface can still operate in the ad hoc or infrastructure mode. This raw virtual interface is used in the passive monitoring procedure. A new Click element is implemented that estimates the delivery ratios on different links passively. See Figure 2.7. All frames received on the raw virtual interface are processed by this element to update various link quality related statistics. Then the statistics is evaluated in regular intervals (H) to check if there is any benefit of switching the current path. The switching logic uses this estimation to send the signaling message to the source node.

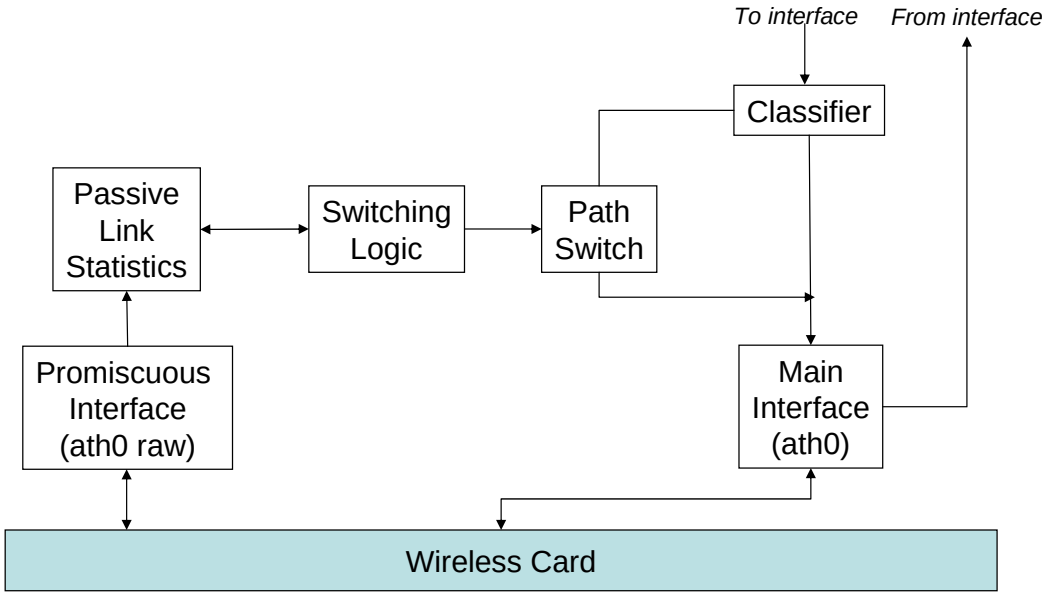


Figure 2.7: Block diagram describing operation of Deflect beneath the routing/packet forwarding layer.

Some implementation choices have been made for simplicity. In order not to overload a cooperative relay node of monitoring multiple number of two hop paths, we restrict relay nodes to monitor at most 2 paths at a time. In general, more than one cooperative relay can exist for a single 2-hop path. Thus, multiple such cooperative relays can signal the source node to take over relaying. In our implementation, the source node simply accepts the first one that volunteers and ignores the rest.

2.4 Experimental evaluation

We have conducted several experiments to evaluate the performance benefits of Deflect compared to a conventional routing protocol using the ETX metric. Experiments were conducted on an indoor mesh network testbed consisting of 20 nodes deployed in NEC Labs. Each node is a Routerboard 200 series processor board [98] with 256MB of RAM, 512MB of Compact Flash for secondary storage, and an Atheros-based 802.11a/b/g card. 802.11b mode is used for the experiments reported here. The ETX metric for each link was collected by existing software elements (`ETXMetric` and `LinkStat`) that are part of the Click Toolkit. The evaluation is done in two parts – (i) Evaluating the accuracy of the passive estimation technique; (ii) Characterizing the performance of the path switching mechanism in terms of its reaction time and packet reordering, and resultant benefits on metrics such as delivery ratio, delay and jitter.

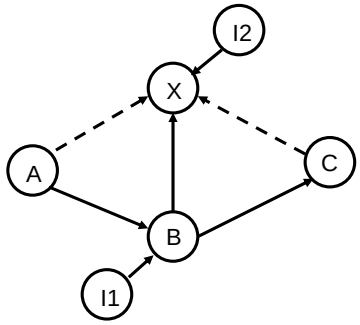


Figure 2.8: Experimental setup to demonstrate effective estimation of link quality. The interferers $I1$ and $I2$ cause packet errors primarily at B and X respectively.

2.4.1 Evaluation environment

In order to investigate and characterize the performance of link estimation and path switching in Deflect, we used a controlled environment for generating experimental scenarios. In order to avoid unpredictable effects of external radio sources, all experiments were carried out during night. One node (called the *interferer*) generates interference traffic on a target link. The interference traffic is generated in *bursts* – with alternating on and off periods. The *duty cycle* (the relative duration of on and off periods) is fixed, but the absolute duration of on and off periods is varied to control the dynamic nature of the interference. The interference load (in Kbps) during the on period is also varied. Finally, the location of the estimator node (cooperative relay) and transmit power level of the interferer are also varied between experiments. The interferer operates on an overlapping channel because our experience has shown that this makes the interferer more likely to succeed on CCA (clear channel assessment), while causing collision at the receiver.

2.4.2 Accuracy of Link Quality Estimation

Deflect primarily works by passively estimating link qualities of neighboring nodes. We have performed an extensive set of experiments to evaluate the accuracy of the estimation technique in various set ups. For brevity, we present a careful set of evaluations focusing on four nodes in our testbed (in Figure 2.5), acting as nodes A, B, C, X. The nodes are $A = rb$-7, $B = rb$-15, $C = rb$-3, and $X = rb$-8.

In order to provide illustration of the passive estimator in action, Figure 2.9 shows an example 6 sec trace of packet delivery ratios on link AB and its estimate at X (i.e., the quantities d_{AB} and d^X_{AB}) with sampling time window $H = 100ms$. In this case, there are two interferers in action. Interferer $I_1 = rb$-12 generating interference on node B primarily, and interferer $I_2 = rb$-5 generating interference on node X primarily (Figure 2.8). The traces in Figure 2.9 also show the estimated

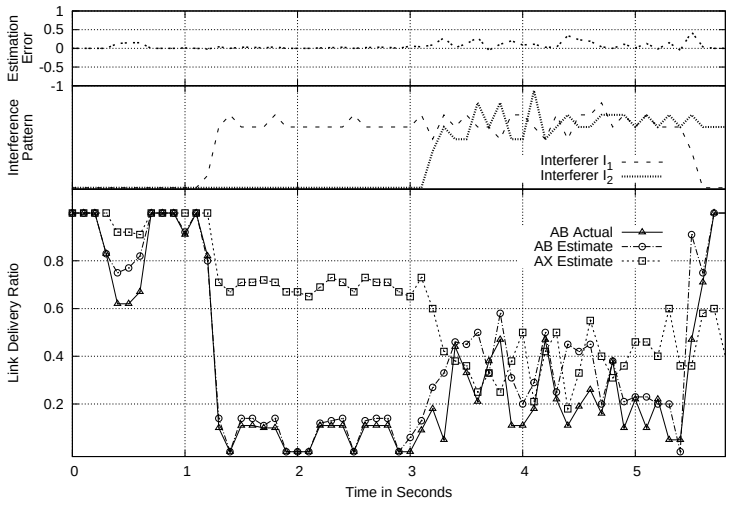
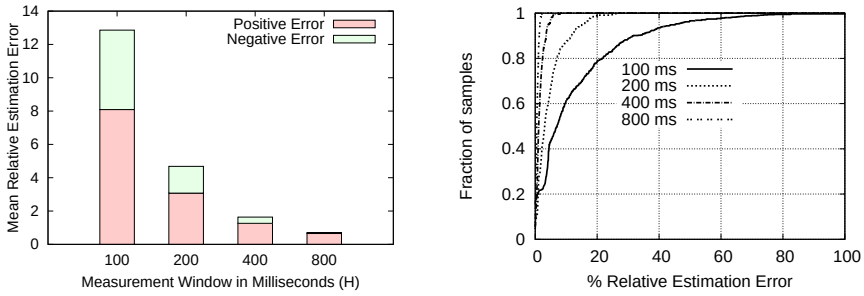


Figure 2.9: Trace demonstrating the potential benefit of path adaptation and accuracy of link quality estimation.

delivery ratio on link AX (i.e., d_{AX}^{X}). The behavior of the two interferers I_1 and I_2 are shown in terms of number of packets transmitted. Note that there are times when only one interferer is on and also times when both interferers are on. Figure 2.9 shows that actual and estimated delivery ratios on the AB link are very close, and the estimated delivery ratios for AB and AX links follow the interference patterns, with delivery ratios going down when the respective interferers are on.

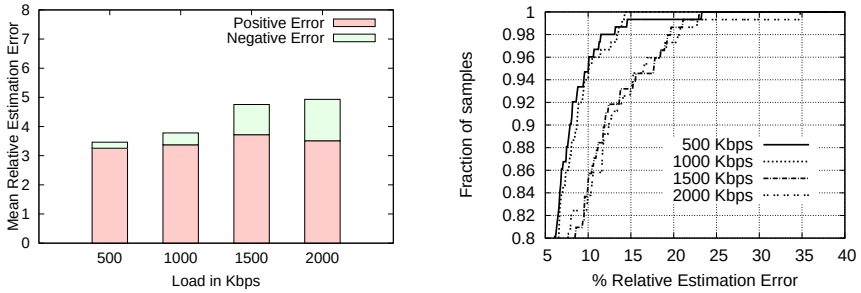
In the next set of experiments, we study the estimation of the quality of link quantitatively. As a start, we separate two types of errors: positive ($d_{AB}^{X} < d_{AB}$) and negative ($d_{AB}^{X} > d_{AB}$). In Deflect negative error is less desirable as it may result in unnecessary path switching (false positive decision). We compute the relative estimation error (as a percentage) as $100.0 \times [(d_{AB}^{X} - d_{AB})/d_{AB}]$ % in each sampling time window period (H), and present statistical data (mean and CDF) for the error. When we present mean relative estimation error, we not only present the mean of the absolute value of the error, but also separate out positive and negative components to show their relative contributions. When we plot CDF, we again plot the CDF of the absolute error.

Effect of sampling time window (*H*): We consider only interferer I_1. The interferer is on for $w = 100$ms and off for 300ms (duty cycle = 25%). When it is on, it transmits broadcast packets at 2Mbps. Figure 2.10(a) shows the mean relative estimation error for different values of H. Figure 2.10(b) provides a more detailed information related to the distribution of the error for different H. We observe that the mean error reduces from 13% to 5% by increasing H from 100ms to 200ms – a significant drop. It drops further for larger H. Increasing H further reduces the error, but also makes the path switching less reactive. A value of 200ms for H is thus a good tradeoff. The 5% estimation error can be compensated by choosing a larger threshold T that triggers the path switching.



(a) Mean (b) CDF

Figure 2.10: Effect of sampling time window (H) on relative estimation error (%).



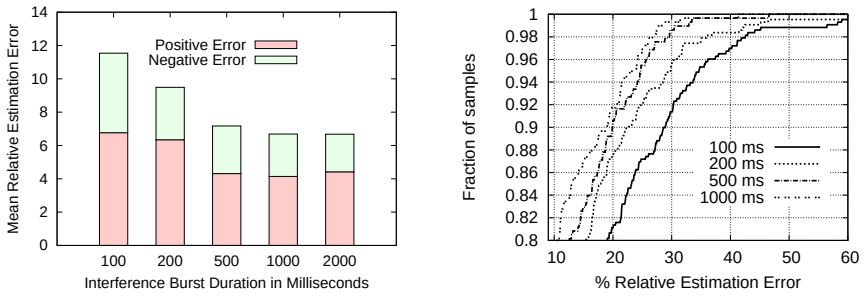
(a) Mean (b) CDF

Figure 2.11: Effect of load on AB on relative estimation error (%).

Effect of load on link *AB:* High load may result in relatively higher fluctuation of the actual delivery ratio on $A - B$ for a given interference pattern. A possible reason can be the MAC unfairness at higher load. Higher fluctuation will increase the error in estimation. To understand the effect, we vary the load from 500Kbps to 2Mbps (limited by the effective throughput of a 2 hop path with 802.11b card). Figure 2.11(a) and Figure 2.11(b) show the dependency of the relative estimation error on load. Even though the graph shows an increasing trend for the error with load, we note that change of mean error is less that 4% for an increase of load by a factor of 4 (0.5Mbps to 2Mbps).

Sensitivity to time varying interference: We generate different interference patterns by changing the on and off periods, but keeping the duty cycle constant at 25%. Smaller on or off periods result on more fluctuation of the interference load and thus of the delivery ratio. This experiment is designed to understand how fast the estimator can catch up with the actual delivery ratio when the latter fluctuates over time. We use the lowest sampling time window ($H = 100ms$). Figure 2.12(a)

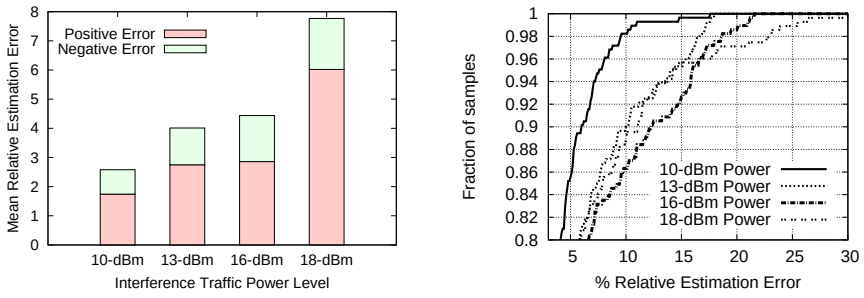
and Figure 2.12(b) show the dependency of relative estimation error for different on (burst) periods. We note that the mean error ranges from 6% to 12%. From Figure 2.12(b), we observe that around 80% of the samples have less than 15% error, when the on period is 200ms or more. These results together with the results in Figure 2.10(a) suggest that the passive estimation can react in the granularity of 200ms or more.



(a) Mean (b) CDF

Figure 2.12: Relative estimation error (%) in time-varying interference pattern.

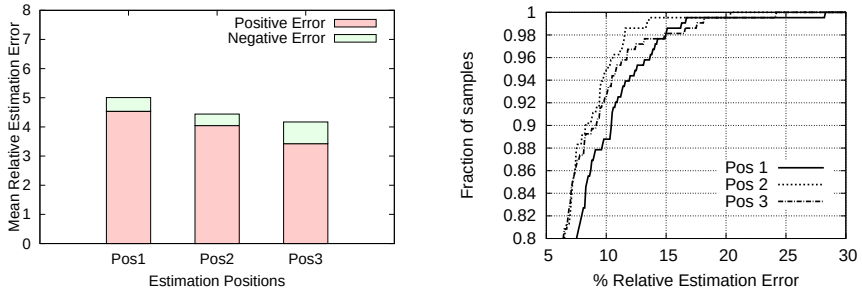
Sensitivity to interference level: Here, we vary the interference level by setting different transmit power levels of the interferer I_1. The goal of this experiment is to understand the dependency of the error on the interference power. From the Figure 2.13(a) and Figure 2.13(b), we note that increasing interference power increases the error. This is likely because the increasing power also results in interference (*albeit* weaker) at the estimator node X. Interference at X results in reduction in the number of samples and/or distorting the samples collected leading to increase in error.



(a) Mean (b) CDF

Figure 2.13: Relative estimation error (%) for different interference levels.

Sensitivity to the location of the estimator node: In this experiment, we try to find if the estimation error depends upon the location of the estimator. For the two hop path $A - B - C$ considered in the mesh testbed, we have three nodes that can act as cooperative relay X. Figure 2.14(a) and 2.14(b) show the estimation error for these three nodes. We note that error is almost the same for all the three nodes. The difference is less than 1%. Figure 2.14(b) shows that the error distribution in the samples is similar for all the three locations as well.

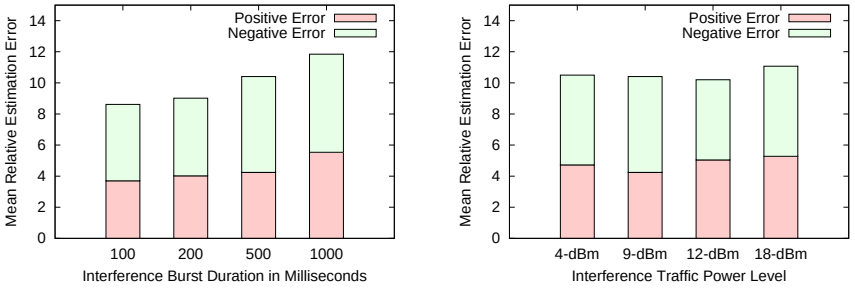


(a) estimation window (b) estimation load

Figure 2.14: Relative estimation error (%) for different choices of X.

Summary of results: From the above pathological cases studied in the experiment, we can make the following important conclusions: a) with the right choice of parameters the link quality estimation is quite accurate even for a wide variety of interference conditions; b) the negative errors are usually smaller (often much smaller) than positive errors, preventing the protocol from over-reacting; c) estimation error is not dependent on the choice of the estimating node (cooperative relay), providing a great diversity of relay choices; d) a sampling time window as small as 200ms is reasonable to keep errors low (less than 5% overall, with negative error less than 2%).

Estimating Link *AX:* So far, we concentrated on link AB only. Estimating link BC is similar, as explained in Section 2.3.1. We also present some results related to link AX. Figure 2.15(a) shows the relative estimation error for different on (burst) periods of the interferer, where the sampling time period H = 100ms. We observe now that the error increases with increase in burst period. This trend is different from what we have seen before for link AB. This is because with more stable interference, packet collision probabilities are stable for longer durations of time. This presents less samples to the estimator in certain periods. Changing the power level of interferer in this case seems to have little impact on the estimation error (Figure 2.15(b)). In general, we note that estimation errors are generally small, often under 10%.



(a) Interference duration in ms (b) Interference power level in dB

Figure 2.15: Relative estimation error for link AX.

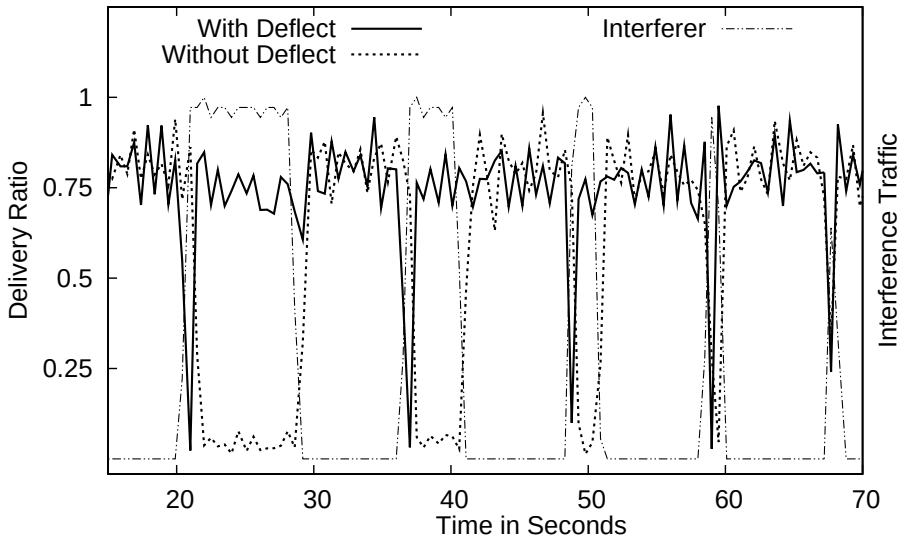
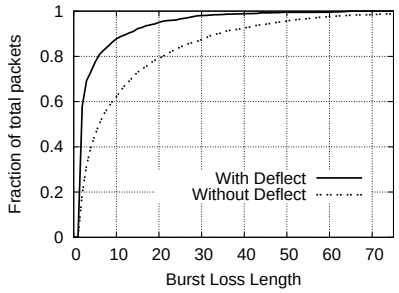


Figure 2.16: Trace demonstrating the benefit of Deflect in presence of time varying interference.

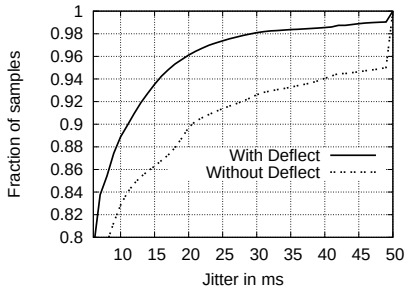
2.4.3 Evaluation of Path Switching

In order to better understand the benefits, we use an implementation of the OLSR routing protocol [32] using ETX as the link quality metric. Then, we run Deflect underneath routing. An illustration of the benefit of using Deflect is in the trace given in Figure 2.16. When the interferer is on, it transmits back-to-back broadcast packets. The trace uses interference bursts that are usually of a shorter duration than the ETX evaluation period (10sec). We notice that with Deflect, delivery ratio is quite stable due to path switching.

For a more quantitative characterization of benefits, we focus on four performance measures important for streaming/interactive media: *burst loss length, jitter,*

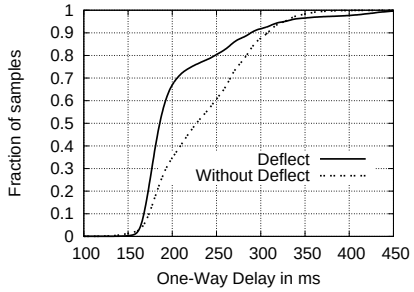


(a) Burst Loss (in packets)

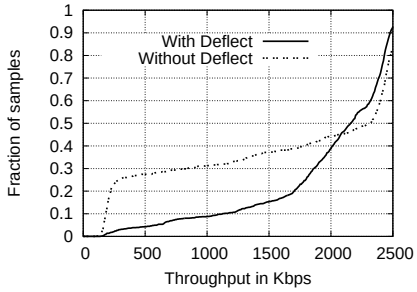


(b) Delay Jitter (in ms)

Figure 2.17: CDFs of burst loss length and delay jitter with and without Deflect.



(a) One-way Delay (in ms)

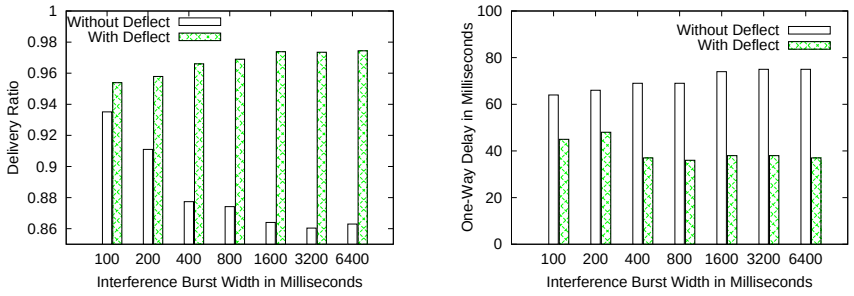


(b) Instantaneous Throughput (in Kbps)

Figure 2.18: CDFs of one-way delay and instantaneous throughput with and without Deflect.

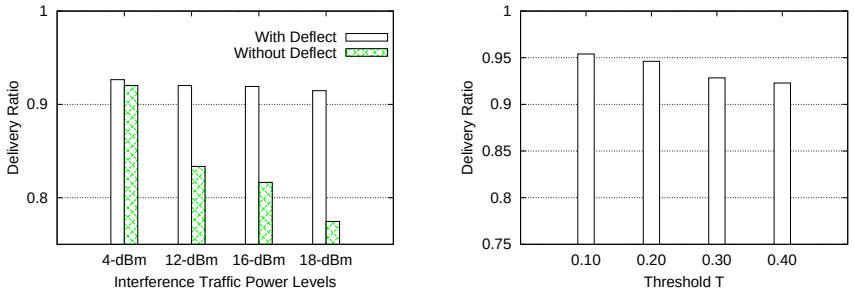
one-way delay and *throughput*. Figure 2.17(a) shows that at the 80-percentile point Deflect reduces the burst loss length to about one third. The median value is also less than half. Figure 2.17(b) shows less jitter as well with Deflect, though the improvement here is less dramatic. In Figure 2.18(a), we observe similar benefits in terms of one-way delay. In particular, 70% of the samples has delay less than 200ms using Deflect compared to only 34% of the samples without using Deflect. Figure 2.18(b) shows the throughput for routing with and without Deflect, where again we observe that almost 30% of the time, throughput without Deflect is below 500Kbps. In general, we get many more low throughput samples when Deflect is not used.

Reaction speed of Deflect: To study how fast Deflect can react to changing interference, we conduct a series of experiments by using on-off interference as in the previous subsection, and vary the on (burst) period of the interferer. The



(a) Delivery ratio (b) One way delay

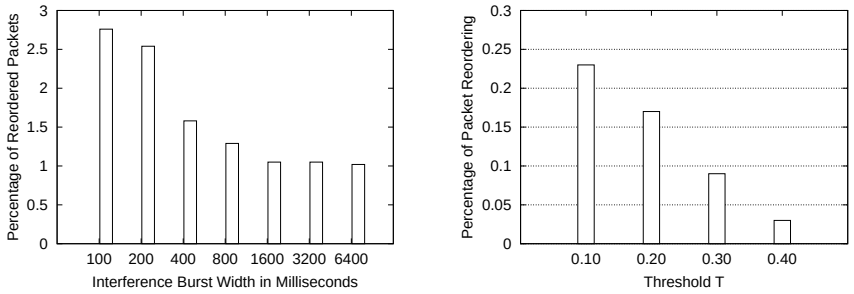
Figure 2.19: Performance of Deflect with different interference burst sizes.



(a) Delivery ratio vs interference power.

(b) Delivery ratio vs T.

Figure 2.20: Performance of Deflect with different interference power and T values.



(a) Varying burst size.

(b) Varying T.

Figure 2.21: Amount of Packet reordering.

interferer, when on, transmits back-to-back broadcast packets. We keep the duty cycle the same as before at 25% so that the total amount of interference is kept constant. Figure 2.19(a) shows the delivery ratio with and without Deflect. Lower burst duration implies a faster changing interference pattern. From the figure, we observe that even for burst size of 100ms, Deflect provides some benefit. With 200ms and more, the gain is significant. A similar trend is observed from one-way delay as shown in figure 2.19(b). As one would expect that Deflect's ability to react fast also results in some packet reordering. We present the percentage of reordered packets in Figure 2.21(a). We observe that the percentage of reordered packets stays quite low, between 1% to 2.8%. We observe only very minor reordering without Deflect.

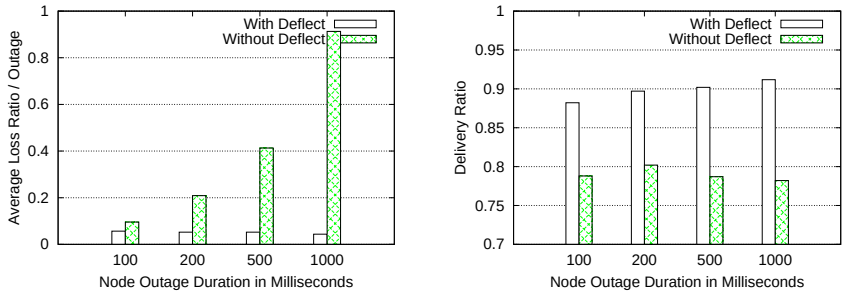
Effect of interference power and threshold T: The delivery ratio also depends upon the interference level. Figure 2.20(a) shows this. It also shows that Deflect is robust against power. Delivery ratios for Deflect with varying threshold T is shown in Figure 2.20(b). Note decreasing performance with increasing threshold. Reordering also changes with T (Figure 2.21(b)), with more reordering will lesser T as path switches are more frequent.

Effect of Link Outage: In our mesh testbed deployed at NEC labs, we found frequent occurrence of mesh nodes going down. We attribute this unstable behavior to exceptions occurring in the embedded hardware and bugs in the software. Deflect can detect such outages and quickly reroute the traffic via an alternative path. To study the reaction time of Deflect, we simulated node outages for short intervals of time by shutting down the wireless card on the node. Figures 2.22(a) and 2.22(b) show the performance of Deflect for different outage periods.

Ambient interference: Here, instead of using a controlled interference pattern, we study the impact of ambient wireless interference on routing with and without Deflect. To study this, we ran CBR UDP traffic on several isolated 2-hop paths in our mesh testbed during working hours, when a large number of WLAN APs and clients operate in the vicinity of our mesh testbed. We ran the experiment for 8 hours and collected statistics based on delivery ratio samples collected every 2 seconds. Figure 2.23(a) demonstrates the gain with Deflect in the presence of such ambient interference. Figure 2.23(a) shows a 3 second trace showing instantaneous throughput with samples collected at more frequent intervals.

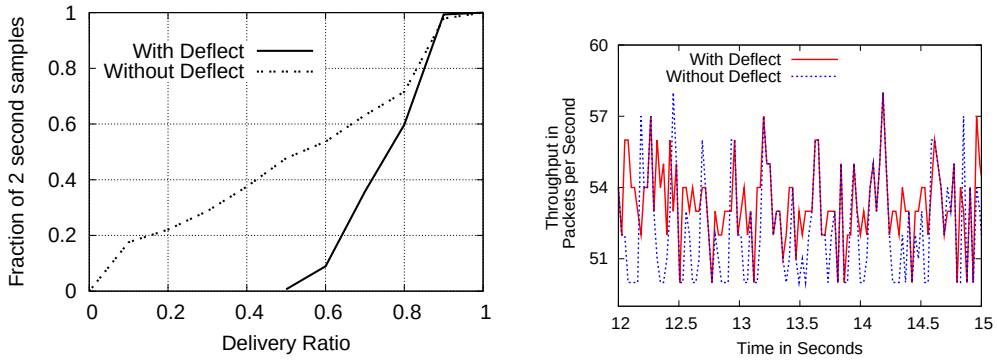
2.4.4 Performance of Real Time Applications

We have also studied the benefit of Deflect for supporting VoIP in multihop mesh networks. For the G.729 voice codec used in the experiment, the VoIP quality is specified by the R-score [33]. R-score is determined based on loss (from network and jitter buffer) and delay using the formula given in [82]. A higher R-score indicates a better VoIP quality. VoIP traffic for G.729 is generated using a stream of 60 Byte UDP packet sent every 20ms. We generated 4 VoIP call in the experiment



(a) Loss ratio per outage (b) Delivery ratio

Figure 2.22: Performance of Deflect under link outages.



(a) CDF for delivery ratio (b) Sample trace

Figure 2.23: Performance of Deflect under ambient interference conditions.

and created a interference pattern as shown in Figure 2.24. Figure 2.24 shows that Deflect switches paths as the interference turns on within a few seconds (with occasional noisy changes). If Deflect is not used, routing sticks to the same path (path 1). The R-score with Deflect stays close to 80, while without Deflect, R-score fluctuates with the interference, often going down to 0.

For our second application, we use mpeg4 streaming of the commonly used Foreman video. To measure the video quality we use the Peak Signal to Noise Ratio (PSNR) as the video quality metric. For every frame in the video, the PSNR value was computed and compared with the Reference PSNR (the PSNR of the frame in the original video). Results on PSNR was collected based on a segment of the video where several random interferers are turned on in the neighborhood only for the first half of the video segment. The Figure 2.25 shows the PSNR for each frame with and without Deflect, while comparing it with the reference PSNR.

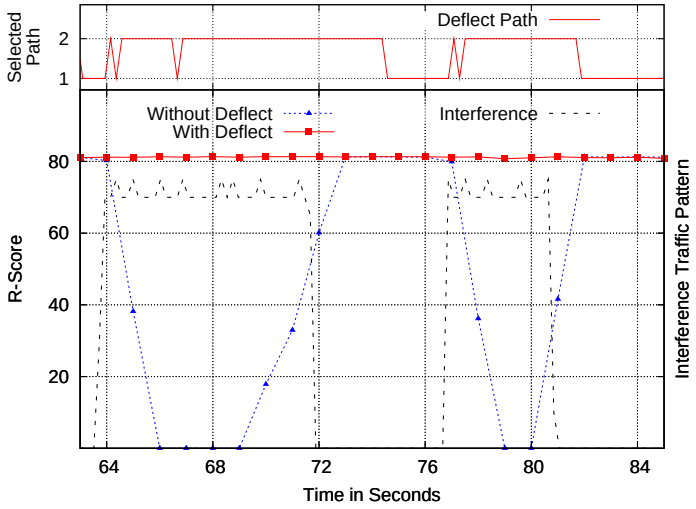


Figure 2.24: Sample trace showing the performance of VoIP calls with and without Deflect.

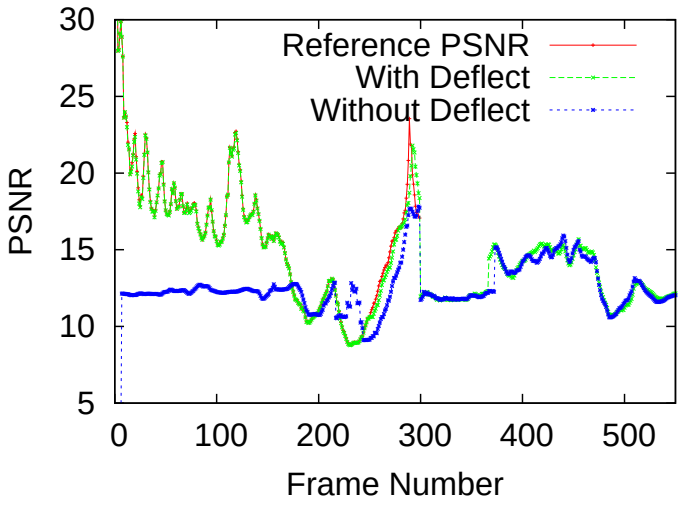


Figure 2.25: Sample trace showing the PSNR values for a video stream with and without Deflect.

We note that for the first 200 frames, there is a significant difference between the PSNR without Deflect and with Deflect. Over the whole span, the reference PSNR is almost closely followed by using Deflect.

Load in link AB (Mbps)	Max UDP throughput (Mbps) at X	
	w/o snooping	with snooping
0.00	10.0	10.0
10.0	6.40	6.11
20.0	5.67	5.51
28.0	4.76	4.24

Table 2.1: Impact of snooping on achievable throughput

2.4.5 Snooping Overhead

Deflect is based on snooping in promiscuous mode. This causes an additional CPU overhead on the cooperative relay nodes. We conducted a set of experiments on our Routerboard nodes to find out how much impact this overhead has on the forwarding capacity of a relay node. To do this, we increased the load on the link AB in order to increase snooping overhead on part of X. In 802.11b we observed no change in the forwarding capacity at X even when link AB carries saturation traffic. It turns out that 802.11b link rate is not high enough to cause any CPU bottleneck. To get higher link rate, we switched the interface to 802.11g. For this case, the table 2.1 shows the maximum forwarding capacity at X with and without snooping. There is indeed a small reduction in capacity when snooping load is high. But we do not consider this to be a significant issue as the CPU power in embedded router systems is steadily increasing.

2.5 Related Work

There has been a significant interest in exploiting link and path diversity to improve performance of wireless networks. Two recently proposed approaches – *Divert* [74] and *ExOR* [26] – are most related to our work. In Divert [74], fine-grain path selection is used in a wireless LAN environment. If the link between a client and the access point (AP) degrades according to a short-term frame delivery statistics, an alternative AP is used for communication without causing an actual handoff. Deflect unlike Divert, works in a multihop network, and does not rely on a wired distribution system for coordination between nodes. The uniqueness of Deflect is in its sophisticated passive link estimation technique, the absence of which makes path adaptation difficult in mesh network.

ExOR [26] uses an opportunistic forwarding framework in a multihop network, where packets at each hop are broadcast instead of unicast. A coordination protocol is used to determine which node – among those that received the packet correctly – will actually forward the packet further towards the final destination. Such coordination

is needed to ensure that more than one node do not end up forwarding the same

packet. This coordination could be complex and expensive in a dense network, where there could be many potential forwarders. ExOR implements this coordination via a gossiping protocol that works by breaking down the packet transmissions in batches, and including a state of successful packet transmissions in packet headers. Because of the nature of the gossiping protocol, ExOR is useful only for bulk transfer applications, and not for real-time applications that Deflect targets. Also, the scalability of the gossiping protocol is unclear, as it requires per-flow state maintenance.

Multipath routing protocols [68, 80, 86] require the explicit computation of multiple paths during route computation, and can rank them in quality based on a chosen routing metric.

Such protocols essentially provide the packet forwarding layer with multiple next hop alternatives. The packet forwarding layer can now switch paths when packet losses occur in a next hop link. Here, the forwarding node must switch paths to an alternative next hop *without any knowledge of the instantaneous condition of the potential alternative next hops*. This may boil down to a blind search for the best alternative. In contrast, Deflect does not explicitly search for alternate paths. Cooperative relays in the neighborhood of the original path simply estimate whether they can offer better alternatives, and they – instead of the forwarding node – take the initiative of path switching. No blind search is needed.

Some routing protocols for mobile ad hoc networks attempt to improve on path lengths by snooping on the radio medium. Examples include DSR protocol [66, 51] and SHORT protocol [45]. These protocols are primarily directed for quickly "shortening" paths in a mobile network, where some nodes may have stale routing information, but can overhear more up-to-date information providing better routes. Our path adaptation approach is also based on snooping, but it it is used to estimates link quality metrics directly.

A number of MAC-layer protocols have been developed to determine the forwarding hop in a multihop network based on instantaneous channel condition. Selection diversity forwarding [62], and MAC-layer anycasting [48, 99] are based on this idea. The GeRaF protocol [126] is also similar, but is only applicable when geographic routing is used. A number of similar protocols work in the context of wireless LANs. Prominent examples include the MAD (medium access diversity) protocol [49] and opportunistic packet scheduling protocol [120]. They exploit multiuser diversity at the AP in the AP-to-client communication by probing channel conditions at multiple clients and giving certain preference to clients with good channel conditions. All these approaches require a new MAC layer. In contrast, Deflect works on existing 802.11 MAC and commodity radios.

Packet combining techniques have been used in literature to obtain diversity. In the MRD (multi-radio diversity) technique [73] multiple radios are simultaneously used to receive the same transmission. This technique is targeted for wireless LAN scenarios. For uplink diversity, this can be done using more than one AP that has

overlapping coverage. For downlink diversity, more than one radio interface can be used in the client. When multiple copies of the same frame are received with errors by the different interfaces, they can be combined to recover from the error.

Finally, communications and information theory community have been looking at the concept of cooperative diversity for some time. The idea is somewhat similar to packet combining as above, but uses relaying of the received signal in the physical layer. See, for example, [109, 101]. Much of these ideas have so far been in the realm of theory, because of restrictive assumptions on channel models, functionalities at the physical layer, synchronization, and traffic behavior.

2.6 Conclusions

In this work, we designed and implemented Deflect, a fine grained local path adaptation mechanism that works underneath the routing layer. Deflect provides a zero overhead mechanism for estimating link qualities, allowing link quality probing to be done at a fine granularity of 200ms. Using the link quality estimation, Deflect enables fast local adaptation of end-to-end routes in response to sudden drop in link delivery ratio or node failures. Our experiments shows that the error in estimation is less than 5% with 200ms probe interval. Deflect can provide a higher delivery ratio in presence of time varying interference. Deflect is particularly useful for streaming applications where short term fluctuations in link qualities can severely degrade performance. Additionally, our results also verify that a) the packet snooping overhead does not lead to sacrificing the forwarding capacity of the relay node and b) the packet reordering from path switching in Deflect is less than 4%.

Chapter 3

Slotted Symmetric Transmit Power Control in WLANs

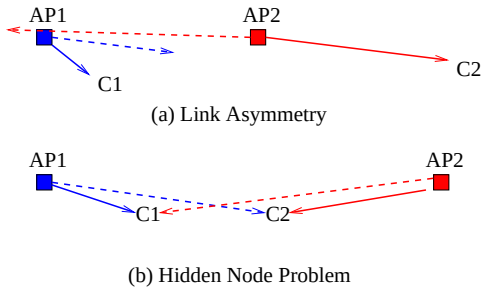
3.1 Introduction

This chapter explores the impact of dynamic power management for spatial reuse in *managed* wireless LAN deployments— the word *managed* refers to deployments in which all access points (APs) are under the same administrative domain. Such managed deployments are common today at multiple scales, from small-range corporate networks [39] and campus-wide networks [3], to hot-spot networks such as T-mobile [11], and city-wide muni-wifi networks such as Google's Wifi [4], MIT's roofnet [25], and Houston's urban network [29].

To provide maximal coverage, these wireless networks often contain multiple APs with overlapping transmission and interference ranges. Greater overlap leads to increased contention to transmit and increased packet collisions, thereby reducing the overall network throughput.

As WLANs become ubiquitous and users adopt the wireless medium as their first-class last-mile access network, network administrators are forced to increase the capacity of their wireless networks. Unfortunately, the capacity of an 802.11 network is limited due to the interference among network nodes and the availability of limited number of channels. Consequently, several research efforts are being focused on mitigating interference as much as possible to get maximum throughput out of these networks. Two major directions for capacity improvement have been toward exploiting channel diversity through efficient channel allocation [16] and spatial reuse through power control [15] or use of directional antennas.

In this chapter, we propose fine-grained transmit power control for increasing spatial reuse by tuning the transmit power of each AP on a per-client basis. Our motivation is that clients closer to an AP can successfully receive packets even when the AP transmits at a lower power than that needed by a farther client. From a practical standpoint, such a proposition is made possible by the availability of



(a) Link Asymmetry

(b) Hidden Node Problem

Figure 3.1: Challenges with dynamic power management. Solid lines indicate transmission and dotted lines indicate interference.

devices with multiple power levels that can achieve per-packet power control with minimum-overhead.

Unfortunately, we observe that while such fine-grained control promises to substantially increase spatial reuse, it can lead to the formation of *asymmetric links* that can degrade the throughput of some clients. The result is an unfair allocation of network resources with the possibility of complete starvation of certain clients. For instance, consider Figure 3.1(a), where AP1 chooses a smaller power to transmit to C1 than AP2 that is simultaneously transmitting to C2. While AP1 can carrier-sense AP2, AP2 might not be able to do so. As a result, the link from AP1 to C1 starves in the presence of AP2 $\rightarrow$ C2.

To this end, the goal of this work is to devise a power control mechanism that is able to increase spatial reuse without sacrificing fairness; we propose a novel *slotted symmetric* power control framework. The main idea underlying this framework is to have all access points operate at the same power level at any given instant of time and have all access points follow a sequence of power levels synchronously. Time is divided into slots, and in each slot all access points operate at same power level, thereby avoiding any link asymmetry. Through hopping over different power levels across successive slots, all access points are able to serve the clients at their corresponding minimum power level requirements. Both simulation based evaluation as well as experimentation with prototype implementation shows that the slotted symmetric power control simultaneously improves the network throughput and provides better fairness to clients.

In summary, our key contributions are as follows:

- We develop a power control technique, Contour, based on the slotted symmetric power control framework. Contour is designed to adapt to traffic load and arrival/departure of clients. Contour's design ensures that there is no inter-AP messaging and that packet level scheduling is done at APs without involving any centralized entity.

- We compare Contour using ns simulations with (a) baseline CSMA/CA

without power control, (b) static power control, and (c) unslotted dynamic per-packet power control. Our results show that removing link asymmetry in Contour leads substantially improved fairness compared to the above schemes. For instance, whereas schemes (b) and (c) above improve the throughput of 50-60% of the clients over CSMA and *hurt* the throughput of the rest of the 40-50% clients, Contour improves the throughput of 95% of the clients and hurts only 5%.

- We implement the proposed Contour dynamic power control technique in Click [57]. We evaluate the prototype for throughput improvement and fairness on a real testbed. Our evaluation shows that a capacity improvement of up to 60% is achievable by employing Contour. To the best of our knowledge, this is the first such implementation and evaluation of a fine-grained synchronous slotted framework in WLANs.

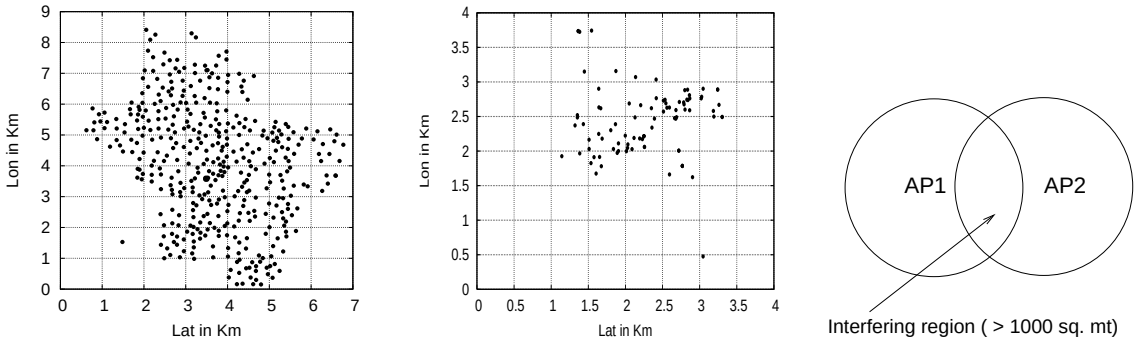
The rest of the chapter is organized as follows. Section 3.2 presents the problem formulation by discussing the case for power management, the enabling technology and the challenges in realizing dynamic power management. Section 3.3 discusses the design of Contour, and Section 3.4 presents extensive simulation results to demonstrate the efficacy of Contour. Section 3.5 discusses important implementation details. Section 3.6 demonstrates through prototype evaluation the efficacy of Contourin improving spatial reuse. Section 3.7 identifies the outstanding issues and future directions of this work. Section 3.8 places Contour in the context of related work, and Section 3.9 concludes.

3.2 Problem Formulation

In this section, we discuss the need for, and the feasibility of performing dynamic power management for spatial reuse in wireless LAN deployments. We then identify the challenges in realizing dynamic power management that will guide the design of Contour.

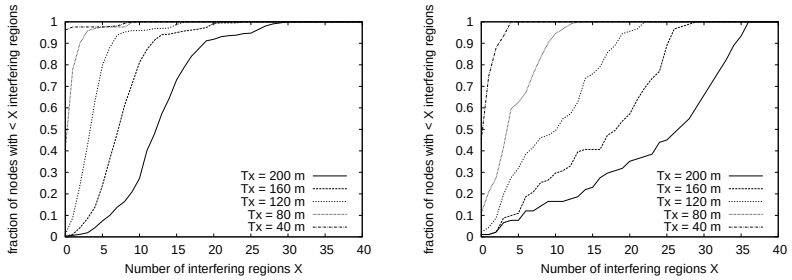
3.2.1 Case for Dynamic Power Management

WLAN technology has become an attractive building block for designing last-mile access networks of varying scales. Examples of such networks include small-scale corporate networks and campus-wide networks to large-scale city-wide networks. Such networks often involve dense deployment of access points in the interest of providing coverage to clients; dense deployments in turn lead to interference between access points that can result in sub-optimal network performance. In such networks, dynamic power control can mitigate interference while not compromising on coverage, and hence increase the number of simultaneous AP↔Client transmissions. Such spatial reuse has two consequences—(1) for the same number of clients,



(a) Google's Wifi (b) MIT's Roofnet (c) Interference Region

Figure 3.2: (a) and (b) show the distribution of APs in Google's wifi and MIT's Roofnet. (c) depicts the interfering region between two APs.



(a) Google's Wifi (b) MIT's Roofnet

Figure 3.3: CDF of interfering regions in Google's wifi and MIT's Roofnet.

the average per-client throughput can be increased, and (2) for the same per-client throughput requirement, the number of clients supported can be increased.

Tx Level	Tx range (meters)	Intf. range (meters)
5	200	300
4	160	240
3	120	180
2	80	120
1	40	60

Table 3.1: AP transmit power model for determining interfering regions.

To demonstrate that instances with significant overlap in interference ranges already exist, we study two managed networks that provide Internet access on a city-wide scale—Google's Wifi [4] and MIT's roofnet [25]. Figures 3.2(a) and 3.2(b) show the distribution of APs in Google's wifi and MIT's roofnet respectively. For each of the networks, we study the amount of overlap in interference ranges between different APs of the network. For this experiment, we use a simple system model of an AP as shown in Table 3.1; each AP has maximum transmit and interference ranges of 200 and 300 meters respectively, and supports 5 power levels. At lower power levels, the transmit and interference ranges decrease proportionally. Further, we consider that the interference ranges are circles. We define two APs to have an interfering region if the overlapping area of their interference ranges at any transmit power level is greater than 1000 square meters (see Figure 3.2(c)). The number of interfering regions for an AP influences the throughput achieved by the AP; an AP interfering with k other APs will get a chance of at most $\frac{1}{k+1}$ times to serve its clients in the region.

Figure 3.3(a) shows the CDF of interfering regions in Google's wifi when APs transmit at different power levels. At the maximum power level corresponding to 200 meters of transmit range, more than 90% of APs have 5 or more interfering regions. However, at power level 1, only 5% of the APs have 5 or more interfering neighbor AP. This graph suggests that if APs can choose a lower power level when possible (i.e., when the client they are serving is in closer transmit range), the interfering regions can be substantially reduced. This increases the number of simultaneous transmissions between APs and their close-by clients. For the farthest clients, an AP may still need to choose maximum power level, thereby behaving like existing networks with no power control. We observe similar benefits in roofnet, as demonstrated by Figure 3.3(b).

We envision that these deployments will only become denser in the future for two reasons. First, providers may want to provide complete coverage even in the presence of unavoidable obstacles that attenuate signal strength significantly; providers often place an extra AP near such obstacles [4, 29]. Second, developing techniques for better spatial reuse will, in turn, incent denser deployments to increase the throughput for each client.

3.2.2 Enabling Technology

Technology trends suggest that wireless interfaces are increasingly providing fine-grained power control of packet transmission, hence making dynamic power management in WLANs both viable and useful. For instance, Figure 3.4 shows the power levels (in terms of signal to noise ratio) observed by a client at different distances from an AP that is transmitting packets at different transmit power levels (as indicated by the x-axis). The AP uses an Atheros [2] chipset AR5212 802.11a/b/g PCMCIA card for the top line and an Atheros MiniPCI card with the same chipset

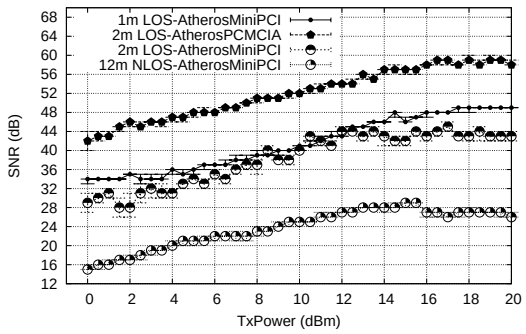


Figure 3.4: Power levels observed by a client at various distances from an AP.

for the bottom three lines; we use the Linux MadWifi [64] device driver, version 0.9.2 with these cards[1].

The top line in the graph indicates that the client (in line-of-sight with the AP) can perceive different signal strengths at a fine granularity (for example, every 2dBm change in transmit power level) when the AP transmits at different power levels. That is, an AP can transmit at 8 discrete power levels at a granularity of 2dBm that have perceivable difference at the client. We make similar observations with the MiniPCI card, for different distances of the client from the AP, and when the client is not in line-of-sight with the AP. Further, our measurements show that these cards support per-packet power control with minimum overhead; i.e., each packet can be transmitted at a different power level. In summary, the graph suggests that technology already exists, and would only improve, to enable APs to select the minimum transmit power for each client that is sufficient for successful transmission of packets.

3.2.3 Challenges in Dynamic Power Management

While per-client power management can improve spatial reuse, it can also can suffer from two potential problems—link asymmetry and hidden nodes. To illustrate the problems, we consider a small network of four nodes—two APs communicating with two clients using the same channel. We identify the different scenarios (see Figure 3.5) of how the AP to client communications might interfere (Similar analysis for a multi-hop wireless network was done by Garetto et. al. [41]). In each scenario, an arrow from node A to node B represents that packets originating from A reach B. Bi-directional arrows indicate that both nodes can hear each other. Dotted arrows indicate that the presence or absence of the arrows does not matter. For

[1]To enable transmit power control (TPC) on these cards, we recompiled the MadWifi driver with TPC feature enabled.

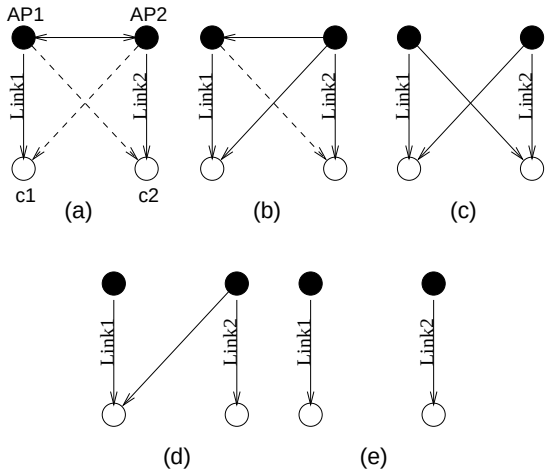
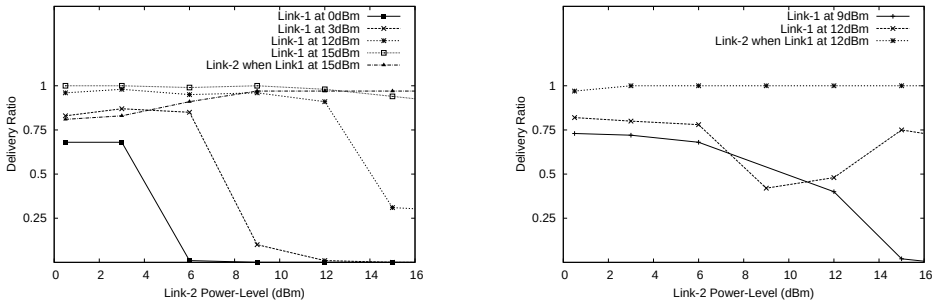


Figure 3.5: Scenarios of interaction between two AP→Client transmissions.



(a) Configuration 1 (b) Configuration 2

Figure 3.6: Unfairness and under-utilization due to asymmetric link and hidden node problems introduced by dynamic power management.

instance, in scenario (a), since AP1 and AP2 can perceive each other's transmissions, only one of them will transmit at any instant of time. So AP1's transmission does not affect C2's reception of AP2's packets.

Depending on the distances between the nodes, the network of APs that always transmit at the same power to their clients can be in any of the scenarios (a), (c), (d) and (e) in Figure 3.5. While (a) ensures that the channel is shared equally between the two transmissions, (e) ensures that the transmissions can happen simultaneously. Scenarios (c) and (d) represent two instances of the hidden node problem; while APs are oblivious of each others' presence, packets sent by them collide at their clients. In a network performing per-client power management, even a subnetwork originally in scenario (a) may get converted to one of (c), (d) and (e). Further, power management introduces an extra scenario of link asymmetry—Scenario (b), where only one AP can sense the other's transmission and not *vice versa* thereby leading to unfair sharing of the channel.

Link asymmetry and hidden node problems can degrade the throughput and fairness provided by the original network. To demonstrate their affect, we study two different four-node configuration containing two APs and two corresponding clients; the configurations differ in the placement of the clients relative to the AP locations. We plot in Figure 3.6, the *delivery ratio* observed by the two AP→Client links in each of the configurations when the transmission power of each link is changed. The delivery ratio is calculated as the ratio of packets successfully received at the client to those sent by the AP. In Figure 3.6(a), the line "Link-1 at 0dDm" shows that Link-1 observes a good delivery ratio when Link-2 is also at a low power level. As Link-2's power increases, "Link-1 at 0dBm" approaches zero delivery ratio, thereby demonstrating the problem with asymmetry (i.e., the system moved from scenario (e) to (d)). The same observation can be made from the other lines at 3dBm and 12 dBm. The lines "Link-1 at 15 dBm" and "Link-2 when Link-1 at 15dBm" show that when Link-2 is also sending at high power level ($>$ 12 dBm), both the links share the channel indicating Scenario (a).

Figure 3.6(b) shows similar results for "Link-1 at 9dBm". However, the figure also shows a different behavior for "Link-1 at 12dBm". This behavior is explained by the occurrence of Scenario (b). When Link-2's power is low, the network is in Scenario (e). When Link-2 is between 3 and 9 dBm, the network is in Scenario (d); both links transmit simultaneously and Link-1's delivery ratio decreases due to collisions. On further increasing Link-2's throughput, the network reaches Scenario (b), and hence the delivery ratio of Link-1 increases. However, since Link-1 accesses the channel less frequently, the total packets sent on Link-1 itself decreases thereby leading to unfairness. While Scenario (c) is less likely to occur than the others, it clearly leads to loss in throughput similar to (d).

In summary, while the goal of dynamic power management is to make any pair of AP→client links appear like scenario (e) to maximize independent transmissions, a power management scheme should also minimize the occurrence of other undesirable scenarios (b), (c) and (d).

3.3 Design of Contour

Our goal is to improve spatial reuse in a managed WLAN deployment through dynamic power management while addressing the challenges posed by power management. Given a set of power levels P_j,$j \in [1...n]$, that each AP supports, the design of Contour is governed by our observation that an AP can transmit to each of its clients C_i at the lowest power level P_j that minimizes interference to transmissions of other APs, while not affecting the performance perceived by the client C_i. In what follows, we first define the principles on which Contour is designed, provide an overview of Contour's architecture and functionality, and then present the design of its individual components.

3.3.1 Principles

Given a set of APs, each with a set of associated clients, we define two novel principles to address link asymmetry and hidden node problems introduced by dynamic power control.

1. At any instant of time, each AP transmits at the same power level P_j, $j \in [1...n]$.

   This principle, in particular, ensures symmetry such that either two APs hear each other or neither of them do so.

2. Two AP $\rightarrow$ Client transmissions may not be scheduled simultaneously if one of them affects the delivery ratio of the other.

   This principle may be easily realized in conjunction with the first principle by upgrading the power level of one of the AP to client links.

Realization of the second principle would address the hidden node problem. Note that the hidden node problem can also be addressed using RTS/CTS. However, researchers have argued against the use of RTS/CTS, in general, since it incurs significant overhead and degrades the throughput [121]. We mention in passing that our technique does not preclude the use of RTS/CTS; it can be used in conjunction with ours if it can be used selectively selectively so as not to incur significant overhead.

3.3.2 Overview

In designing Contour, we make the following assumptions.

Deployment Assumptions: First, each AP in the deployment that uses Contour is connected to a central controller either directly through a wired or a wireless medium, or indirectly through other APs. Contour does not require any other connectivity between APs. Second, bulk of the traffic flows in the AP to client direction since users of WLANs often are downloaders of data.

Hardware and MAC Assumptions: First, Contour can not modify the MAC layer for compatibility with commodity hardware. Second, APs can send packets in both unicast and broadcast modes. Finally, each AP allows several power levels at which packets can be transmitted; the granularity of power levels is similar on all APs and is known to the central controller.

Contour **Functionality:** We now outline the steps that realize the two principles discussed in Section 3.3.1 to perform dynamic power management:

- Each AP is synchronized to a global real-time clock as defined by the central controller. We discuss in Section 3.5, a simple and light-weight technique

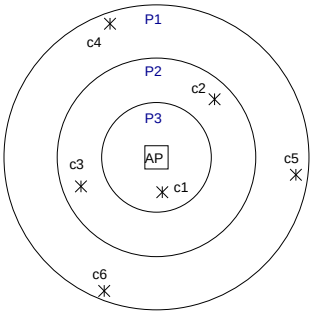


Figure 3.7: AP associates a minimum power level to each client at which the delivery ratio is greater than a threshold: c1 $\rightarrow P_3$; c2,c3 $\rightarrow P_2$; c3,c4,c5 $\rightarrow P_1$.

using a commodity GPS receiver that achieves synchronization between APs within 200μs, and scales to any number of APs.

- An AP associates each of its clients to a minimum power level (in mW) at which the delivery ratio is greater than a threshold (see Figure 3.7).

- The controller determines an envelope of power levels (in mW) and the time τ_k to spend in each power level P_k and directs the APs to start following the envelope at a specific real-time. The envelope is a sequence of tuples of the form $[(P_1, \tau_1), (P_2, \tau_2), ..., (P_n, \tau_n)]$, where $\sum_{k=1}^{n} \tau_k = \mathcal{T}$. $\mathcal{T}$ represents the period of the envelope. The times τ_k are initially set to a default allocation of $\mathcal{T}/n$.

  Each AP repeats the envelope starting at the specified real-time, as shown in Figure 3.8. Since APs are synchronized to the same global real-time clock, following the envelope ensures that all APs transmit at the same power at any instant of time till a new envelope is directed by the controller.

- Each AP determines refinements to the envelope to better adjust to its clients' requirements (including their position, traffic, etc). The AP sends refinement hints to the controller. The controller derives a new global envelope based on all the suggested hints and directs the APs to start following the new envelope starting at a given real-time in the future.

- If the AP detects that a particular client is observing a low delivery ratio, it upgrades the client to the next power level.

Observe that the behavior above decouples synchronization of APs from communication of new envelopes between the controller and APs, thereby ensuring that the communication delay has minimum impact on the system behavior.

The design of Contour ensures that the following two claims hold true.

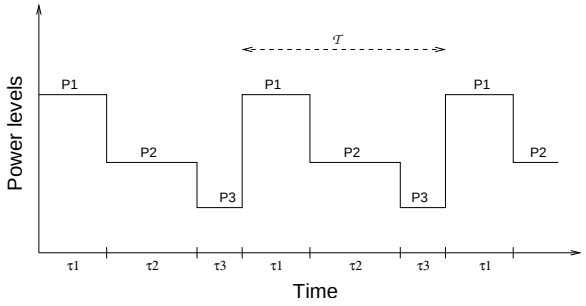


Figure 3.8: Each AP follows the envelope in cycles. All APs transmit at the same power level at any instant of time.

Claim 1: *No asymmetric links due to power management are introduced by* Contour.

Claim 2: *In a network of APs using* Contour*, if a client C_i experiences the hidden node problem in steady state, the client C_i experiences the problem even in the network of APs that do not use* Contour *and all nodes transmit at their maximum power level.Hence,* Contour *only decreases or leaves unaffected the number of hidden nodes in the system, it does not increase the hidden nodes.*

While the first claim directly holds true by principle 1, the second claim follows from the realization of the two principles and the observations in Figure 3.5. The basic idea of the second claim is as follows: If a client that experiences low delivery ratio is at a lower power level than the maximum level, an AP using Contour upgrades the power level of the client. This process repeats as long as the delivery ratio is still lower than a threshold (i.e. the hidden node problem still exists) or till the client reaches the maximum power level. If the delivery ratio is low even when the client is at the maximum power level, then the network without Contour would also have the hidden node problem for this client.

3.3.3 Packet Scheduling with Contour

Figure 3.9 shows the scheduling architecture of each AP using Contour for packets on the downlink(AP→Client). Contour employs multiple queues in the downlink direction to enable easy packet scheduling; each queue corresponds to a power level at which the packets can be transmitted to reach the clients successfully. On a packet arrival from the network, an AP identifies the power level of the client for which the packet is destined, and enqueues the packet in the corresponding queue. Packets are dequeued by the pull switch when the output device is ready to send packets.

The envelope provided by the controller defines the transmit power levels to use

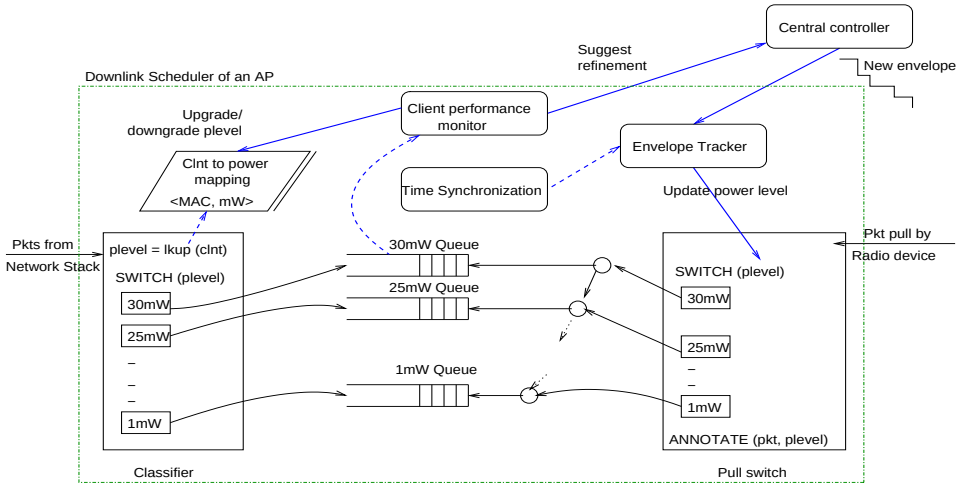


Figure 3.9: Scheduling architecture of an AP using Contourfor downlink traffic.

at each instant of time. Based on the envelope, the envelope tracker runs a timer and sets the *allowed* power level in the pull switch at each transition point. The output device, when ready to send a packet, pulls a packet through the pull switch from the non-empty queue with highest level equal to or below the allowed level. Allowing packets to be pulled from not just the queue with the allowed level, but also from all the queues *below* the allowed level ensures better utilization. Note, however, that packets from queues at lower levels are still transmitted at the current allowed level to avoid link asymmetry.

3.3.4 Envelope Refinement

The performance monitor maintains an exponentially weighted moving average of arrival rate, λ_i, for each queue. Based on this rate, the monitor periodically generates refinement hints such that the time spent at each level is proportional to the arrival rate at that level, i.e., the refinement hints are generated as a sequence of tuples

$$\left[\left(P_1, \frac{\mathcal{T}}{n}.\frac{\lambda_1}{\rho_1}\right), \left(P_2, \frac{\mathcal{T}}{n}.\frac{\lambda_2}{\rho_2}\right), ..., \left(P_n, \frac{\mathcal{T}}{n}.\frac{\lambda_n}{\rho_n}\right)\right]$$

where ρ_i represents the maximum transmission rate at power level P_i and $\mathcal{T}$ represents the period of the envelope. The monitor sends the refinement hints to the controller.

In refining the envelope, the controller ensures fair sharing of the channel at each power level across all APs. To achieve this, the controller constructs a new global envelope in two steps. First, each level is assigned a minimum time allocation to

avoid starvation. The remaining time out of the period $\mathcal{T}$ is then distributed among the power levels that require more than the minimum allocation. In particular, let t_k^j denote for brevity the time $\frac{\mathcal{T}}{n}.\frac{\lambda_k}{\rho_k}$ at power level P_k requested by AP j in the refinement hints.

1. The controller first calculates the minimum time spent by all APs at each power level as

$$\tau_k = \texttt{MIN}\left[\left(\forall j \quad \texttt{MAX}\left(t_k^j\right)\right), \frac{\mathcal{T}}{n}\right]$$

2. Let $\mathcal{S} = \mathcal{T} - \sum_k \tau_k$ represent the total slack out of the period $\mathcal{T}$, where $\sum_k \tau_k$ denotes the total time used up in the first step. Further, let $\mathcal{D}_k = (\forall j \quad \texttt{MAX}(t_k^j)) - \tau_k$ denote the deficit at each power level. We distribute this total slack by taking a greedy approach; we iterate over the power levels with non-zero deficit and fulfill the deficit of the highest power level in each iteration until all the residual slack is allocated or there exist no more power levels with non-zero deficit. The remaining slack, if any, is distributed among all the power levels proportional to their current allocation.

The controller sends the new envelope to all the APs with a new real-time far enough in the future that every AP receives the new envelope before their clocks reach the real-time.

3.3.5 Adapting Client Power Level

When a new client associates with an AP, the AP chooses the minimum power level (marked as the default level) at which the delivery ratio exceeds a threshold. The AP monitors the delivery ratio of each of its clients at regular intervals and upgrades the client to the next power level if the delivery ratio in the interval is below the threshold. The client is downgraded to its default level every few seconds to ensure that clients will not get permanently upgraded to higher power levels due to transient problems in delivery ratio. Adapting the client power level as above addresses two problems—(1) it addresses the hidden node problem, and (2) it makes the network robust to fluctuating channel conditions. In our prototype, the delivery ratio threshold is set to 90%, the monitoring interval is set to 1 second, and the client is downgraded to its default level 30seconds after it was last upgraded. However, each of the parameters is configurable in our prototype.

To estimate the delivery ratio of each client without assistance from the client, each AP gathers statistics of number of retransmissions, number of packets dropped due to excessive retransmissions, and the total number of packets transmitted to each client. Delivery ratio is then calculated as the fraction of the number of successful transmissions to the total number of transmissions.

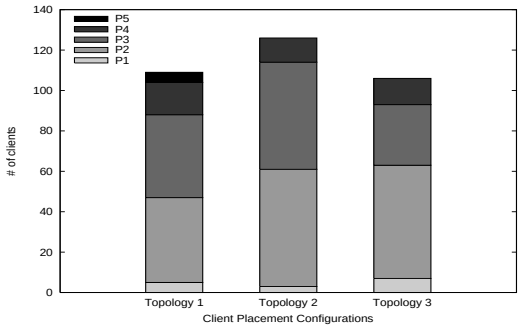
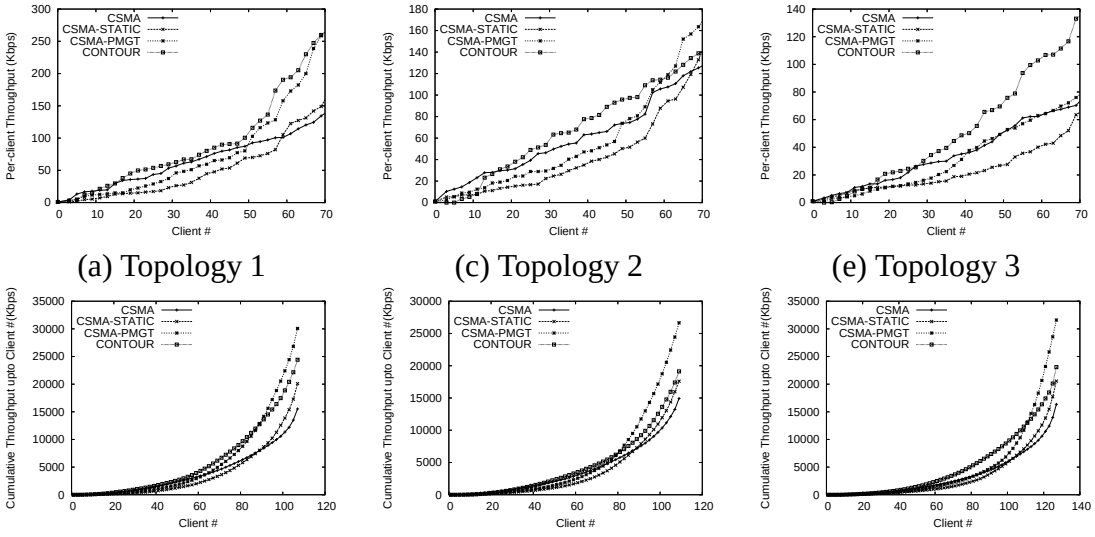


Figure 3.10: Client power level distribution.



(a) Topology 1 (c) Topology 2 (e) Topology 3

(b) Topology 1 (d) Topology 2 (f) Topology 3

Figure 3.11: Graphs (a),(c) and (e) show the individual client throughput sorted in the increasing order from left to right, and (b),(d),(f) show the aggregate throughput of the lowest X clients.

3.4 NS Simulations

In this section, we evaluate Contour through packet-level simulations in ns2.29 over several topologies generated using subsections of the Google Wifi network [4]. Each topology contains APs from a 1x1 kilometer subsection of the Google network, and clients generated at random locations in the same subsection. Each AP has between one and ten clients, with five clients on an average. For brevity, we show results using three topologies; similar results were obtained with other topologies we used. Each AP is considered to have a maximum transmission range of 250m (as in ns2 by default), and clients that are not within 250m of any AP

are discarded. The APs can transmit at six power levels numbered from 1 to 6 with maximum transmission ranges of 30, 90, 130, 170, 210 and 250 meters respectively. Our ns2 modifications for power control are based on Kawadia's ns2.26 code [54]. At each level, the carrier sense and interference ranges are double the transmission range. Each client associates with the AP closest in distance. Figure 3.10 shows the number of clients associated at each power level in each topology, and the topologies have 24 APs.

3.4.1 Algorithms and parameter settings

To demonstrate the efficacy of Contour, we compare its performance with three algorithms.

1. CSMA: 802.11's base-line technique of using CSMA/CA with no power management. APs and clients use power level 6 for transmission.

2. CSMA-STATIC: a static power management technique in which each AP transmits at the power level required to reach its *farthest* client. Clients transmit at the power level used by the associated AP.

3. CSMA-PMGT: per-client dynamic power management technique in which each sender chooses the optimum power for each packet to reach the receiver.

For simulations with Contour, we set the period $\mathcal{T}$ of the schedule to 60ms, and the envelope is refined every 6 seconds. The MAC data rate is set to 11Mbps for all the experiments. Since our measurements with a real system showed no overhead of dynamic transmit power control (TPC), we do not account for any overhead of TPC in our simulations. RTS/CTS is turned off, and we use two-ray path loss radio propagation model. MAC-level synchronous acks for each packet (from both APs and clients) are transmitted at the same power level at which the packet was transmitted.

3.4.2 Results

For the algorithms above, we study the aggregate network throughput and per-client fairness. Through these simulations, we demonstrate that Contour provides (1) increased network throughput compared to CSMA, and (2) provides better fairness compared to CSMA-STATIC and CSMA-PMGT, and even greater network throughput in some cases.

Fairness: We first demonstrate that Contour has better fairness than CSMA-STATIC and CSMA-PMGT. To do so, in Figure 3.11(a), we setup backlogged downstream (from APs to clients) CBR traffic in one topology, and plot the individual throughputs of clients using all the four algorithms. The throughputs are

sorted in the increasing order. The graph in Figure 3.11(a) shows that both CSMA-PMGT and CSMA-STATIC have 50 and 60 clients respectively out of 106 whose throughputs are lower than with CSMA. This behavior is because of link asymmetry introduced by the two algorithms. Whereas, Contour has only about 8 clients whose throughputs are below that of CSMA; these clients have lower throughput because of using a global schedule across all APs. This graph demonstrates that Contour makes the throughput of most of the clients better unlike CSMA-PMGT and CSMA-STATIC, thereby exhibiting better fairness properties.

Network throughput: Figure 3.11(b) shows the aggregated throughput obtained from the same data in Figure 3.11(a); for instance, the aggregated throughput at x-value 60 represents the sum of the lowest 60 client throughputs. We make two observations. First, Contour gets higher network aggregate throughput than CSMA, as do the other two power management algorithms. Second, the shapes of CSMA-PMGT and CSMA-STATIC curves demonstrate that the throughputs of a few clients dominate and lead to a higher aggregate improvement over CSMA, while most clients have lower throughput than CSMA. We observe similar results in Figures 3.11(c),(d),(e) and (f) for the other two topologies.

Effect of upstream traffic: One key concern with Contour being applied only at the APs is that the upstream traffic from clients can introduce asymmetry, thereby reducing the benefits of Contour. To study the effect, we setup various amounts of upstream traffic and compare the aggregate network throughput with both CSMA and Contour. Figure 3.12 varies the amount of upstream traffic generated by each client. The graph shows that Contour improves the downstream aggregate throughput over CSMA even at high upstream traffic. The total throughput (upstream and downstream), however, decreases with increasing upstream traffic. Note that for upstream traffic, clients with Contour use the power level at which they are associated.

Contour**'s effect on delay:** Finally, we study the effect of slotted scheduling in Contour on packet delays in air. Any increased delay can have adverse impact on applications. For instance, if the WLAN hosts VOIP applications, the quality of voice calls can degrade because of increased delays induced by Contour. Similarly, the round-trip time (RTT) for TCP sessions can increase, thereby reducing the throughput of applications (Recall that the throughput of a TCP connection is inversely proportional to its RTT).

Hence, we perform two experiments—one with voice-like CBR sessions, and one with TCP sessions. The voice-like sessions are two-way and setup between each client and its associated AP; each session generates 50 packets per second in either direction (client to AP and vice versa), each packet is of 100 bytes. We plot in Figure 3.13(a) and (b) the CDF of the delay between the time when the packet was sent by the sender-side application, and the time when the receiver application

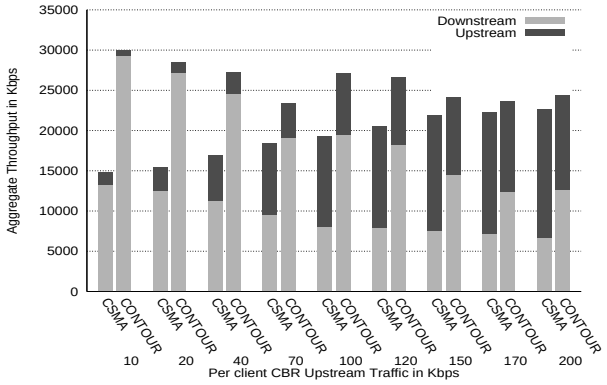
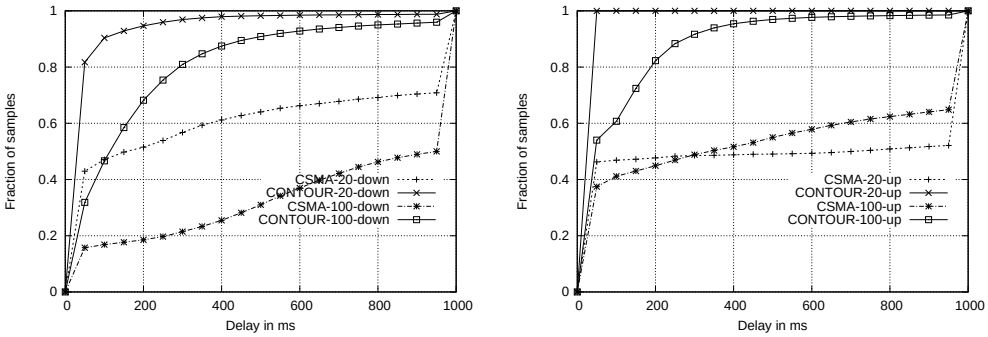


Figure 3.12: Impact of upstream traffic.



(a) Downstream (b) Upstream

Figure 3.13: Delay profile for voice-like traffic.

received it. The total number of packets successfully sent are normalized to those sent by Contour. We show the results for two cases: with voice sessions setup at 20 randomly selected clients, and with sessions at 100 clients. In both cases, for a given delay budget, Contour is actually able to send greater number of packets than CSMA. Contour also causes minimal impact on TCP traffic; for brevity, we do not show the graph here.

Summary: Although Contour's slotted scheduling causes certain APs to idle during certain periods, the results show that the technique globally provides better fairness and increased throughput.

3.5 Prototype

In this section, we briefly describe our prototype implementation.

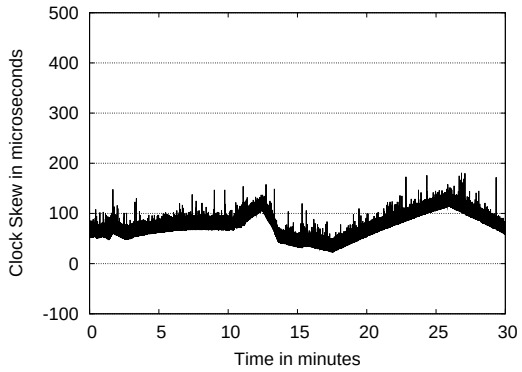


Figure 3.14: Setup for synchronization with GPS.

3.5.1 Contour Implementation

We realize the architecture shown in Figure 3.9 using the Click Modular Router [57]. We implement a new element *SlottedPull*, which is a modified *Pull-Switch* element supported by Click. The SlottedPull element implements the packet scheduling logic and the envelope tracker. The envelope tracker uses the *gettime-ofday* function for accessing the real-time and runs a timer to update the power levels. We patched the Linux kernel 2.6.16.13 with the high resolution timer subsystem [5] patch, which provides microsecond accurate timers with minimal cpu overhead. The central controller is implemented as a module running on a separate machine that communicates with the SlottedPull element using the Click control socket. APs are connected to the central controller via a wired ethernet connection.

Each AP in our prototype setup is a Dell laptop running Linux 2.6.16.13, and uses an Atheros [2] chipset AR5212 802.11a/b/g PCMCIA card; we use the Linux MadWifi [64] device driver, version 0.9.2. This card provides two methods to control transmit power. The first method is via a global setting, which determines the power at which all outgoing packets are transmitted. The second method is by specifying the power level within the *transmit descriptor* of each outgoing packet. Since Contour requires the second method, we enable the second method by recompiling the MadWifi [64] driver with Transmit Power Control (TPC) feature enabled. The second method provides fine grained control of transmit power on a per-packet basis with no additional overhead over the first method. The card supports specifying 60 power levels at a granularity of 0.5dBm; thus a value of 30 corresponds to a transmit power of 15dBm. The card we used, however, had a maximum transmit power limit of 30mW, equivalent to 15dBm.

3.5.2 Synchronization with GPS

Each AP in the network using Contour is synchronized closely to a real-time clock to enable all APs to transmit at the same power at any instant of time. While there are several techniques to achieve close synchronization [110], we use a simple and light-weight technique using a GPS receiver at each AP. The technique is derived out of White's [12] proposal, but we extended it to achieve quicker synchronization. The technique uses the clock pulses provided per second by a Garmin LVC-18 GPS receiver [42] through one of the six bare wires it supports. These pulses are used as a timing source by the NTP daemon. The raising edge of this pulse is synchronized to within 1μs of the second boundary of UTC clock time. Another bare wire of the GPS receiver carries NMEA [6] complaint GPS data, which is used for obtaining the absolute value of UTC clock times.

A separate user-level program *shmpps* [9] reads the PPS input from the serial port and feeds it to NTP daemon via shared memory. Figure 3.14 shows our setup for synchronizing each AP to the GPS clock. To enable quicker synchronization, a *UTCInit* module initializes the clock using $settimeofday$ to the UTC timestamp at system boot-up time before the NTP daemon starts; it does so by interpreting NMEA data to fetch the UTC timestamp and set the clock at the exact PPS edge. By initializing the timer before starting the NTP, the time to synchronize (within 200μs) is always less than 20 minutes; without the initialization; such synchronization could take few hours depending on the initial clock skew between the current system clock read by the NTP and the GPS clock. After initializing the clock, the NTP daemon starts and disciplines the clock continuously using the pulse received every second from the *shmpps*.

Since every AP synchronizes to the UTC clock obtained from the GPS receiver, this synchronization technique scales to arbitrary number of APs.

3.6 Prototype Evaluation

In this section, we evaluate different aspects of Contour using our prototype implementation. In what follows, we describe the experimental setup, microbenchmarks and an experiment showing the overall benefits of Contour in a setup with two APs and eight clients in a small-scale realistic deployment.

3.6.1 Setup

Figure 3.15 shows our experimental setup. Our setup contains ten nodes—two 802.11b APs, each with four associated clients. The clients are placed randomly in different office cubicles to emulate a realistic scenario. The APs are marked as AP# and the clients are marked as C#. The APs and clients are Dell laptops with the Atheros PCMCIA cards. All nodes operate in the 802.11b mode. The central

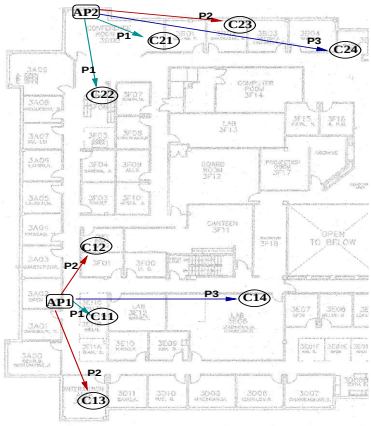


Figure 3.15: Experimental setup.

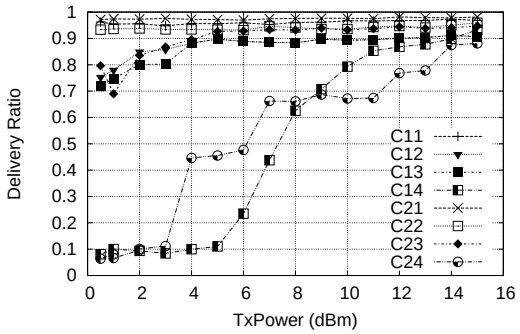


Figure 3.16: Determining the minimum power level for each client.

controller runs on another Dell laptop that is connected to all the APs through a wired ethernet network. Each AP and the controller are connected to a GPS receiver of their own. The APs are located near the windows to obtain a good GPS signal[2].

Figure 3.15 also shows the minimum transmit power for each client required by an AP to ensure a delivery ratio of 90%. To determine the minimum transmit power for each client, each AP sends a series of UDP packets at different power levels to each client and measures the delivery ratio. Note that this same technique can be used in a real system when a new client associates with an AP to determine the transmit power level for the client. Figure 3.16 shows the graph of delivery ratio with changing transmit power (in dBm) for different clients. Based on the measured delivery ratios, each AP chooses the minimum power level at which the delivery ratio exceeds 90%. In this case, C11, C21, C22 are placed at a power

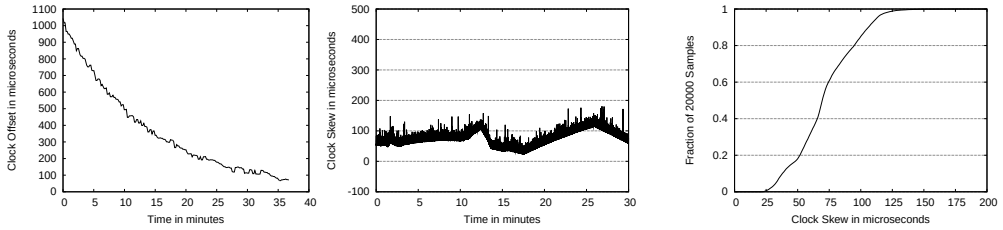
[2]Note that the GPS signal may not be strong enough for synchronization in deep indoor deployments of APs, and an alternate synchronization method would be required. However, Contour is independent of the specifi c synchronization mechanism used as long as the synchronization is accurate within an acceptable range.

level of P1 (0dBm), C12, C13, C23 are at P2 (=6dBm), and C14, C24 are at P3 (=14dBm).

Our experiments are divided into two parts. First, we present microbenchmarks to evaluate the efficacy of different components of our prototype. Second, we present the overall benefits of Contour in a realistic deployment shown in Figure 3.15.

3.6.2 Microbenchmarks

3.6.2.1 Time Synchronization

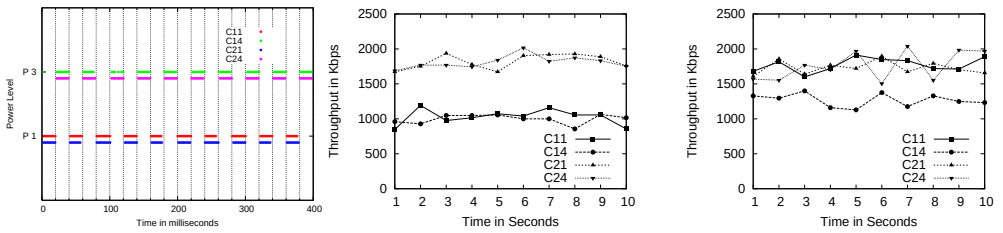


(a) Cold start to synchronized state latency

(b) Clock skew between two APs in steady state

(c) CDF of clock skew

Figure 3.17: Time Synchronization.



(a) Packet transmissions following Envelope with two power levels

(b) AP1 and AP2 always transmit at power level P3

(c) AP1 and AP2 use the envelope to switch between power levels P1 and P3

Figure 3.18: Packet scheduling with Contour.

Our goal in this experiment is to answer the following two questions about our time synchronization technique–(1) How quickly can an AP synchronize with the GPS clock, and (2) How closely can two APs synchronize using the GPS clock? To address the first question, we plot the clock offset observed by the NTP daemon that represents the difference between the local clock and the UTC timestamp indicated by the GPS pulse. Figure 3.17(a) shows that the AP gets synchronized to within

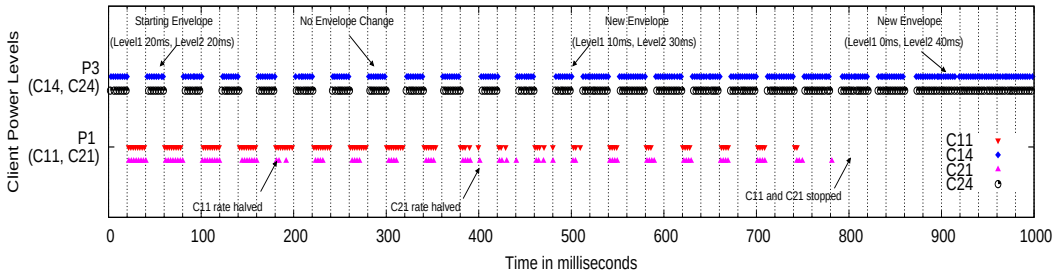


Figure 3.19: Envelope refinement.

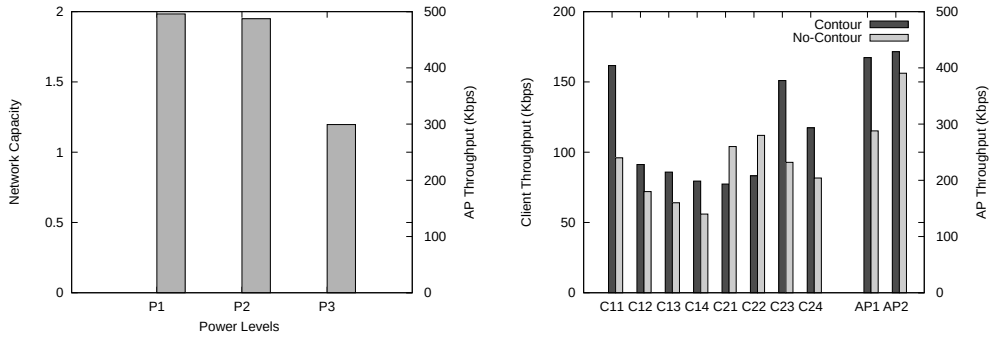


Figure 3.20: Performance of Contour in a deployment of two APs, each with four associated clients.

200 μs in 20 minutes after the *UTCInit* module sets the initial time and the NTP daemon starts.

For addressing the second question, we measure the clock skew between two APs with their clocks synchronized using the GPS receiver. We establish a CBR UDP flow at a rate of 10 packets/sec between the two APs over a wired ethernet. The sender AP timestamps every packet before transmission over the wired network. The receiver AP first calculates the difference between the transmit timestamp in the packet and receive time (according to the local clock) for each packet. From this value it subtracts half the average ping delay between the two APs to obtain the clock skew. The ping delay does not vary much between samples because of the wired network connection. Figure 3.17(b) plots the clock skew between to APs over a 30 minute experiment, and Figure 3.17(c) is the corresponding cdf plot. The graphs show that the clock skew is within 200μs, and the skew is within 100μs 80% of the time.

3.6.2.2 Scheduling with Contour

The goal of this experiment is to demonstrate that two APs can schedule packets for different clients at different times as indicated by the envelope. We choose

four clients C11, C14, C21 and C24 for this experiment. C11 and C21 are at the lower power level P1, and C14 and C24 are at the higher power level P3. We setup four long-running UDP flows from AP to each of the clients. Each AP uses the following envelope of power levels: $[(P1, 20ms), (P2, 0ms), (P3, 20ms)]$. We plot in Figure 3.18(a) a point for each packet transmitted to the client using the transmit timestamps of the packet on x-axis. The graph shows that the timestamps of C11, C21 are together, and C14, C24 are together and different from C11, C21. The graph indicates that the two APs are closely following the envelope.

To demonstrate the benefits of following the envelope in this experiment, we perform two experiments. We first set both APs to always operate at the highest power level P3. At P3, the APs can sense each other's transmissions and hence share the channel. At P1, neither of them can sense the other, and hence can transmit simultaneously. Figure 3.18(b) plots the throughput obtained by each of the clients. In this case, since both APs share the channel, all clients should receive the same throughput. We observe, however, that the clients of AP1 get lower throughput compared to AP2 because AP1 is transmitting packets at a lower rate compared to AP2. Even though both APs use the same power level, the asymmetry is arising due to difference in carrier sensing at AP1 and AP2. Hence, AP1 perceives the channel as busy due to AP2's transmissions more often.

We now repeat the experiment with Contour that uses the above envelope. Figure 3.18(c) depicts the throughput obtained with Contour for each of the clients. he graph shows that the throughput of C11 and C21 improves substantially because APs transmit simultaneously to C11 and C21. C14 and C24, on the other hand, receive similar throughput as in Figure 3.18(b) since the APs share the channel at power level P3.

3.6.2.3 Envelope Refinement

This section evaluates the envelope refinement part of Contour. We use the same topology of two APs and four clients as in the previous benchmark. We start with long-running flows for each client. The two power levels P1 and P3 initially have equal slot widths of 20ms. We initiate a series of changes to the flow rates during the course of the experiment and evaluate how the refinement algorithm in Contour is able to adapt to the change in packet arrival rate. Figure 3.19 shows the changes and adaptation in the envelope (The plot is derived similar to Figure 3.18(a) by using the packet timestamps). After the 200th millisecond, we halve the packet arrival rate for C11 from 400 pkts-per-second to 200 pkts-per-second. As a result AP1 suggests a refinement of 10ms at P1 and 30ms at P3 to the central controller after monitoring arrival rate in the queue corresponding to P1 for 80ms (two periods). AP2 on the other hand does not observe any change in the arrival rate and hence does not suggest any changes in the envelope. Thus the envelope remains unchanged. Next, flow C21 is halved to 200 pkts-per-second. As a result, both APs

independently request for an envelope of 10ms at P1. The controller evaluates the new envelope using both the requests and initiates a change in the envelope at 500th millisecond. Next we stop flows for both C11 and C21 after 800th millisecond. This results in a new envelope that only uses P3 for all slots.

3.6.3 Overall Benefits

This experiment uses all the clients shown in Figure 3.15 to evaluate the throughput improvement obtained with Contour. In this experiment, we measure the throughput achieved by each client when run together with long-running UDP flows from the APs to the clients. In Figure 3.20(a), we first plot the network capacity, which is defined as the cumulative throughput obtained when both APs simultaneously transmit. The graph shows that the capacity is high at lower power levels because of greater number of simultaneous transmissions. At power level P3, both APs share the channel. Figure 3.20(b) shows the throughput achieved by each of the eight clients with and without Contour. The graph clearly demonstrates that cumulative throughput is increased with Contour. However, some clients receive lower throughput with Contour than the case without Contour. This is because, Contourcurrently provides fairness at a per-power-level granularity, and not on a per-client basis. We will explore the problem of providing fairness on a per-client basis as a part of future work.

3.7 Open Issues and Limitations

Although our prototype evaluation demonstrates the benefits of per-client power management, we observe that Contour is only a first step towards reaping such benefits. Contour can be enhanced in at least three fronts.

- In this work, we fix the transmission bit rate of each AP and study the benefits of power control. However, bit rate and transmit power are related; the lower the bit rate required to support a client's throughput requirement, the lower can be the power. Hence, Contour can further reduce the AP's power level for a client based on its throughput requirement, thereby increasing the benefits of power control.

- The envelope-based technique currently provides fair sharing of resources across clients of the same power level. This can have a bias against power levels that have greater number of clients. As a result, some clients may receive lower throughput with Contour compared to a system without power management. Defining the right notion of client-level fairness and ensuring it in such a framework is an interesting topic that we plan to address in our future work.

- In the interest of avoiding link asymmetry introduced by power management, Contour makes each AP transmit at the same power level at any instant of time. Such a technique may be inefficient as the network of interfering APs grows in size. A more efficient technique would be to divide the network into smaller clusters of interfering APs, where each cluster includes APs with maximum interfering regions, and the number of interfering regions across the clusters is relatively few. These clusters can then use different envelopes that can be independently scheduled.

One limitation of Contouris that its benefits are maximum when all APs conform to the envelope. If some APs in the vicinity do not conform to power control and adversely affect the throughput of APs using Contour, the APs will eventually adapt their clients to be serviced at higher power levels, thereby reducing the benefits of power management. Nevertheless, we believe that Contour will reap the benefits of power management where there are no non-conforming APs or when the non-conforming APs do not contend for resources.

3.8 Related Work

In wireless networks transmit power control research has focused on two issues – (i) power saving and (ii) spatial reuse. Transmit power control is critical for energy savings for any mobile system running on battery. A significant portion of research here has focused on topology control in ad hoc and sensor networks [92]. The idea here is to create useful network topologies that use only the minimum amount power in the network. A bunch of other protocols mainly consider sleep and wakeup scheduling for conserving energy [31, 124, 108] or power-aware routing [46, 114] without directly using transmit power control. The 802.11 protocol standard also considers energy savings by essentially scheduling sleep and wakeup periods.

We will, however, focus more on research that performs transmit power control for better spatial reuse and immunity from interference. In [76, 27] the authors studied topology control aspects of transmit power control for interference reduction in a multihop network.

Kawadia and Kumar developed the COMPOW protocol [79] so that all nodes have the same minimum transmit power level for the network to be connected. Their clusterPOW [54, 53] protocol addresses a problem with COMPOW when nodes are clustered (so low transmit power is possible within cluster), but clusters are far apart, thus high transmit power is needed to connect clusters. Their MINPOW protocol [53], on the other hand, finds a globally optimal solution. These protocols are essentially topology control solutions. The same authors also developed the LOAD-POW protocol [53] where a cross-layer approach is proposed to influence routing decisions. Here, the next hop node is not fixed. The next hop chosen based on the

power level that can be safely used without interfering with ongoing communications. Bejerano and Han developed cell breathing technique where power control is used to implement association control in wireless LANs [24].

Transmit power control work has been extensively studied in the context of MAC protocols. Several papers developed new MAC protocols – often variations of 802.11 – for multihop networks targeting better spatial reuse [75, 52, 77, 78, 60]. Inherent in these works is a concept of *interference margin* which is the amount of additional interference a receiver can tolerate without collision. Once potential senders know such a margin of the receivers in the neighborhood, they can adjust their transmit power to eliminate possibility of any collisions. However, new MAC protocols are needed as the margin must be advertised on a per-packet basis. In several other works, power control is combined with rate control for better system throughput [125, 55, 59, 105]. This is due to the observation that with the same transmit power, i.e., same SINR, the packet capture probability increases with lower bit rate. Our work is limited so far to the same bit rate. But the same framework can be considered to do power and rate control jointly.

On the practical side, the authors in [17] studied power control in 802.11 devices, but concluded that power control is infeasible with the current hardware and driver limitations. Our observations in this work, however, has been very different. The underlying scheduling framework that we propose for managing power has close resemblance to work proposed in [93, 58, 96]. They have shown that a packet scheduling framework over existing commodity 802.11 hardware is practical. Our work focuses on using such a framework for power management.

3.9 Conclusion

We have presented Contour, a novel dynamic power management technique for improving spatial reuse in managed wireless LAN (WLAN) deployments. WLAN deployments are often characterized by dense placement of access points to provide maximal coverage to clients. Such density, however, leads to interference between different simultaneous AP to client transmissions, thereby leading to low overall throughput. To mitigate interference, Contour enables each AP to choose the lowest transmit power for each client sufficient for sending packets successfully. Further, Contour addresses two main problems introduced by dynamic power management—creation of asymmetric links and hidden nodes. It solves the asymmetric link problem by ensuring that all APs stay in the same power level at the same time. To solve the hidden node problem, it dynamically adapts the power level for each AP so that APs colliding at clients can hear each other.

Through a prototype implementation, we demonstrate that Contour improves spatial reuse significantly. The implementation uses GPS-synchronized clocks on the APs, computes a global schedule (envelope) that describes how long APs should

stay on what power levels. Appropriate clients are scheduled on the APs at appropriate power levels depending on the computed envelope. Overall throughput improvement in a small network with 2 APs and 8 clients is about 40% in our experiments. A more significant improvement is expected in a larger network.

By implementation and evaluation of Contour, we demonstrate the feasibility of dynamic power control on a slotted framework synchronized across all access points. We believe this feasibility study has wider applicability beyond power control in other wireless domains that require a synchronous slotted framework.

Through simulations and prototype evaluation, we characterize the benefits of Contour in terms of spatial reuse and fairness. The results makes a first case for fine-grained power control in current WLANs for improving capacity.

Chapter 4

Channel Hopping Protocol for Jammer Resilience in WLANs

4.1 Introduction

802.11a,b, and g protocols were designed under the assumption that all nodes are interested in transmission of data, and follow the rules of the protocol regarding when to send and when to permit other nodes to send. These protocols were not designed to account for jammers who would seek to interrupt transmissions. As these networks become more prevalent and society becomes more dependent on their use, the potential for their jamming will increase, whether one is simply out for kicks, or for advantage in a military or business setting.

This chapter explores how to utilize channel hopping to make 802.11 networks resilient to jamming attacks. While Frequency Hopping Spread Spectrum (FHSS) was available in the original 802.11 standard [118], it was not made available in the subsequent, more popular 802.11a,b and g that are our focus.

In the wireless domain, a jammer with unlimited resources can always successfully jam any wireless transmission by flooding the entire spectrum that could be used by the client. However, it is not unreasonable to assume that the jammer may be restricted to a configuration similar to that of a legitimate client. For instance, in many situations, such as a in a secure group meeting or in an open field, the equipment used to jam would be visible to the legitimate participants. To remain remain inconspicuous, the jammer would need to jam with conventional (802.11) hardware such as a single laptop with one or two wireless interfaces. The software can be modified, but under the hardware constraints, the jammer has limited ability to flood the spectrum, which makes mounting a defense possible.

Our results showing what can be done utilize a combination of mathematical analysis and experimentation on an actual 802.11a testbed. We use conventional 802.11 hardware and modify the software to implement a jammer, whose goal is to disrupt communication between an access point (AP) and a receiver connected to

the AP. The jammer continually scans for channels that are utilized, and upon finding one, floods the channel with noise for some time before repeating the process. The jammer ignores 802.11 MAC specifications on backing off, effectively shutting down communication on the channel.

To protect against this type of jammer, *we develop a channel-hopping protocol that can be used by receivers to "hide" their transmissions from the jammer.* The AP and receiver utilize a shared, secure pseudo-random sequence of channels which are used sequentially by the pair to communicate. In the absence of a jammer, there is a short outage period as communication shifts channels each move, slightly reducing throughput.

We also develop a "smart" jammer, who is aware that the transmission it seeks to jam is using this channel-hopping procedure. However, because the pseudo-random sequence is unknown to the jammer, each time the legitimate transmission moves channels, the jammer must again scan for the channel that is utilized by the transmission.

The time between channel switches must be strategically chosen by the entities that wish to communicate: if the times are too small, then the high frequency of outages will reduce throughput. If they too large, then the jammer will find and successfully jam large fractions of the transmission.

Our experimental results demonstrate that, when using conventional off-the-shelf 802.11a cards to implement both the legitimate transmission and an optimally configured channel-scanning jammer, the legitimate transmission can achieve 60% of the throughput rate that is achievable under normal (no jammer, no channel hopping) use. In contrast, without hopping, the jammed transmission's throughput is only 10% of its normal (unjammed) rate. In addition, we show that if our hopping method, configured to protect against an ideal jammer, is applied in the absence of a jammer, the throughput is 90% of what is achievable without hopping.

Our mathematical analysis supports the experimental findings, and also explores several "what-if" scenarios. In particular we look at how effective channel hopping is as the number of channels is increased. We also look at how resilience to jamming can be improved if multiple channels can be used simultaneously by the legitimate players (as well as by the jammer).

The chapter proceeds as follows. In Section 4.2, we give a more specific formulation of the jamming situation we wish to protect against. Section 4.3 explores the challenges of building a hopping protocol, even in the absence of a jammer. Section 4.4 explores how to optimize the hopping in the presence of an intelligent jammer. Section 4.6 explores how to design and analyze future settings with multiple clients and/or clients with multiple cards. Section 4.7 looks at some additional design considerations, and Section 4.8 concludes the chapter.

4.2 Problem Formulation

The initial, generic problem we consider is that of an AP attempting to communicate with a single *legitimate* client over an 802.11-like wireless network. We refer to this communication as a *legitimate* communication to distinguish it from the transmissions sent by the jammer. In the legitimate communication, data may be flowing from the client to the AP, or from the AP to the client.

We let L represent the *number of channels* that can be utilized for this communication. For instance, in 802.11b, $L = 11$, or if one insists on the channels being orthogonal, then $L = 3$.

In Section 4.6, we will analyze settings in which the AP and perhaps also legitimate client are equipped with (possibly) multiple transmission devices such that they can simultaneously communicate across k of these L channels.

The legitimate communication may change channels over time, i.e., the communication may switch to a different, possibly overlapping set of k of the L channels than were used previously. When such a channel change occurs, we say that the communication *hops* between channels. We refer to the time that the legitimate communication stays fixed to a particular channel as the *residence time*.

Even in the absence of a jammer, when the legitimate communication hops channels, there is an *outage* period whose time we represent by τ. The outage is due to the time it takes to reprogram the card to transmit on a new channel, and also any lack of synchrony between the times at which the AP and client respectively switch channels. During the outage, the throughput drops to zero.

A separate client, called a *jammer* may attempt to block the legitimate communication. As mentioned in the introduction, it is not unreasonable to assume that the jammer's hardware configuration is identical to that of the legitimate client. In particular, the jammer's ability to transmit and/or listen on a channel stops for a brief period of time when it switches channels. Also, we assume that the jammer can simultaneously transmit on only a subset of the L channels available within the spectrum.

4.2.1 Channel Hopping

If the legitimate client and AP were to communicate upon a static set of channels, the jammer could listen, identify the set of channels, and proceed to block these channels indefinitely. Clearly, if the legitimate communication wishes to continue, it must hop to a new set of channels.

One approach, as is taken in [123, 122], is *reactive*: the legitimate communication hops channels only after the legitimate participants have identified that the current channels they are using are being jammed. A second approach, taken by us, is to *proactively* hop channels without attempting to verify the status of the channels being hopped from or hopped to. Let us take a high-level look at the pros and cons

of these two approaches.

- **outage (con for proactive)**: Since each channel switch results in an outage, one would like to minimize the number of hops per unit time. This is done by reactive switching. Proactive switching will likely hop more often than is necessary.

- **jammer detection (pro for proactive)**: by proactively hopping at a high enough rate, there is no need for the legitimate communication to detect the presence of a jammer. Obtaining an accurate estimate of a channel's status in a short period of time is not easy. For example, the DOMINO system [97] as reported requires several seconds to make an accurate determination of a "greedy station". In [61], the accuracy of the method is not evaluated with respect to the time it is allowed to sample network traffic, as the concern is identifying selfish users in a steady-state system.

Our experimental results in Section 4.4 will show that a jammer can switch channels and determine whether the channel is being utilized in only a few milliseconds, i.e., confirming use is much easier than confirming that the channel is maliciously being jammed. Hence, when the number of channels is small, as is the case in conventional 802.11, a jammer can quickly locate the channel being used. Hence, a well-designed jam-resilient scheme will not have a very large residence time in these networks, whether hopping proactively or reactively.

4.2.2 Pseudorandom Channel Hopping Sequence

In order to implement channel hopping in a manner that is secure from the jammer and minimize the throughput loss, we assume that the channel hopping sequence is pseudo-random, and that this sequence known by only the AP and the legitimate client. For instance, the AP can decide the method for generating the pseudorandom sequence and using public key encryption, encrypt the description of the method using the client's public key. Methods for generating and exchanging secure pseudorandom sequences are well known [103] and NIST offers several possible standards that can be used [83]. If such pseudorandom channel hopping sequences are applied, even if the jammer knows the past history of channels used, its ability to determine the next channel in the sequence is no better than a random guess. The overhead of exchanging the pseudo-random sequence at the start of the communication is discussed in Section 4.6.

4.2.3 Coordinating Hopping Times

The changing of channels must be coordinated between the AP and the client of the legitimate transmission. One option is to have the AP send an explicit message

indicating that the residence time for the current channel has expired, and the communication should be switched to the next channel in the sequence. However, if the jammer successfully jams the channel, this explicit change order could be blocked. The other option is to transition at pre-determined points of time. In this latter case, the skew in the AP's and client's clocks can negatively impact throughput. To address this issue, we have the AP send periodic pings to the client, allowing the client to adjust its switching time to the AP's clock.[1] Only if the jammer is successful for extremely long periods of time can the clocks get sufficiently skewed that the pseudo-random sequences become disconnected. Handling such a disconnect is also discussed in Section 4.6.

4.2.4 Design of the jammer

We assume that the jammer is aware of our channel hopping protocol and will do as much as it can, given its limited set of hardware resources, to interrupt the legitimate communication. Given the fact that the jammer cannot guess the hopping sequence, what works best is to quickly scan the entire spectrum of channels looking for the legitimate communication to jam. When a channel is found containing this communication (e.g., a packet with MAC addresses of nodes whose communication the jammer wishes to disrupt), the jammer transmits on the channel, interfering with this other communication.

Our results about the resilience of our hopping solution depend upon our claim that we examine the "best" jammer against such a solution. Our work does make the following assumptions about our jammer:

- We assume (conservatively) that when the jammer transmits, it does in fact block the legitimate communication. Packet capture [112, 56] and hidden terminal [115] phenomena may reduce the effectiveness of a jammer whose geographical position is not ideal for jamming.
- The jammer must periodically stop jamming and listen for a brief period of time on the channel to determine if the legitimate communication has already hopped to another channel.

4.2.5 Other issues

Before delving into the specifics of the development of the channel-hopping process for the legitimate communication, we briefly iterate over some additional challenges that a complete channel-hopping system must address.

[1] A lesser issue is the propagation delay, which is small due to the proximity of communication points in 802.11 networks. Furthermore, the propagation delay experienced by the beacons is the same as that experienced by the data packets.

Connection establishment: The AP and client must connect initially to exchange information such as public keys and the pseudo-random sequence. This exchange must also be done via hopping, lest the jammer simply jam this initial attempt at an exchange. One possible approach is to hop, choosing channels truly at random. We analytically evaluate the throughput of this approach in Section 4.6.
Sequence Drifting: If the jammer is able to disconnect the AP and client for a long enough time, the skew in clocks may cause the client and AP to drift such that they are at different steps within the pseudorandom sequence. A short overlap in communication is sufficient to re-synchronize the pair.
Overlapping Channels: The different channels in the 802.11 spectrums are not all truly orthogonal, such that the jammer can listen on one channel and identify the presence of a transmission on a nearby channel. In addition, the jammer need only jam a channel "close" to the channel utilized by the legitimate communication to reduce, but not halt, the legitimate communication. How aggressive the jammer is at finding the exact channel being utilized is one of the parameters we vary in our design space of the jammer.
Multiple cards: It is non-trivial to implement a client that simultaneously utilizes multiple channels within a single 802.11 band. Even if the channels are deemed orthogonal, the reality appears to be that when two transmitting cards on orthogonal channels are close enough, they still interfere with one another's transmissions. We do not experiment with multiple channels, but instead utilize mathematical analysis in Section 4.6 to explore what could be done if multiple cards could be implemented on a single receiver without interfering with one another.

Our current prototype has the AP and client exchange the pseudorandom sequence offline. Also, our experiments are performed on a testbed where the AP, legitimate client, and jammer each have a single wireless interface and hence can each transmit or listen to a single channel at any given time. We also assume a static legitimate session where the client associates with the same AP for the duration of its communication. We do not consider the issue of handoff between APs, and leave this issue as future work. We present some preliminary speculation in Section 4.7 about how this can be handled.

4.3 Design

In this section, we describe the design of our channel hopping protocol and the jammer. The goal of the channel hopping protocol is to minimize the loss of throughput due to outages across channel hops in the absence of a jammer or due to disrupted communication through successful jamming. The jammer on the other hand, must be agile to react to a channel switch by the client and the AP and it must prevent any useful communication on a channel when jamming it.

4.3.1 Channel Hopping Protocol

Channel switching at the client and the AP must be synchronized to minimize outages. Since the channel hopping must proceed at a high rate, and the communication may be disrupted by the jammer, a synchronization protocol, that waits for all participants to acknowledge the switch before proceeding, is impractical. Independent clock synchronization, across clients and the AP, at the granularity of a few milliseconds is challenging. The synchronization is further hampered by load-dependent overheads of physically modifying the channel used by the hardware.

Assuming perfect synchronization is impossible, the channel hopping protocol must handle the drift across the systems and resynchronize whenever possible. This resynchronization must allow clients whose pseudo-random sequence has deviated from that of other clients and the AP. This allows the system to re-establish the sequence of channel hops as well as timing through minimal overlap of communication channels.

4.3.1.1 Initialization

In our system, the client and the server use the same algorithm to generate the pseudorandom sequence. The initial seed is exchanged offline, or when a client joins the network, it waits on a single randomly selected channel without hopping. When the AP selects this channel for communication, the client associates with the AP and starts hopping.

4.3.1.2 End-of-Slot Message

Channel hopping proceeds without synchronization across clients and the AP. The hopping period is chosen offline and all clients use a timeout at the *local* machine to hop channels. In addition to the timeout, the AP broadcasts a special `End-of-Slot` message to signal the end of its residence in the current channel. This message also contains the next number being chosen by the AP in its pseudo-random sequence. When a client receives this message, it *(i)* immediately switches channels, *(ii)* resets its pseudo-random sequence to reflect that of the AP, and *(iii)* readjusts its timeout to the residence time from the current switch.

Figure 4.1 depicts, via a timeline, our implemented channel hopping mechanism. In this example, the first two channel hops are initiated via the transmission of the `End-of-Slot` message. For the third hop, this message is lost, and hence the timeout triggers the switch at the client. The `End-of-Slot` message is sent as a management packet, using an unused vendor-specific identifier included in the IEEE 802.11 implementations at the highest priority to prevent queueing delays at the network interface.

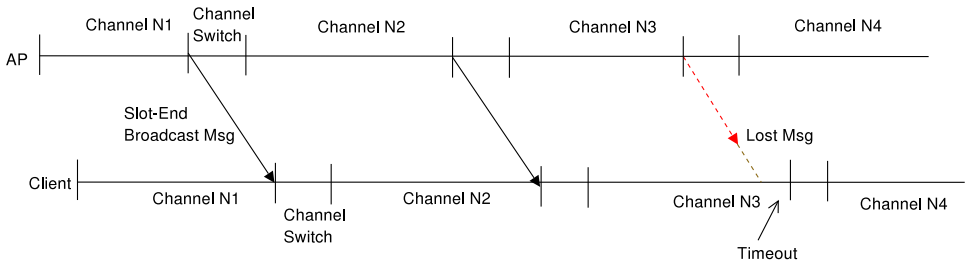


Figure 4.1: Channel Switching Protocol.

4.3.1.3 Channel Switch

Following the default behavior after a channel hop, a client loses its association with the AP and the association process must be started all over again. We disabled this default behavior in the driver so that client retains the association even after a channel hop. Additionally, the AP keeps the state information of the client.

Our current implementation drains all the transmission queues. This leads to loss of packets, not only during the switch, but also of those packets queued for transmission at the network interface. These packets can be rescued by buffering them during the channel switch and reintroducing them at the interface after the switch has completed.

4.3.2 Jammer

Our experimental jammer listens on a channel for an amount of time bounded above by a constant that we call the *Listen Interval* to detect ongoing traffic. If the jammer receives a packet containing the MAC address of a participant in the legitimate connection it seeks to jam, it has found its channel to jam. The jammer immediately starts sending packets as fast as possible for a fixed amount of time we call the *Jam Interval*. When the interval is completed, the jammer returns to its listening state on the same channel, and process repeats. The goal is to occupy the channel utilized by legitimate communications for as much time as possible to create as much interference on the channel as possible.

The 802.11 MAC specification [14] is designed such that nodes utilizing the 802.11 protocol share the channel via CSMA/CA. This ensures that no node can hog the channel, blocking other transmission attempts. Moreover, the MAC protocol is typically closed source and cannot be changed. Nonetheless, there are several other ways to implement a jammer.

4.3.2.1 Our Approach: Jammer Broadcast

A simple scheme that we implement for our experiments has the jammer sending large broadcast packets back-to-back. Broadcast traffic can degrade the performance of ongoing unicast traffic since these packets do not have to follow the back-off scheme used by 802.11 to retransmit unicast packets. In addition, broadcast traffic is sent at the lowest rate, which stretches the time used to send a packet. Furthermore, increasing the packet size beyond the typical MTU(1500bytes) can also improve the jamming efficiency.

4.3.2.2 Other Jamming Approaches

There are other, perhaps more effective ways to implement a jammer. However, at the time of submission, we were unsuccessful at implementing these methods. We describe each briefly:
Disabling Carrier Sensing: by disabling the carrier sensing (CS) a node can transmit a packet even when the channel is already busy. This allows the jammer to corrupt ongoing legitimate transmissions. In addition, the channel can be kept busy by sending back-to-back packets and prevent other nodes from accessing the channel. Madwifi driver exposes certain control registers that permit us to modify the behavior of the card. One such register provides a method to enable or disable carrier sensing. However, at the time of submission, we have not yet been successful at implementing this method.
Modifying slot times: Fixed slot times defined in the protocol (DIFS, CWMIN, CWMAX) are parameters that control the amount of time between different packet transmissions and back-off window times. Decreasing these values below the standard defined values increases the priority of access to the channel of the jammer. Though this scheme does not ensure complete access to the channel, it can increase the channel utilization time for the jammer node while starving the remaining nodes in the channel.

4.4 Analysis

In this section, we begin with an analysis of the optimal residence time. We derive formulas that determine the optimal time as a function of the number of channels and the maximum hopping rate of conventional 802.11 cards (used by both the legitimate client and the jammer).

4.4.1 Optimal Hopping Time

Let τ be the time required to hop, and that no data can be sent while the hop is occurring. If the channel is switched after transmitting on the current channel for

$s * \tau$ time units, without an jammer, the throughput is clearly $s/(s+1)$ times what it would be without hopping.

Suppose legitimate communication utilizes a single channel at a time, and the jammer can listen to and jam a single channel at a time, and suppose the jammer also requires time τ to hop channels. Furthermore, suppose the jammer can determine instantaneously when it has hopped to a channel on which legitimate communication exists and immediately jams this channel. Furthermore, let us assume that the jammer knows precisely when the channel ceases to be used by the legitimate communication (i.e., the client and AP are in the process of hopping). Last, to simplify presentation, we normalize the throughput rate without channel hopping to one. Hence, any throughput rates stated below indicate the fractional rate with respect to no channel hopping.

In this model, the jammer can "check" j channels in time τj. Since there is a total of L channels, the probability that the jammer has hit the right (pseudo-randomly selected) channel after checking j channels is j/L for all $0 \leq j \leq L$.

Hence, if the legitimate communication has a residence time of $s\tau$ on a channel (putting time $(s+1)\tau$ between hops), with $s < L$, the expected throughput is $\frac{\sum_{j=1}^{s}(1-j/L)}{s+1}$ which reduces to

$$\frac{s}{s+1} - \frac{s}{2L}. \tag{4.1}$$

For a poor choice $s \geq L$, the throughput is $\frac{L-1}{2(s+1)}$.

Note a simple generalization, where it takes the jammer time τ to switch channels and identify the channel's use by the legitimate connection, but the legitimate communication takes time $\alpha\tau$ to complete its hop. α can increase due to imperfect synchronization between the AP and client, but may also decrease due to the time required by the jammer to identify traffic on the channel it has just moved to. When $s < L$, Equation (4.1) then extends to:

$$\frac{2Ls - s(s+1)}{2L(s+\alpha)}. \tag{4.2}$$

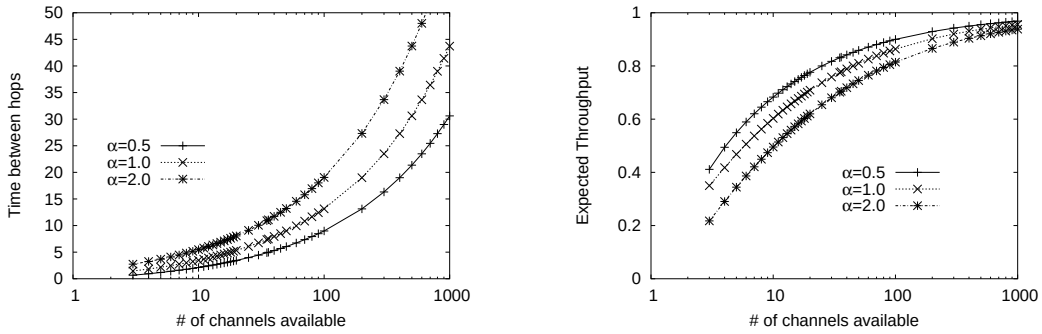
For $s \geq L$, the throughput is $\frac{L-1}{2(s+\alpha)}$.

The optimal residence time is easily determined solving the derivative with respect to s equal to 0, yielding $s = \sqrt{\alpha^2 + (2L-1)\alpha} - 1$. For $\alpha = 1$, this reduces to $s = \sqrt{2L} - 1$.

The maximum attainable throughputs for general α can be obtained by plugging in the above value for s into (4.2). For $\alpha = 1$, this reduces to

$$1 - \frac{2\sqrt{2L} - 1}{2L}.$$

For example, when $\alpha = 1$, $L = 3$ (e.g., 802.11b orthogonal) yields a throughput of approximately 0.35, and when $L = 12$ (e.g., 802.11a orthogonal), the throughput is approximately 0.63.



(a) Optimal time between hopping

(b) Throughput

Figure 4.2: Optimal wait time and expected throughput with jammer sequencing through channels

Figures 4.2(a) and 4.2(b) respectively plot the optimal residence time between hopping and optimal throughputs as the number of channels, L, is varied along the x-axis. The different curves plot these values for different values of α. Note that both optimal time and optimal throughput are relatively insensitive to changes in α. In Figure 4.2(a), the time is in multiples of the time it takes the card to switch channels. Noting that the x-axis is log-scale, the time is sub-linear in the number of channels. We see that for conventional 802.11, the client must switch at a rather high rate and cannot spend significant time on a channel. In Figure 4.2(b), the expected throughput is normalize to what can be achieved in a jam-free setting without hopping. Conventional 802.11 standards see significant throughput degradation. For instance, if only the 3 orthogonal channels in 802.11b are used, when $\alpha = 1$, one can expect less than half of the throughput when there is no hopping and no jamming. While this is undesirable, the results are rather impressive, considering there are only 3 channels to hop upon. With 11 channels, we can achieve over sixty percent of the throughput when there is no jammer, and with 36 channels, almost 80% of the jam-free throughput.

4.4.1.1 Varying Residence Times

To further obfuscate the channel switching sequence, it is possible not only to vary the channel, but also have a pseudo-randomly varying residence time for which the communication remains on a channel before hopping. Such information can also easily be encoded into a pseudorandom sequence: for instance, the odd

entries in the sequence can indicate the channel while the even entries indicate the residence time for that channel.

To simplify the analysis, we assume that each residence time is a multiple of τ. Define p_j to be the fraction of residence times that are at least of length τj. Clearly $p_1 = 1$ (it never makes sense to switch to a channel and switch again before trying to use it) and $p_j \geq p_{j+1}$. The expected throughput becomes

$$\frac{\sum_{j=1}^{\infty} p_j(1 - j/L)}{\alpha + \sum_{j=1}^{\infty} p_j}.$$

It is easy to show that an optimal throughput occurs when there is some i for which $0 < p_i \leq 1$, $p_j = 0$ for all $j > i$ and $p_j = 1$ for all $j < i$. In other words, the difference in the times between hops should differ by at most τ. This is easily proved by contradiction. Due to space limitations, we only sketch the proof. Noting that the p_j are monotonically non-increasing, assume the claim is false such that $0 < p_j, p_{j+1} < 1$. In the absence of an jammer, this system has some throughput T. Note that by increasing p_j to 1 and decreasing p_{j+1} accordingly, we could maintain the same throughput T. The jammer is less likely to successfully jam during the time interval $[(j-1)\tau, j\tau]$ than during interval $[j\tau, (j+1)\tau]$. Hence, the above shift will decrease the expected amount of time during which jamming is successful, increasing the resulting expected throughput, contradicting the claimed optimality. By reducing the granularity of the p_j from τ to an ϵ as small as desired, this process can be continued until the two possible lengths of time that a residence time can take differ by less than ϵ.

4.5 Evaluation

In this section, we begin with a description of our prototype implementation and the experimental setup. We then proceed to characterize the channel switching times for conventional hardware and the performance observed by clients while hopping channels without a jammer and in presence of the jammer. For our experiments with the jammer, we identify the optimal parameters for our jammer (when the client throughput is minimized) through experimental observations.

4.5.1 Prototype Implementation

We implemented the channel hopping as a loadable kernel module for Linux-2.4.26 kernel. This module interacts with the madwifi driver and the in-kernel IEEE 802.11 protocol implementation. The system has three primary components, accurate and efficient timeout mechanism, `End-of-Slot` message transmission, and switching the channel while maintaining the protocol state. In the following we discuss each of the following in detail.

4.5.1.1 Timeout Mechanism

We implement software timers similar to those proposed by Aron et. al [21], which uses entry points into the kernel, e.g. hardware timers, system calls, interrupts, to maintain time in software. A study of the timeout mechanism reveals that 90% of the time, the timeout triggers within 1ms of its scheduled time (due to lack of space, details are relegated to a technical report). Our mechanism does not suffer from variance due to CPU load and other system activity.

4.5.1.2 End-of-Slot Transmission

The AP broadcasts an end of slot message before starting the channel switch. This message is sent as a management packet, using an unused vendor-specific identifier included in the IEEE 802.11 implementations. Since our timer mechanism can initiate a switch from the interrupt context, the packet cannot be allocated inline, instead we allocate the packet once at initialization, and reuse it through the operation of the system. Our hardware supports multiple queues which carry traffic with different QoS requirements. We use the highest priority hardware queue for this transmission to prevent queueing delays at the network interface.

4.5.1.3 Channel Switch

To perform a channel switch, the OS must stop all active DMA engines, cancel any timeouts, clean up interface specific state (e.g. mapped DMA buffers) and reset the interface with the new channel. The IEEE 802.11 state machine is not altered during this operation. This allows the associations with the clients to be maintained while switching the channel.

4.5.2 Experimental Setup

All nodes, client, AP, and jammer nodes are equipped with a Atheros 5212 a/b/g wireless mini-pci cards and run the Linux-2.4.26 kernel with the mad-wifi driver. To evaluate our system with the best possible jammer, the jammer is physically placed between clients and the AP to avoid hidden terminal and packet capture phenomena which reduce its effectiveness.

In our experiments, we restrict the channels used by the legitimate communication to a set of 12 non-overlapping channels in the 802.11a, 5 GHz band. Our jammer is aware of this limited spectrum utilization, and therefore only needs to do a sweep of these 12 channels in order to detect the channel used by the legitimate communication. We note that this restriction favors the jammer, as there are fewer channels to scan and the jammer is assured of a direct hit when it locates the channel with traffic. Unless specified otherwise, the residence time in a channel for the clients and the AP is 100ms.

4.5.3 Results

4.5.3.1 Channel Switching Latency

We first evaluate the latency involved in switching the operating channel of the wireless card. This operation involves setting some registers on the wireless card with appropriate values and performing a software reset of the card. In addition, the transmit queue containing packets waiting to be transmitted is drained before this operation.

Figure 4.3 shows the cdf plot of the switching latencies obtained for 500 channel switch operations where channel residence time is fixed at 50ms. The distribution is bimodal, with a large percentage of cases (65%) the switching latency lies between 3ms and 5ms. In the other 35% of cases, the switching time lies between 13 and 15ms. We conjecture that higher latency is due to the time required to drain the transmit queue, which is a function of the instantaneous traffic on the node.

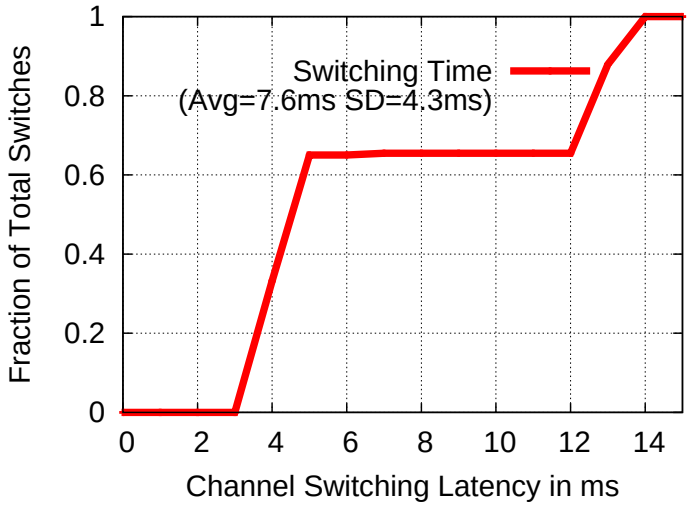


Figure 4.3: CDF of Channel Switching Latency.

4.5.3.2 Jamming Parameters

Figure 4.4 depicts the throughput of the legitimate connection as the listen and jam interval are varied. In this Figure, the AP transmits at 30Mbps, and the residence time is set to 100ms. The jam interval is varied on the x-axis, and the different curves depict different jammer listen intervals. The trend is rather flat: changing the lengths of either of these intervals seems to have little effect. However, a listening time of 5ms appears best. We suspect that a 1ms interval may miss traffic, and the 20ms interval is overkill. The optimal jam time is 50ms in this instance.

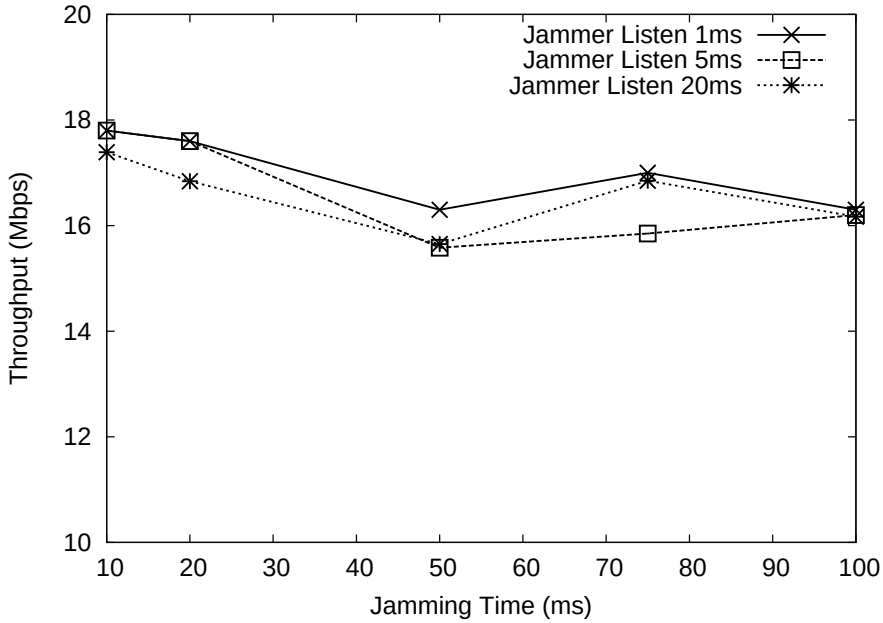


Figure 4.4: Throughput for different Jamming Parameters.

4.5.3.3 Jamming Overheads

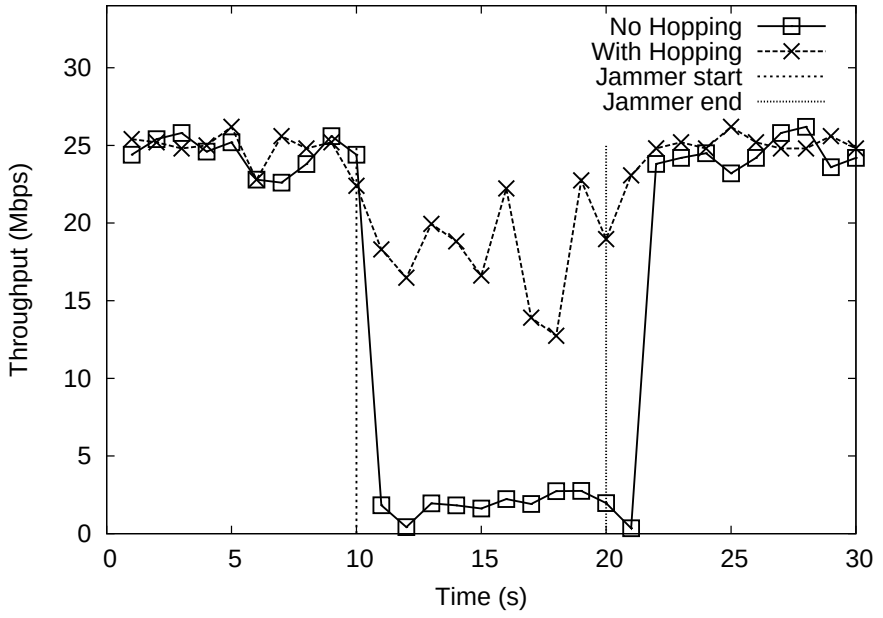


Figure 4.5: Throughput over time

Figure 4.5 depicts an instance of two experiments. In one, the legitimate transmission does not hop. In the second, it does. In both experiments, we use our scanning jammer who listens and jams (and never needs to change channels when jamming the non-hopping legitimate transmission) with a listen interval of 5ms and a jam interval of 50ms. In both experiments, the jammer starts jamming at 10 seconds and turned off at 20 seconds. While throughput degrades in both experiments during the jam, the degradation is significantly less when the legitimate transmission proactively hops.

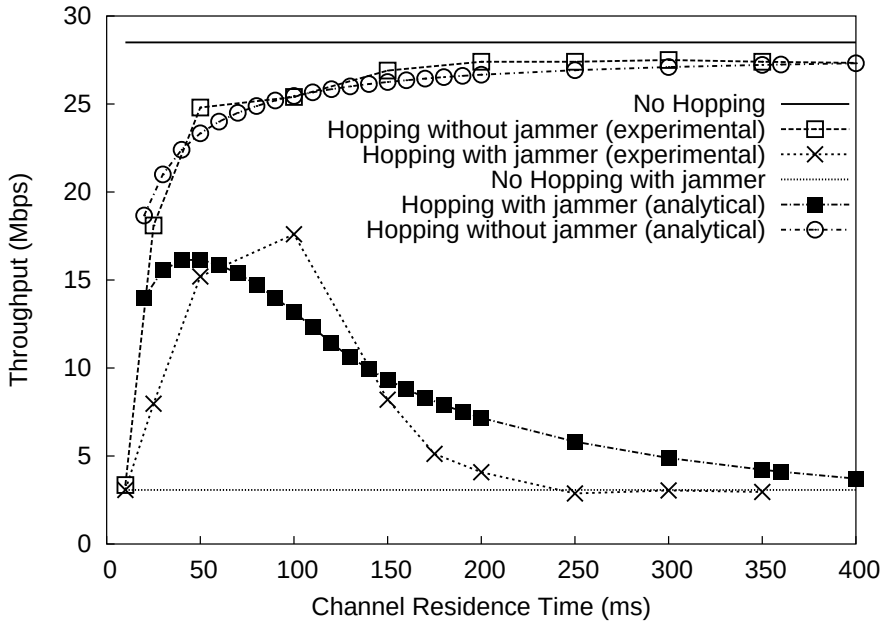


Figure 4.6: Throughput Vs Switching Frequency.

Figure 4.6 depicts the average throughput of the experiment. One figure is the average throughput in the presence of a scanning jammer, the other is in the absence of the scanning jammer. On the x-axis, we vary the channel residence time of the legitimate client. We also include plots using the mathematical analyses from earlier in this section: $s/(s+1)$ for the non-jammed transmission, $28\times$ Equation (4.2) with $\alpha = 1.5$ and $s = 10x$. [2]

We see that by channel hopping with the right residence time, the throughput achieved is approximately 60% by a jam-free, hop-free transmission. We note the analytical accuracy of hopping with no jammer is very precise, while there is some noticeable difference in the experimental result and the analytical model. While

[2]We multiply by y by 28 since full throughput here is 28 Mbps, whereas (4.2) normalized maximum throughput to 1. We multiply x by 10 since s is normalized to the time to switch, which, according to Figure 4.3, averaged close to 10ms.

$\alpha = 1.5$ was the best fit (in our opinion), we did not directly measure α (ratio of jammer listen period to legitimate transmission outage period), and may not be using the right α. Also, recall that the analysis assumes a "perfect" jammer who knows precisely when the legitimate transmission switches channels. The combination of these two inaccuracies may explain this difference.

4.6 Multiple Channels

Prior to this section, we have assumed a single legitimate communication that can utilize a single channel ($k = 1$). We now consider several extensions of our problem. First, we consider what happens when an AP has multiple interfaces such that it can simultaneously transmit and/or receive on multiple channels, but clients and the jammer have only a single channel. Then, we consider the case where each client and the jammer can simultaneously utilize on multiple channels as well.

4.6.1 Multiple Channels at the AP

If the AP has multiple interfaces, allowing it to transmit and receive on multiple channels, while each legitimate client has a single interface, there are two methods to use multiple channels can be utilized. The first associates two channels to a single client, while the second associates a channel with each client. When there are more channels than clients, both methods can be implemented simultaneously.

4.6.1.1 Soft Handoff

As mentioned earlier, the outage that occurs is the result of two phenomena. The first phenomenon is the time required to execute the code that sets the interface to the new channel. The second is the additional time incurred due to a lack of exact synchronization of client and AP switching times. This second phase can be eliminated by having the AP initiate a duplicate transmission on the channel being hopped to before terminating the transmission on the channel being hopped from. The benefit is conceptually similar to what is experienced by a mobile node that is transiting between two access points. The AP can terminate transmission on the channel being hopped from once the AP has identified the client on the new channel, or, if the client cannot be identified by the AP (e.g., the AP is transmitting in broadcast mode) it should be sufficient to wait for a short period of time.

Note that early initiation of the channel being hopped to can be observed by a jammer, so care should be taken not to start the transmission too soon. In contrast, continuing the transmission on the channel hopped from can be used to fool or "catch" a jammer, causing it to jam an obsolete channel.

4.6.1.2 Dealing with Multiple Clients

We consider an AP, operating in a spectrum with L channels, that contains $k \leq L$ interface cards such that it can simultaneously transmit and receive on k channels. Suppose there are N clients that associate to this AP. If $N < k$, then the AP could, in theory, assign each client to a different channel to increase the utilization of the spectrum.

We distinguish between two classes of jammer here. The first is the *outsider* who is not considered by the AP to be a legitimate client. Such a jammer has no knowledge of the pseudorandom sequences utilized by the legitimate sessions. If clients are assigned to different channels, a jammer has a greater likelihood of jamming *some* legitimate transmission, but the likelihood of any particular transmission being jammed does not increase.

The other type of jammer is the *insider*, who pretends to be a legitimate client and is assigned its own pseudorandom sequence. Knowing that other clients will not be assigned to the same channel gives this jammer a small piece of knowledge that increases its ability to jam (it can reduce the spectrum it needs to scan by one channel). We are also unaware of prior work that looked specifically at the problem of assigning pseudo-random sequences to a set of clients such that for all i, the ith entries in each client's sequence are distinct. A simple method for implementing this procedure is discussed in the extended technical report of this work.

However, it is simpler to permit each client to follow its own individual PSR, permitting clients to occasionally associate to the same channel at the same time.

4.6.1.3 A 2-dimensional PSR

Note that if $N > k$, it becomes necessary to allow multiple clients to associate to the same channel. A problem with allowing each client to have its own PSR is that the number of channels required by the N may occasionally be larger than k, stranding a client without a channel.

This case can be handled using a 2-dimensional PSR. An AP utilizes one PSR to assign interfaces to channels, which we call the *interface PSR*. This PSR is communicated to all trusted clients, such that an outsider jammer will not know (without probing) the channels being utilized by the interfaces. Then, each client is assigned its own unique PSR which indicates the *interface* to which the client is bound. The client then uses the interface PSR to look up the channel upon which this interface will be transmitting. Note that the process described in the Appendix can be used here to ensure that no two interfaces are simultaneously assigned to the same channel.

4.6.1.4 Unused Channels

If an AP has more interfaces than clients, it can transmit dummy traffic on these other interfaces on unused channels to confuse the jammer.

4.6.2 Multiple Channels at the AP and Client

To this point, we have considered a jammer interested in jamming a single legitimate communication, where both the legitimate communication and the jamming signal are each capable of only transmitting on a single channel at any given time. Today's wireless hardware is unable to simultaneously transmit on multiple channels. However, with the expectation that this limitation will soon be overcome, we wish to explore how multiple channels might be used by the legitimate communication to increase resilience to the jammer, as well as by the jammer to strengthen its attack. In addition, it is expected that the development and deployment of cognitive and software radio [47, 43] will vastly increase the available spectrum and hence expand the number of available channels. Our analysis here permits us to consider how a significant increase in the number of available channels increases the resilience of channel hopping to jamming.

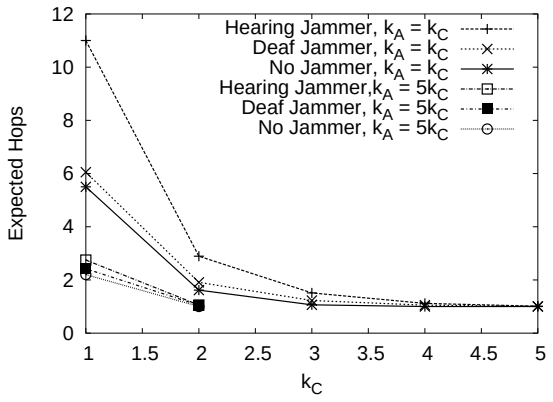
In this section, we assume that both the legitimate transmission and the jammer are also able to transmit on multiple channels. We use k_C, k_J, and k_A to respectively equal the number of channels that can be simultaneously utilized by the client, jammer, and AP. Normally, we expect that $k_A \geq k_C$, $k_A > k_J$, and that k_C and k_J are close.

We consider a single residence interval, in which the AP chooses k_A channels from the set of L channels, the client chooses k_C channels from a set S_C of size $\alpha \in \{k_A, L\}$ and the jammer chooses a set of k_J channels from a set of size $\beta \in \{[1, k_C], k_A, L\}$. This general model captures several cases of interest:

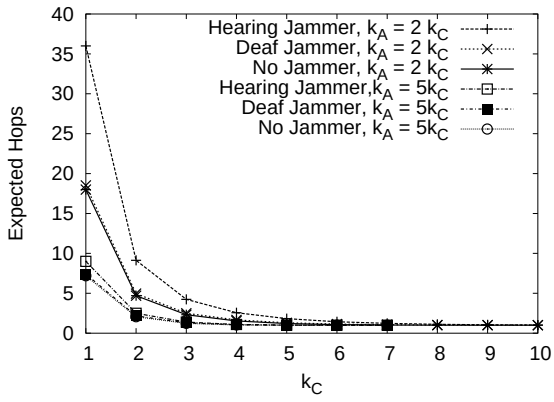
- If the client and AP utilize a pseudorandom sequence, then S_C is the set of channels chosen by the AP with size $\alpha = k_A$. If the client and AP are unable to utilize a pseudorandom sequence and must switch to true random, then $\alpha = L$.

- If the jammer is able to scan all channels at an infinitely high rate and can hear the exchange between the AP and client on i channels where $k_J < i \leq k_C$ channels used for this exchange, then the jammer can jam $\beta = \min(i, k_J)$ channels. Note that it is possible that $i < k_C$ when the client and AP are randomly (and not pseudorandomly) choosing channels. If the client remains silent such that the jammer can only hear AP transmissions, then $\beta = k_A$. If the jammer has insufficient time to scan and must select the channels it jams at random, then $\beta = L$.

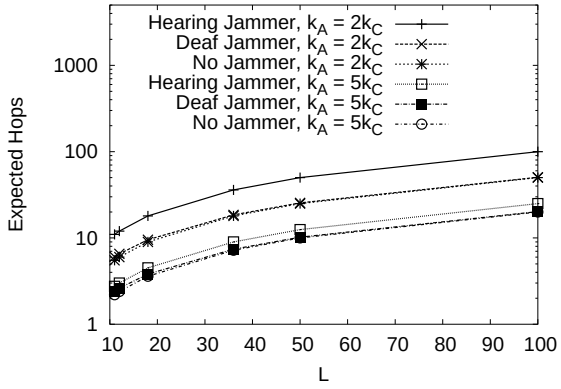
We compute two measures under this model:



(a) $L = 11$



(b) $L = 36$



(c) $k_C = 1$

Figure 4.7: Random search performance

- the probability that the transmission on *some* channel from (to) the AP to (from) the client is successful (not jammed).
- The *expected number* of these successful transmissions.

The first measure gives the expected throughput of communication between the AP and the client in the case that the same information is transmitted on each channel. The second measure gives the expected throughput of this communication if different information is transmitted on each channel. The first measure is useful for real-time data that must be delivered with little latency, and is also useful for the connection establishment phase when the AP and client are trying to exchange small pieces of information in sequence. The second measure is useful for file transfers where the goal is to maximize the push rate of data, and piece together the file later (easily accomplished via smart coding techniques such as [28].

Let T_{pm} be a random variable equal to the number of successful transmissions when the client and AP select channels according to a pseudorandom sequence. To compute $E[T_{pm}]$, we let X_i be a random variable that equals 1 if the ith interface of the client successfully received a transmission (i.e., the jammer did not block it), and 0 otherwise. Then $E[T_{pm}] = E[\sum_{i=1}^{k_C} X_i] = \sum_{i=1}^{k_C} E[X_i] = k_C \Pr(X_1 = 1)$ since the X_i are identically distributed and are indicator r.v.s. Thus,

$$E[T_{pm}] = k_C(1 - k_J/\beta).$$

If the same data is sent on all channels, then the expected throughput, $E[T_{ps}]$ equals the probability that some transmission makes it through:

$$E[T_{ps}] = \begin{cases} 1 & k_C > k_J \\ 1 - \frac{\binom{\beta - k_C}{k_J - k_C}}{\binom{\beta}{k_J}} & k_C \leq k_J \end{cases}$$

If we assume that the client and AP do not utilize a pseudo-random sequence, but must also coordinate, we can adjust the above formulas:

$$E[T_{rm}] = k_C(1 - k_J/\beta)(k_A/L)$$

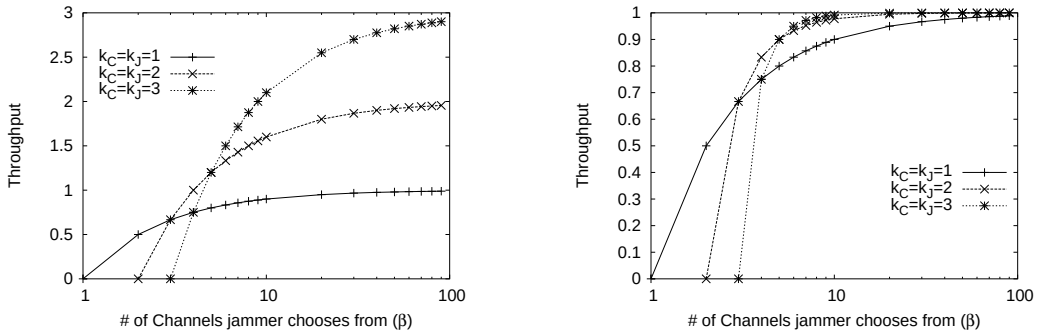
where $\beta = k_A$ if the jammer knows (immediately scans) the channels used by the AP but can only jam k_J of them ($k_J < k_A$), and $\beta = L$ when the jammer must guess the channels used by the AP.

$$E[T_{rs}] = \begin{cases} 1 - \sum_{i=1}^{k_C} \frac{\binom{k_A}{i}\binom{L-k_A}{k_C - i}}{\binom{L}{k_C}} & k_C > k_J \\ 1 - \sum_{i=1}^{k_C} \frac{\binom{\beta - i}{k_J - i}}{\binom{\beta}{k_J}} \frac{\binom{k_A}{i}\binom{L-k_A}{k_C - i}}{\binom{L}{k_C}} & k_C \leq k_J \end{cases}$$

where i iterates over the number of channels that the AP and client choose in common. When there is no jammer, this reduces to $E[T_{rs}] = 1 - \binom{L-k_A}{k_C}/\binom{L}{k_C}$.

Figure 4.7 considers the time it takes for an AP and client to "meet" on a common channel when each uses a truly random sequencing of channel hops as opposed to pseudo-random. This truly random sequencing is needed initially, before client and AP can agree on a pseudo-random sequence, and may also be necessary if jammed for a sufficiently long time whereby the hopping sequence falls beyond an entire residence interval. In all three sub-figures, the y-axis plots the expected time (in number of times hopped) between intervals where the client and AP share a common channel. In Figures 4.7(a) and 4.7(b), we vary k_C on the x-axis. The different curves represent different jamming scenarios: a "deaf" jammer that must select its k_J channels from the set of all L possible channels, a "hearing" jammer who can restrict this set to the $\beta = k_A$ channels that are being used by the AP, and no jammer. In addition, we plot curves for the case where $k_A = k_C$ and where $k_A = 5k_C$. These two subfigures have $L = 11$ and $L = 36$ respectively. Figure 4.7(c) varies L on the x-axis and fixes $k_C = 1$. The different curves cover the different scenarios covered in the previous subfigures. In all figures, $k_J = k_C$.

We see that as long as k_A is large with respect to k_J, the jammer is unlikely to keep the legitimate connection separated for a long time, even when the hops are not sequenced identically. Furthermore, in the absence of a jammer, a larger k_A or k_C is still necessary to keep this time small.



(a) Multi

(b) Single

Figure 4.8: Throughput of Pseudorandom Sequences with Multiple Channels

Figure 4.8 explores the achievable throughputs in the presence of a jammer that can jam as many channels as the client can utilize. Figure 4.8(a) plots the expected throughput when different information is sent on the different channels, and Figure 4.8(b) plots the expected throughput when the same, redundant information is sent down these different channels. The y-axis plots the expected throughput, while the x-axis varies β. Hence, the x-axis can be thought of as L when the jammer is "deaf" and as k_A when the jammer "hears". Noting that the x-axis is plotted on a

log-scale, we see that by increasing β, a jammer is again easily thwarted, and the throughputs quickly reach their asymptotic maximums. It is interesting to note that when k_A is very small, the jammer benefits from larger values of $k_C = k_J$, but this benefit is quickly diminished. Note that by $\beta = 10$, the asymptotic maximums are almost already reached. Hence, in conventional 802.11, if an AP transmits noise on channels not used by legitimate communications, this should be enough to sufficiently thwart a jammer.

4.7 Discussion

Our work thus far has made two critical assumptions that allowed us to explore the efficacy of proactive channel hopping in the presence of a jammer. Before concluding, we briefly consider two issues that our current prototype does not address.

4.7.1 Wasted Throughput in the Absence of a Jammer

Since the focus of this work is to develop a mechanism to resist jamming, there is an underlying assumption that a jammer is present in the network. To protect against this assumed jammer, our legitimate communication hopped channels roughly twenty times a second. While the loss in throughput was minimal, it may still be desirable to further minimize the loss.

A straightforward solution is to develop a 2-state channel hopping procedure. One state is a *reactive* state, where the presence of a jammer is known or is highly suspect. In this state, the hopping procedure follows our previous description. If no jammer is believed to be present, the hoping procedure can reside in a *passive* state. In the passive state, only every nth hop is executed for some $n > 1$. Over time, the reduced hopping rate reduces the loss in throughput due to the outage that occurs during each hop. However, hopping is not stopped, so that if a jammer appears, the client and AP can still hop to a jam-free channel. In addition, if one node moves to the reactive state while the other remains in the passive state, the pair will synchronize channels each time the node in the passive state hops. This permits the node in the reactive state to request that the node in the passive state move to the reactive state.

4.7.2 Soft Handoff

Our existing implementation does not account for client nodes that may seek to handoff to an alternate AP. Implementing a soft handoff in an 802.11 infrastructure environment where legitimate communications are expected to hop channels is not straightforward. However, we touch briefly upon two ways to implement soft handoff while channel hopping.

Existing 802.11 association protocol works under the assumption that the APs are operating on a fixed channel. A client node sends a probe request message on a channel and waits for a probe response messages from APs in the vicinity that are on that channel. The client sequentially probes all channels to gather information about all APs in the vicinity and finally chooses an AP with the highest received signal strength. The client then initiates the 802.11 association handshake protocol with the chosen AP.

Since the AP's are hopping channels, the client can continually send probe messages on a single channel (or on channels it hops across) to gather the responses. After choosing an AP with the best received signal strength, the client initiates the handshake protocol where the client sends association request message to the AP and waits for a response.

Another technique is to broadcast the current slot number in the pseudo-random hopping sequence inside every beacon packet. Assume that all trusted clients, like the ones inside an enterprise have the knowledge of the starting seed of a PSR corresponding to an AP. Thus a client wishing to associate with an AP, first determines the current slot number and then synchronizes the hopping sequence with that of the AP. It then starts the association protocol. In addition, existing standard authentication and encryption protocols like 802.11x can work on top of this hopping sequence.

4.8 Conclusion

In this chapter, we explored the feasibility of implementing channel hopping within 802.11 to protect a legitimate communication from jamming attempts. We begin by evaluating the best channel scanning and jamming strategy that a jammer can implement to reduce the throughput of a legitimate communication that uses channel hopping to resist jamming, and then explore how to best tune the channel hopping strategy to resist such a smart jammer. We then experimented with this channel hopping strategy in the presence of the jammer.

Our experimental results showed that, in practice, while there is a degradation in throughput, this degradation is minimal in the absence of a jammer. Furthermore, in the presence of a jammer, throughput can be maintained at more than half the rate that exists in the absence of a jammer. Our analytical results extrapolated how to develop AP-client hopping systems in environments with multiple clients, multiple cards, and an increase in usable spectrum. These results show that as spectrum flexibility is increased, channel hopping will become even a more effective means to protect 802.11 wireless networks from jamming.

Chapter 5

Client-Aware Routing Protocol for Seamless Mobility in Mesh Networks

5.1 Introduction

We investigate a mesh networking architecture as an alternative to wireless LANs based on IEEE Standard 802.11 [13]. In a mesh network, *wireless access routers* are deployed to cover a region where wireless access is desired, much like the way access points are deployed in a traditional wireless LAN. However, unlike access points in a wireless LAN, the access routers are not connected to a wired infrastructure. They are rather interconnected via wireless links to form a backbone wireless network. The mobile client nodes (e.g., laptops and palmtops) still associate with one nearby access router, oblivious of the nature of the backbone connectivity.

This method of eliminating the "wires" from the wireless LAN provides a significant deployment advantage. It is envisioned that with plummeting cost of IEEE 802.11-based networking interfaces as well as access point/router platforms, mesh networking will become as ubiquitous as wireless LANs, and will "blanket" communities with wireless coverage at a low cost. Several usage scenarios have been envisioned [18]. Examples include (i) broadband connection sharing for "last-mile" access; (ii) neighborhood or community mesh networking, where a mesh network parallel to the Internet is used for applications of local relevance, such as sharing data or multimedia, or collaborative backup; (iii) community-wide or metropolitan-area wireless networks specifically used for niche applications, such as law-enforcement, emergency management or traffic systems; (iv) any application where rapid deployment of a wireless network is desired over a wide area. Noting these usage scenarios and their potential economic advantages, several companies are exploring commercialization of mesh networking, such as Tropos [117], Packethop [85], Meshnetworks [69], Firetide [40] etc. There are several community

initiatives as well using commodity 802.11-based hardware platforms (see, for example, [102, 23, 30]).

The goal of this work is to design and evaluate a wireless mesh network architecture for community networking applications. The goal is to be able to provide seamless networking services to the mobiles both for last mile access and peer-to-peer access. Our architecture uses 802.11b-based access points (AP) that also double as routers thus providing the service of a wireless access router, following the terminology used before. We will refer to them as APs, or access routers, or mesh routers. The fundamental design goal that we pursue is *client side transparency*. The client mobile stations[1] are unaware of the mesh networking backbone. They view the network as a conventional wireless LAN spread out over an extended geographic area. Thus the clients still associate with an AP using a traditional association mechanism in wireless LANs. When the client moves and re-associates with a different AP, a layer-2 handoff event occurs that in turn triggers appropriate routing updates in the mesh network backbone. Thus, the handoff process involves both layer-2 and layer-3 procedures. We describe how the layer-2 and layer-3 handoffs work together efficiently, and present design choices for the layer-3 handoff process – one using a mobile IP [89] like solution called *Transparent Mobile IP* [116] and the other using a "flat" routing protocol based on link-state routing. We also present a detailed performance evaluation of the handoff latencies in both layers and their impact on transport protocol performance.

The rest of the chapter is organized as follows. In the following section we develop our system architecture. In Section III, we describe the implementation details and in Section IV we present the performance results. Section V describes the related work. The conclusions are presented in Section VI.

5.2 System Architecture

For providing a complete client-side transparency, our design uses 802.11-based access points operated in the *infrastructure* mode. Thus, we refer to our architecture as *iMesh*. This is a departure from the common use of *ad hoc* mode in the experimental study of multihop wireless networks using 802.11-based radios (for example, used in experimental studies reported in [36, 67, 44]. This choice enables us to design the system without any specialized software on the mobiles. If the ad hoc mode were used, when a mobile moves away from an AP, it must find another appropriate "next hop" AP for forwarding its packets. To do this, it must possess the ability to discover its neighborhood via some layer-2 or layer-3 functionality that typical clients do not implement in the ad hoc mode. Note that ad hoc network routing implementations use such a mechanism to detect route breaks. For example, the existing link with the old access point can be considered broken when ACKs do

[1] We will use the words "client," "mobile" and "station" interchangeably in this article.

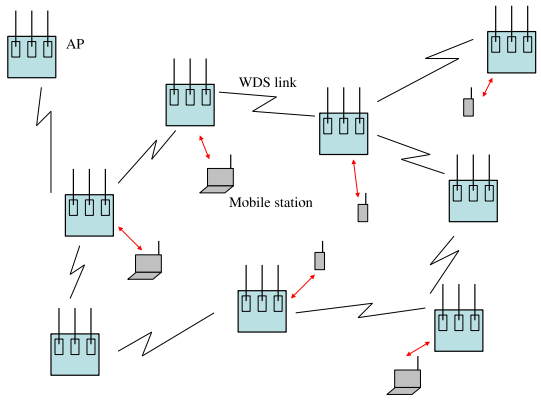


Figure 5.1: The *iMesh* architecture. Each AP can have multiple wireless interfaces (three shown) tuned to different bands/channels. These interfaces form a wireless backbone network using *Wireless Distribution System* (WDS) links. The mobile stations associate with a nearby AP (links shown using straight arrows) as in a regular wireless LAN unaware of the mesh routing architecture.

not come back in the MAC layer of 802.11 even after repeated transmission attempts. Similarly, layer 3 beacon/hello messages can be used. While these may be straightforward to accomplish, the approaches need the client device configured with appropriate software.

This requirement forces us to use infrastructure mode of operation on the AP. It is now sufficient to use the underlying "handoff" capability of the 802.11 client devices to handle mobility. Thus, the client's view of the network is still that of a wireless LAN, while the distribution system (DS) connecting the access points are now made of a wireless backbone network or *Wireless Distribution System* (WDS) [13]. The network interfaces at the neighboring access points use WDS links to communicate between them. A neighbor discovery protocol listens to the layer-2 beacon messages that a 802.11 interface emits in the infrastructure mode in order to configure the WDS links to the neighboring APs. A multihop routing protocol maintains end-to-end connectivity. When a mobile client moves and reassociates with a new AP, the reassociation triggers routing updates in the network so that the packets destined for the mobile can be delivered to this new AP for transmitting to the mobile. The *iMesh* network architecture is shown in Figure 5.1.

The access points are not mobile and are typically powered from a power outlet [18]. This brings about a couple of important design choices. First, the routing protocol can be proactive, such as based on traditional *link-state* or *distance-vector* approaches rather than on-demand approaches studied in connection with mobile ad hoc networks [88]. This is because the mesh network topology will be stable over longer time scales. Second, it is possible now to configure multiple wireless

interfaces on each access point, as optimizing power consumption on access points is no longer an important design goal. The interfaces could now be tuned to different bands (say, 802.11b, g or a) and channels within the same band for bandwidth aggregation and/or to exploit channel diversity in various interesting ways. Use of multiple interfaces on access points, however, gives rise to an interesting channel assignment problem. A WDS link between two neighboring access points can only exist when they have at least one interface on a common channel. Our design accommodates use of multiple interfaces per access point. However, a solution of the channel assignment problem is beyond the scope of this article. Some solutions are available in current literature [95, 18].

An important component of the *iMesh* architecture is how handoffs are handled. This is critical to support seamless mobility. In the following, we describe the approaches in two parts – link layer (layer-2) and network-layer (layer-3) handoffs.

5.2.1 Link Layer Handoff

When a station moves out of range of an AP, it triggers a link layer handoff to search for and reassociate with a new AP. The exact condition that triggers a handoff is implementation specific. For example, a client can initiate a handoff when it fails to communicate with the AP it is currently associated with. Or, the handoff initiation can be more proactive. For example, the client can continuously do signal strength measurements for the beacon messages from APs that it is hearing on the current channel. If the signal strength of the AP it is currently associated with falls below a threshold and the signal strength from another AP is sufficiently higher, the client may trigger a handoff to the second AP. The second condition avoids ping-ponging between two APs due to slight fluctuation of signal strengths.

Handoff is often associated with *probing*. Probing proactively seeks APs to associate with instead of waiting to hear beacon signals. This is because beacon intervals can often be too high (e.g., more than 100 ms). Also, there may not be any AP to associate to in the current channel. In probing, the client broadcasts a *probe request* frame. APs on the same channel respond with *probe response* frames. The client waits for certain amount of time (*probe-wait time*) to collect all the probe responses. Then, the client can switch to other channels to probe. After probing a set of channels (possibly all available channels), the client selects one AP with the best SINR (signal-to-noise ratio) based on the probe responses.

After probing is complete, the station authenticates with the new AP. Following successful authentication, the station initiates *reassociation* with the new AP to exchange information about the connection such as transmission rates, beacon intervals, etc. It sends a *reassociation request* frame to the AP that responds with a *reassociation response* frame. At this point, the link-layer handoff completes.

Several research studies have investigated link layer handoff latency in 802.11-based wireless LAN and various optimizations [71, 107, 106]. Our work has benefited a lot from these experiences. It turns out that the major factor in the handoff

delay is the time spent in probing and waiting for probe responses. Since probe responses may come back at different times (as they go through backoffs in the MAC layer to avoid collisions) too small a *probe-wait time* may miss important probe responses. Also, it is possible that the best possible AP to handoff to is on a different channel than the mobile station is on currently. Thus, the probing must be done in different channels in a sequential fashion. In each step, channel switching also adds to the delay. The probing can be optimized by only probing a small set of channels by exploiting prior knowledge [107, 106]. This has been shown to substantially minimize link layer handoff latency.

5.2.2 Network Layer Handoff

The original IEEE 802.11 standard [13] specified only the MAC and PHY layers of a WLAN system and defined the basic architecture, including the concepts of APs and DSs. However, the inter-connection and inter-operation of different APs in a DS was left as an implementation choice. Later, as 802.11 systems grew in popularity, certain DS related functions in the APs (particularly related to handoff and state exchanges between the old and the new APs during handoff) were specified as an extension – *Inter-Access Point Protocol or IAPP (802.11f)* [8]. The goal was to make APs from different vendors inter-operate across a DS. A very generic DS architecture is assumed, and the inter-AP communication was assumed to run over TCP/UDP/IP. An example of such inter-AP communication relevant to our work is the *move request* message from the new to the old AP after reassociation and a corresponding *move confirm* message from the old to the new AP with any context information (e.g., related to security) to be transferred. However, IAPP does not assume any particular architecture on the part of the DS, and does not specify how the inter-AP communications should be routed or even how the APs should get an IP address for such IP-based exchange to work. These are left as implementation choices.

In the *iMesh* architecture the APs form a multihop network routable at the IP layer.[2] This gives rise to a mobility management problem – how to deliver frames destined to a station when its point of attachment to the mesh network (i.e., the AP) has changed. Two broad approaches are possible that we both implement in our testbed and compare. The first approach uses a technique similar to mobile IP [89], where each station has a unique "home" location or a home AP. The network implementing the DS keeps track of the mobile stations. Packets destined for the station is still delivered to the "home" AP for propagating to the mobile station. It is now the home AP's responsibility to forward the packet to the AP the mobile station is currently "visiting." This is achieved by a protocol called *Transparent Mobile IP* or *TMIP* [116]. The significant difference from the standards-compliant

[2]Note that it is possible to do the routing using purely MAC addresses and using ARP and proxy ARP techniques intelligently [36].

Mobile IP is that the mobile station does not need to implement any specific protocol. This preserves the transparency we desire. There is a centralized server in the mobile network called *Mobile Location Register (MLR)* which keeps the information about the "home" AP for every mobile station. When the mobile hands off to any "foreign" AP, the foreign AP sends a query to the MLR to find out about its "home" AP. The foreign AP then notifies the home AP about the new endpoint of the mobile with a message handshake, and adds a new, one-hop route to the mobile. In addition, it updates the ARP table entry for the default gateway on the mobile client by sending a fake ARP response message containing IP address of the gateway and mac address of the foreign AP. Beyond this point, packets directed to the mobile are intercepted by the home AP are "tunneled" (using IP-in-IP encapsulation) to the foreign AP. The communication from mobile station, on the other hand, can proceed in the normal fashion without involving the home AP. Note that TMIP makes it possible that the mobile station keeps the original IP address even in its "foreign" location.

While the transparent mobile IP approach is straight-forward, as the forwarding path for the mobile is clearly not optimal due to the so-called "triangular routing" scenario. The approach that we promote here is to use a full-fledged multi-hop routing infrastructure in the network of APs in the DS. The routing infrastructure is "flat"; the routing tables in the APs contain the IP addresses of all the mobile stations in the system. Optimizations are possible for very large-scale networks to limit the size of the routing tables, though we do not discuss these here. The basic idea of maintaining the routing infrastructure is to use any handoff as a trigger to generate and propagate necessary routing updates. Thus, the network layer handoff consists of completion of notifications for the Transparent Mobile IP case and convergence of routing updates for the flat routing case.

In the *iMesh* testbed we have chosen a link-state based routing protocol, called Optimized Link State Routing or OLSR [32]. Link state based protocols have the advantage that loop-freedom is easy to derive. Also, having a complete link state database in each node in the network makes it easy in future to use complex routing policies and metrics. As will be discussed in the related work section, several mesh networking initiatives also do use link-state protocols.

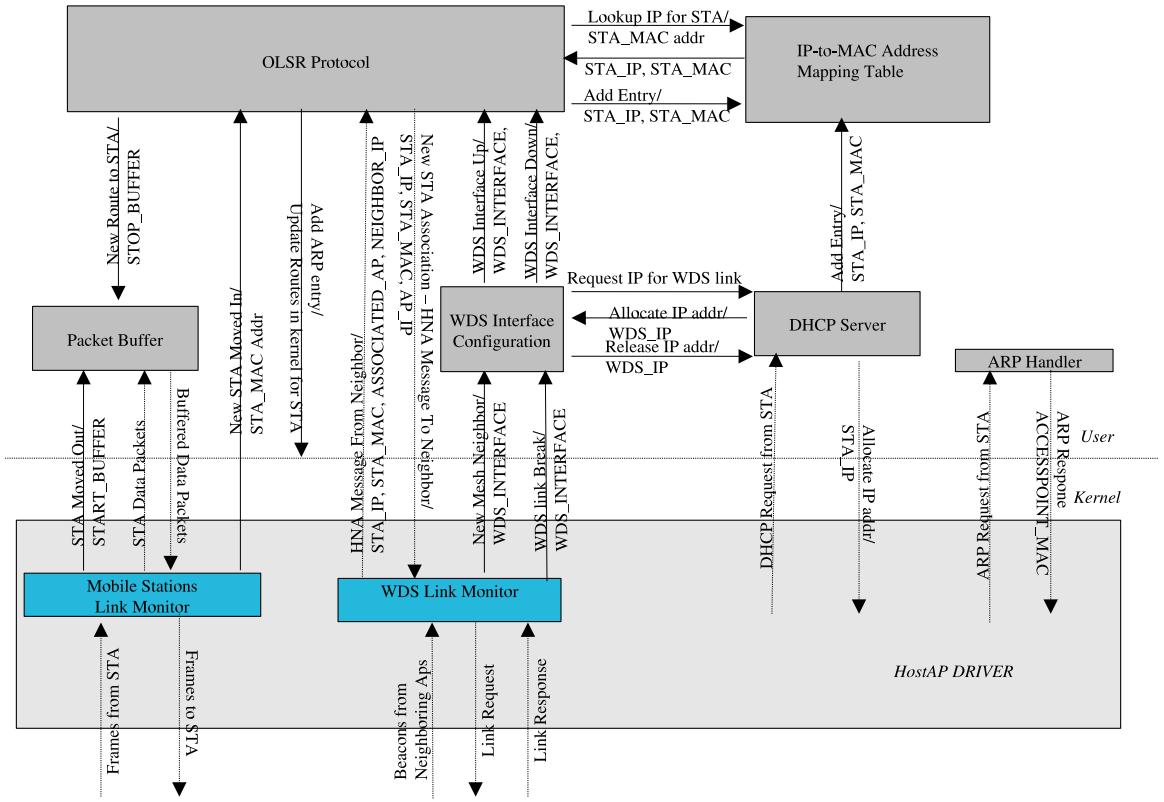


Figure 5.2: Block diagram of major software components in an *iMesh* access point. Actions are written above the arrow and the information passed below it.

5.3 Implementation Details

Use of a software-based AP is vital to our design – so that the functionality modifications for the AP and provisioning of the layer-3 handoff process (including the routing protocol) can be done easily. The HostAP device driver [65] and an embedded version of Linux are used to obtain a software-based AP functionality. In HostAP driver, there is an AP mode, other than the more usual client mode. In the AP mode various AP functions are supported in software – such as authentication (and deauthentication), association (reassociation, and disassociation), power saving (PS) mode, signaling and frame buffering for PS mode, support for WDS etc. The driver has also various features for development, debugging and researching IEEE 802.11 environments like access to hardware configuration records, I/O registers, and frames with 802.11 headers. Presumably, other software-based AP architectures such as Microsoft's native WiFi [70] can be used to build *iMesh*.

In the following, we elaborate on four relevant implementation details for *iMesh*. They are related to (i) auto-configuration of APs at startup, (ii) triggering of network layer handoff, (iii) implementation of the routing protocol, and (iv) packet buffering during handoff to improve performance. A block diagram of the AP protocol stack is shown in Figure 5.2. This figure will be occasionally referred to in the following sub-sections.

5.3.1 Auto-Configuration at Start-up

At start-up, the APs self-configure automatically to join the mesh backbone. The goal here is to automatically discover and create high quality symmetric WDS links with the mesh neighbors. The steps taken are as follows.

1. Configure the wireless interfaces to run as AP (i.e., in Infrastructure Mode) on a pre-defined channel and set the ESSid to a predefined string. This defines the identity of the ESS (extended service set) to be used by the mesh network.

2. Listen for beacons from neighboring APs for a configurable amount of time.

3. On hearing a beacon from an AP, check if this AP belongs to the same ESS by looking at the ESSid field of the beacon frame. If it does, then check the signal strength of the received beacon frame. If it is above a certain threshold value, then initiate the creation of a new WDS link with this AP. This involves a message handshake between the two APs. Send out a *Link Request* message unicast to the remote AP. This is a normal data frame using WDS addressing with a specific tag in the ethernet header so that the receiver can identify this as a handshake frame.

   At the Remote AP side, on receiving the *Link Request* frame, check again if the quality of the signal for the received frame is above the threshold. If the signal quality criterion is satisfied, the remote AP responds with a *Link*

Response frame. At the end of a successful handshake, each of the APs create a new WDS link. This process involves creation of a logical network interface by the HostAP driver that is exposed as a new interface to the upper layers (see the module *WDS Interface Configuration* in Figure 5.2). Upper layer protocols running on the AP can use a WDS interface (note that this is a logical interface) to communicate with a peer AP. For a packet sent on such a logical interface, the HostAP driver uses IEEE 802.11 standard compliant four address frame format. Two of them correspond to mac addresses of node peers for the WDS link, while other two addresses correspond to that of original source and destination for the packet.

4. Before a WDS interface can be used, it has to be assigned an IP address. We use a *user-level* DHCP server module running on each AP. Each of these servers are allocated a unique pool of IP addresses to allocate to the WDS links as well as to the mobile stations. When a new WDS link is created, the *WDS Interface Configuration* module delivers a *Request IP for WDS Link* event to the DHCP server for it to allocate an address. See Figure 5.2.

5. As soon as an IP address is assigned, a trigger is passed on to the routing module to include the interface in the routing protocol (*WDS Interface Up* in Figure 5.2).

5.3.2 Triggering Network Layer Handoff

The access points are implemented by suitably modifying the HostAP driver. The default handoff functionality in HostAP is switched off and handoff is implemented using *iMesh*-specific software implementation.

When a new mobile station joins the network, say by booting up and associating with one of the APs, the station acquires an IP address via DHCP from the address pool of this AP. Let us denote this AP by AP1. The mobile uses AP1 as its default gateway. AP1 maintains a mapping of IP address to MAC address of mobile stations in an *IP-to-MAC address mapping table*. AP1 then adds a host-specific route to this mobile in its kernel routing table. At this stage, the mobile station has complete uplink connectivity.

AP1 then advertises this new route to all other APs in the mesh network through a link state update via the OLSR protocol. The MAC address of the mobile is included in the update message so that all APs in the mesh network can add the IP-to-MAC mapping in their own IP-to-MAC address mapping table. Downlink connectivity is available to the mobile at the end of the link state update, as at this point all APs in the network have a host-specific route to the mobile station with AP1 as the last hop node in the route.

This IP-to-MAC mapping is required to be distributed for a reason. Recall that our design goal is transparency to the mobile station. When the mobile hands off to

another AP, this new AP must initiate routing updates so that the packets destined for mobile station can now be delivered via the new AP (say, AP2). However, since routing uses IP addresses, AP2 must learn the mobile's IP address right after reassociation.

Consider the scenario where the mobile station has an active connection to another host in the mesh network or a host in the Internet via a gateway AP. Initially, it was associated with AP1 as described before. Now assume that it reassociates to AP2. The HostAP driver learns about this new association and notifies the OLSR protocol (*New STA Moved In* in Figure 5.2), which in turn determines the mobile's IP address by looking up the *IP-to-MAC address mapping table*. Since a mobile station can be associated with only one AP at one time, AP2 deletes the existing route of the mobile (that would be using AP1 as the last hop node) and adds a new one-hop route. Then it triggers a link-state update. The OLSR protocol takes care of propagating the update network-wide and route recalculations at every other AP in the network.

5.3.3 Routing

The OLSR protocol runs on all WDS interfaces at every AP. Note again that separate *logical* WDS interfaces are created for each neighboring AP. The AP does not run OLSR on its client side interface (the logical interface the client associates to – typically `wlan0`) as the client is unaware of the routing. The link between the AP and mobile station is treated as an *external route* to the mesh network. The OLSR protocol advertises such external routes via the so-called HNA (Host and Network Association) messages [32] designed specifically to inject external routes to the mesh network.

Whenever a mobile station associates with an AP, the HostAP driver sends an association signal (*New STA Moved in* in Figure 5.2) to the OLSR daemon, which deletes all pre-existing routes to this station and adds a "direct" route to the client via its `wlan0` interface. This "external" route information is encoded as an HNA message and broadcasted in the network via the OLSR protocol. All APs, on receiving the HNA message, delete all pre-existing routes to this station and add a new route via the AP to which it is currently associated. Also, on receiving HNA, the AP deletes the information about this station from its local database of external routes.

As mentioned before, the IP-to-MAC layer mapping for an associated station needs to be propagated across the network too. The IP-to-MAC mapping for that station is piggybacked on the route update. To accomplish this, we change the HNA message header (destination IP address and subnet mask) to contain an extra field for the MAC address of the destination, which in this case is the mobile station.

5.3.4 Packet Buffering

When a mobile station initiates a link layer handoff, it loses connectivity with the old AP and establishes connectivity with the new AP. As a result, when handoff is in progress, packets sent out by the old AP to the mobile station are lost. Buffering can be used to alleviate the packet losses during handoff.

We have implemented a packet buffer in user space using the *netfilter* framework [7] provided in the Linux kernel. When the AP receives an explicit deauthentication frame from an associated station or when successive frames sent to the station are unacknowledged,[3] it assumes that the station has switched to another AP. Now, using a hook into the pre-routing stage of the IP stack, all incoming IP packets are examined. If the destination IP address of a packet matches with the IP address of the mobile station that has lost association with the current AP, the IP packet is queued in the buffer. Buffering continues until the new location of the mobile is learned through the OLSR protocol and the route is appropriately changed. All buffered packets destined to this mobile are now re-injected back into the IP stack. These packets are now transmitted using the new route.

5.4 Performance Evaluation

5.4.1 Description of Testbed

Our *iMesh* testbed uses Soekris Engineering net4521 [111] processor boards, even though only one interface has been used for the work reported here. This compact, low-power, low-cost, advanced communication computer is based on a 133 Mhz 486 class processor. It has two 10/100 Mbit ethernet ports, up to 64 Mbyte SDRAM main memory and uses a 128 MB CompactFlash module for program and data storage. The board was expanded using a MiniPCI type III 802.11b interface and two PC-Card/Cardbus 802.11b interfaces. The interfaces are based on Intersil Prism 2.5 chipsets. The cards are connected to external rubber duck antennas via a pigtail. The processor boards run Pebble Linux V41 distribution [87] (a small Linux kernel suited for embedded devices) with the Linux-2.4.26 kernel.

Our testbed uses six APs and one mobile station. For programming and debugging convenience we have not used Soekris boards in all the APs. Four of the APs and the mobile station are IBM Thinkpad R-series laptops running Redhat 8 distribution with linux-2.4.22 kernel. The remaining two APs are Soekris boards as described before. Each AP is equipped with a single Prism chipset based 802.11b PCMCIA card. We used two types of cards in our testbed, manufactured by En-Genius Technologies and US Robotics. Each card is connected via a pigtail to an

[3] A timeout mechanism is used to take care of false alarms, i.e., situations when buffering is started due to packet losses caused by transient channel errors and not due to the movement of the mobile.

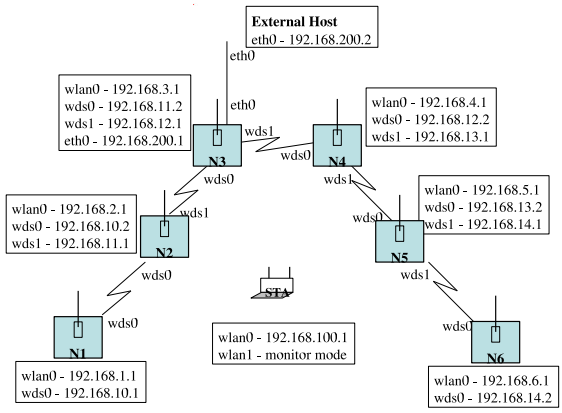


Figure 5.3: *iMesh* testbed used for performance evaluation. $N1$ to $N6$ are the APs with WDS links forming a linear topology. $N3$ is also connected to an external host via the Ethernet interface. STA is the mobile node. For each node in the network, the logical interfaces (`wds0`, `wlan0` etc.) are shown. Each interface has an IP address. The routing protocol operates on the `wds` interfaces. The `wlan` interfaces are for AP to mobile communication.

external antenna. We use the new firmware version v1.5.6 for the cards since this version supports the 4-address WDS frames. As mentioned before, we have used our customized version of open source HostAP driver (0.1.3 version) on each of the APs. The wireless interface on each AP is configured to work in infrastructure mode (*master mode*). All nodes operate on the same channel (channel 1) and they are assigned the same ESSid.

For ease of performing experiments all nodes are deployed close to each other in the same laboratory room and WDS links are formed manually to control the topology of the mesh network instead of using the auto-configuration protocol. We set up a linear topology as shown in Figure 5.3. The mobile station in the experiment is actually kept stationary, and its movement is "simulated" by changing its association to the APs through a script. This makes the experiments repeatable, and stable performance data could be collected.[4] This puts all nodes in the same collision domain and throughput performance suffers in comparison to a true multihop mesh network. Thus, we will be at best *underestimating* the performance numbers. Nevertheless, the reader will soon note that excellent handoff latencies are obtained. Both layer-2 and layer-3 handoff latencies are equal or better than that recently reported in literature [106, 107, 34, 104] for wireless LAN testbeds where layer-3 procedures, when they exist, are run on "wired" ethernet.

The mobile station has two wireless interfaces: PCMCIA card and MiniPCI

[4]We are working on extending the testbed to a wider geographic region so that true multihop links can be formed and using a robotic platform to move a mobile station to get repeatable measurements.

card. The PCMCIA card is configured as a client. This interface is used for associating with an AP. The MiniPCI card is configured in the *RF monitor mode* to sniff all packets on channel 1. We used this interface to collect packet traces during the experiments, enabling us to measure latencies of layer-2 and layer-3 events by a "post-mortem" analysis of packet traces.

Two flavors of wireless interface driver software are used on the client card – regular HostAP driver working in the client mode and a modified *airjack* driver [22]. The airjack driver is capable of sending and receiving management frames in software. More will be said about the client-side drivers momentarily.

Note that specialized drivers are used only to facilitate experimentation (associations need to be changed under program control) and to analyze layer-2 handoff latencies better. The *iMesh* network operation is independent of any specialized support on the client side. To simplify operation, the authentication mode is configured as open authentication for all cases.

In the following, we report the performance of the *iMesh* architecture with the above setup running the OLSR protocol for routing. For comparison we have also ported Transparent Mobile IP [116] or TMIP to our testbed. Note that while we promote a "flat" routing protocol like OLSR for performance reason, the *iMesh* architecture can easily support other layer-3 schemes such as TMIP. In the following subsections, we report four different types of experiments: handoff latency, round-trip time (RTT) and impact of mobility over constant-bit-rate UDP traffic and TCP traffic.

5.4.2 Measuring Handoff Latency

In this subsection, we report measurements of handoff latency for a mobile station (STA) as it switches its association from one AP (oldAP) to another AP (newAP). Handoff is complete when STA re-establishes connectivity to the network via newAP. Handoff begins when the STA loses its association with oldAP. This is indicated by either the receipt of a deauthentication frame at oldAP from STA, or repeated failure of packet transmissions from oldAP to STA, or the transmission of probe request frame from STA. Since handoffs are forced in our experiments via a script on the client side we are able to determine the exact start of handoff by noting the timestamp of the deauthentication frame from the client to the oldAP. Layer-2 handoff ends when the client is able to associate with the newAP. At this time the layer-3 handoff starts.

In our *iMesh* architecture with the OLSR protocol, layer-3 handoff starts with the advertisement of a new HNA route to STA by the newAP. The OLSR protocol handles the broadcast of this message in the mesh network. Layer-3 handoff completes when the routes at all APs have been updated to reflect the newAP as the new point of attachment for STA. The handoff delay here depends on the number of route changes and the distance of these changes from the newAP. We have used

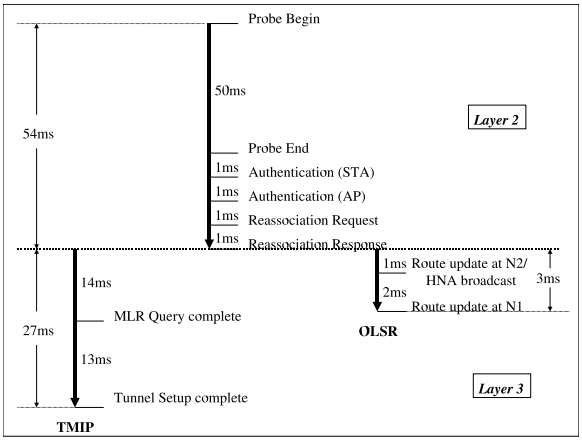


Figure 5.4: Timeline describing a typical fast one-hop handoff from $N1$ to $N2$ for a mobile station with *iMesh* running OLSR and TMIP. The probing delay during layer-2 handoff is optimized by scanning only a single channel. Layer-3 handoff for OLSR is around 3ms while TMIP incurs an average of 27ms. Note that the timeline is not to scale.

a linear topology of APs (Figure 5.3) to experiment with various distances (in number of hops) of these route changes while keeping the size of the testbed reasonably small.

In case of TMIP, the "foreign" AP (i.e., newAP) for the STA first determines the IP address of "home" AP (i.e., oldAP) by querying the central server (which is the AP $N3$ in Figure 5.3) that implements the *Mobile Location Register* or MLR. It then notifies the home AP so that the home AP can tunnel packets to the foreign AP. Layer-3 handoff completes when this procedure is completed.

The RF monitoring interface on the STA collects a timestamped trace of all exchanges in the wireless channel including the management frames. The trace is later analyzed to determine actual handoff delays in the layer-2 and layer-3. Figure 5.4 depicts the handoff timeline with a representative set of timing measurements. Note that the layer-2 handoff delay is dominated by the probe delay as explained before and examined critically in recent literature [71, 106, 107]. The timeline shows about 50ms delay for probing. A single channel is probed. In case the methods used in [106] are applied in our testbed, only one channel will be probed. Recall that the entire network operates on the same channel in this set of experiments. If multiple channels need to be probed, the 50ms probe time will be multiplied by the number of channels to be probed, and the handoff delay will increase accordingly. The authentication and reassociation activities are very fast and are done in a fraction of time relative to the probe delay. The layer-2 handoff with a single channel probe completes in about 54ms.

The timeline also shows layer-3 handoff delay for the one hop route change

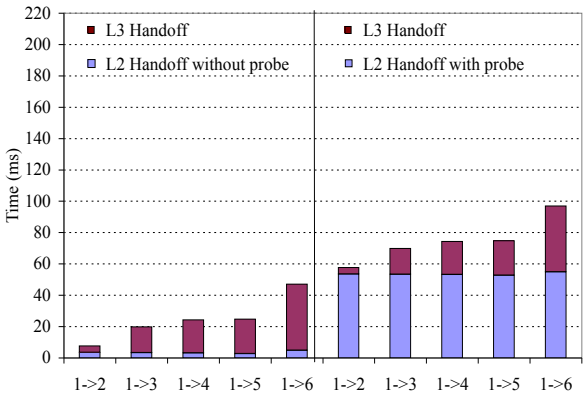


Figure 5.5: Handoff latencies for *iMesh* using OLSR routing without probing and with probing on a single channel. The notation $i \rightarrow j$ indicates hand-off from Ni directly to Nj.

case. This happens, for example, when STA moves from $N1$ to $N2$. In this case the handoff completes when the route is updated at $N1$, as the routes do not change in any other AP. This takes just 3ms. On the other hand, for the case of TMIP, an order of magnitude longer time is taken to complete the MLR query and the notification process with the home AP so that packets can be tunneled.

We now present a set of handoff latency measurements for handoffs for different hop lengths of route changes. For these studies, $N1$ is always the oldAP. The newAP is one of the five remaining nodes thus making up to five hop route changes. We think that our experiments are quite comprehensive, as in a deployed mesh network, route changes are unlikely to be any more than a few hops, if we assume that there are no coverage holes. This is because typically the distance between the newAP and oldAP will be less than twice the radio range. This makes the shortest path length in hops between these two APs very small. While route updates may be transmitted network-wide in OLSR, actual route changes will be limited to within a small neighborhood of newAP, because of the stated closeness of newAP and oldAP,

Two sets of handoff results are presented in Figures 5.5 and 5.6. The results for both architectures are presented without probing and with single channel probing. Different wireless device drivers were used to implement these schemes. To implement the no probing case – that provides the fastest layer-2 handoff – the HostAP driver is used in client mode with probing disabled. Thus, the layer-2 handoff in this case involves only authentication and association frame exchanges. To implement the single channel probing case, a modified airjack driver [22] is used. The *MinChannelTime* and *MaxChannelTime* intervals for the *probe-wait time* timer is set to 20ms and 30ms respectively.

Note that while these techniques aggressively reduce layer-2 handoff delay, they

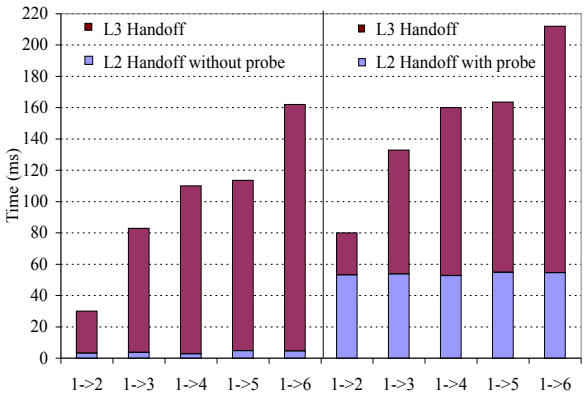


Figure 5.6: Handoff latencies for TMIP without probing and with probing on a single channel. The notation $i \rightarrow j$ indicates handoff from Ni directly to Nj.

are not unreasonable. Research in [106, 107] has shown how probing can be limited using prior knowledge. Such techniques will use only a single channel probe in our testbed. No probing can be reasonable in novel systems where beacons from APs can be used to learn the neighborhood of different APs and then to handoff to another AP in the same channel. Regardless of how layer-2 handoff is designed, our emphasis indeed is on layer-3 handoff because of the requirement of client-side transparency and the fact that handoff in 802.11 is controlled by the client in the most part.

Looking at Figures 5.5 and 5.6 layer-2 handoff delays are independent of amount of route changes as expected. Layer-3 delays on the other hand is proportional to the number of nodes along the path between newAP and oldAP in the case of *iMesh* with OLSR. In case of TMIP, latencies are much higher as the messages communicated between newAP and MLR and between newAP and oldAP have to travel over longer paths. Note that the absolute values of handoff delays are excellent when *iMesh* is used with OLSR. The maximum layer-3 latency is noted for a five hop long route change, and that is around 40ms (with less than 100ms total handoff latency), while one hop route change is accomplished within about 3ms (with less than 60ms total handoff latency). When probing is used, layer-3 handoff is actually faster than the layer-2 handoff, indicating that layer-2 handoff optimizations are more important. Note that these handoff latencies are comparable or better than observed on recent handoff studies on wireless LANs [106, 107, 104].

5.4.3 Round Trip Time Experiments

We now turn our attention to analyzing the impact of shortest path routing in OLSR vs. use of triangular routing in TMIP by measuring round-trip times. In this and the remaining experiments in the following section a “walk” of the mobile station STA is simulated. Initially, STA is associated with $N1$. At 10 seconds

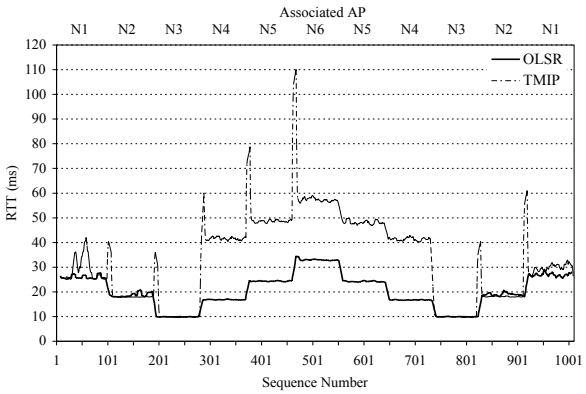


Figure 5.7: RTT Measurements for pings for *iMesh* with OLSR and TMIP with handoffs at intervals of 10 sec.

intervals it changes its association to $N2$ through to $N6$, and then retraces its "path" back to $N1$ in the same fashion, changing associations at 10 seconds intervals. The total walk duration is 110 seconds and it involves 11 handoffs.

During the walk, STA continuously pings an external host connected to the gateway $N3$ at 100ms intervals using 1500 byte ICMP packets and measure the RTT values for each ping. Figure 5.7 depicts the RTT values for TMIP and OLSR-based routing for different packet sequence numbers. For the TMIP case, $N1$ is the home AP for STA and the MLR is running on $N3$. The path length between STA and external host is optimal for iMesh at every instant since shortest path routes are used. With TMIP, when the "foreign" AP is either $N2$ or $N3$, the path lengths are the same as that of OLSR and hence both schemes have similar RTT values. The reason for triangular routes not being used in these cases is that the packets from external host to STA encounter the foreign AP along the path to the home AP and are directly forwarded tho the STA instead of being sent to home AP. When foreign AP is one of $N4$, $N5$ or $N6$, outgoing ping request packets from STA choose the shortest route to external host while the ping response packets first reach $N1$ and then are tunneled back to the foreign AP and thus have to travel six more hops than the OLSR case. The spikes at the beginning of each handoff for TMIP occurs when a tunnel to an old foreign AP is deleted and a tunnel to new foreign AP is created. All in-flight ping responses that reach an old foreign AP are forwarded back to the home AP and then tunneled to the new foreign AP.

5.4.4 Inter-arrival Measurements for CBR Traffic

We measure the inter-arrival times experienced at the mobile station for a 500Kbps CBR (constant-bit-rate) UDP packets sent from the external host. Figures 5.8 and Figure 5.9 plot the inter-arrival times for each packet received at the station. The same "walk" described in the previous subsection is also used here. Note substantially reduced variations in the times for OLSR routing. Figure 5.10

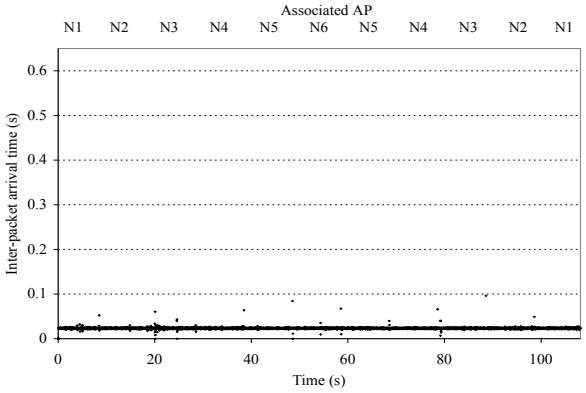


Figure 5.8: Inter-arrival times for 500Kbps CBR UDP traffic for *iMesh* with OLSR with handoffs at intervals of 10 sec.

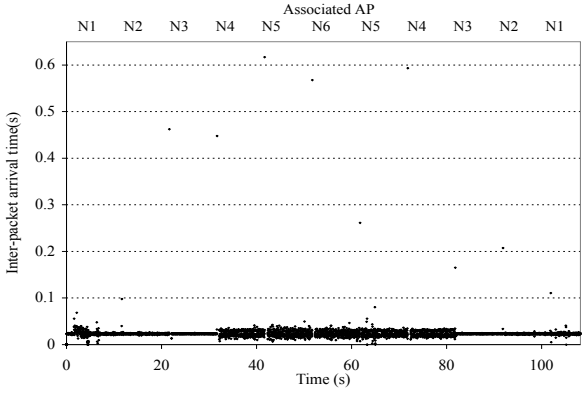


Figure 5.9: Inter-arrival times for 500Kbps CBR UDP traffic for TMIP with handoffs at intervals of 10 sec.

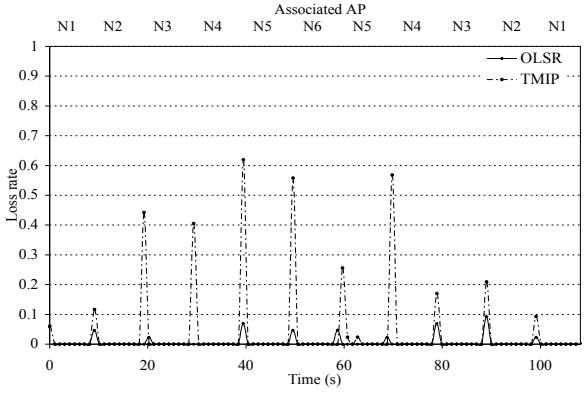


Figure 5.10: Average loss rates for 500Kbps CBR UDP traffic for *iMesh* with OLSR and TMIP with handoffs at intervals of 10 sec.

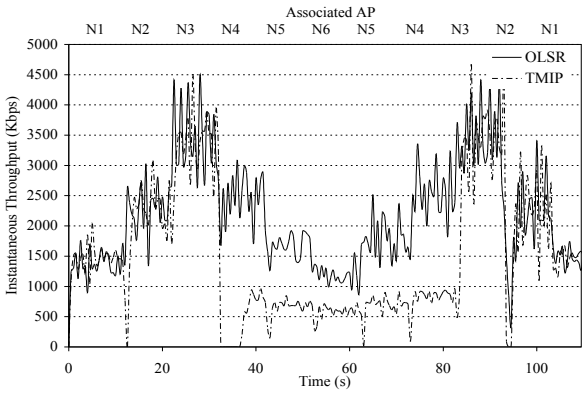


Figure 5.11: Instantaneous TCP throughput for OLSR and TMIP.

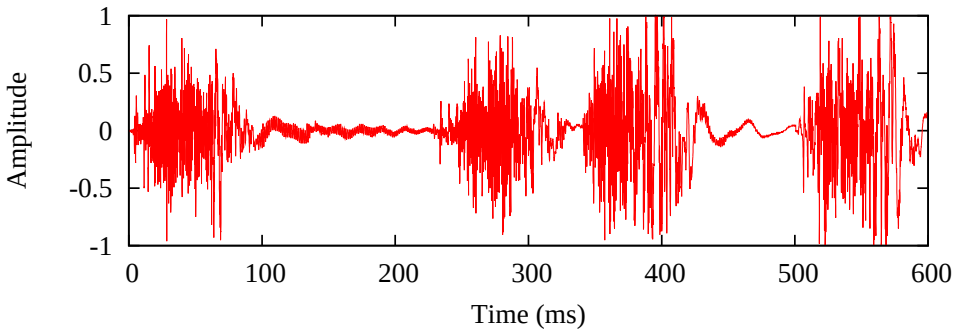
plots the average loss rates of 1 second intervals for both OLSR and TMIP. Note significantly higher losses for TMIP at the instants of handoff.

5.4.5 TCP Throughput Experiments

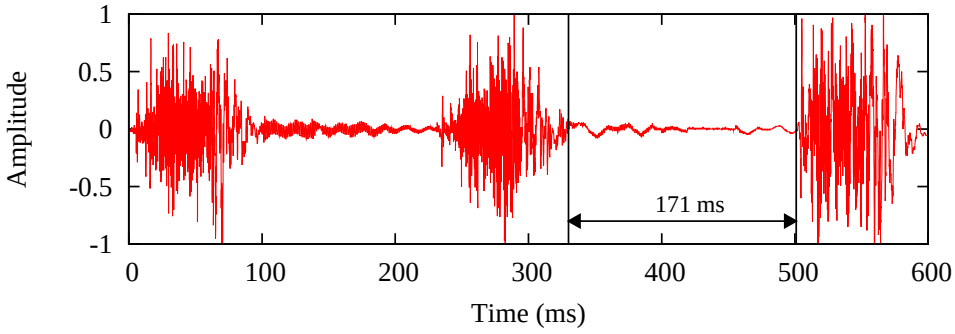
We also study the impact of frequent handoffs on the performance of a long-lived TCP connection. The same "walk" is used again. The mobile station starts a TCP transfer from the external host at the start of the walk. Figure 5.11 depicts the instantaneous throughput at the mobile node during the walk for both OLSR and TMIP. Note that in TMIP, the instantaneous throughput sometimes drops to zero, likely due to larger handoff latency. TMIP's throughput also suffers significantly in the situations where its path length is larger than OLSR's path length. (during the middle of the walk). Overall, our measurement shows that OLSR achieves around 42% improvement in the aggregate TCP throughput over TMIP during the entire transfer.

5.4.6 VoIP Experiments

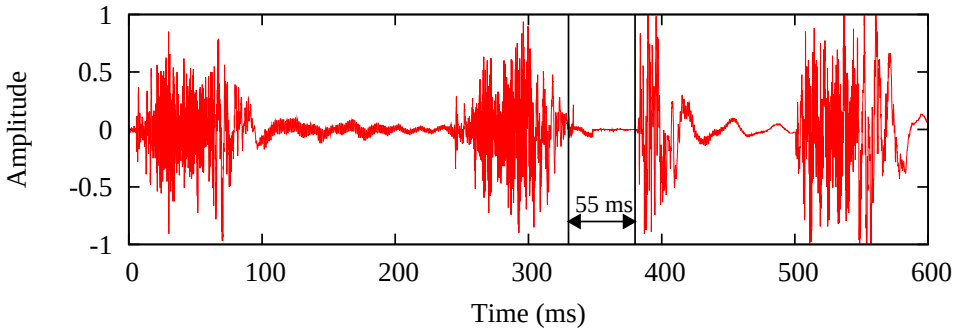
We analyze the perceived voice call quality during handoff with Skype, a popular peer-to-peer VoIP client [10]. We setup a Skype VoIP call between an external host connected to the wired network and a mobile STA. Similar to the previous experiments, the STA is forced to associate with different APs to simulate a roaming client. We modified the Skype settings at the external host so that the microphone input to Skype is fed via another application that can play wave audio file, while at the STA, the audio output of Skype is recorded by redirecting it to a file. For the following experiment, we use the same audio file as input to Skype and compare the recorded audio output at STA during three different scenarios: with no handoff, with two hop handoff using TMIP and with two hop handoff using OLSR. Figure 5.12 depicts a 600ms trace of sound signal values recorded during the three



(a) Received audio signal with no handoff.



(b) Received audio signal during a handoff with TMIP.



(c) Received audio signal during a handoff with OLSR.

Figure 5.12: Skype application audio signal plots recorded at the mobile client. A two hop handoff is initiated at time 330ms.

different scenarios. Though both handoff schemes incur a call disruption during handoff, OLSR performs significantly better with around 55ms of silence period.

5.5 Related Work

There are various industry initiatives to provide mesh networking support either from a service provider or from an equipment manufacturer point of view. However, little public information is available about the architectural choices and performance. The information we could gather from the white papers are quite limited. For example, Firetide [40] provides a complete routing solution on a mesh routing backbone using the TBRPF routing protocol [84] used for mobile ad hoc network routing. However, here the routers do not serve as APs. APs must be "connected" to the routers using a separate wired ethernet link. Vivato [119] uses two radios on the APs, but only one of them is used in infrastructure mode; the other is used to bridge to similar other radios on other APs. The goal is to fill coverage "holes" in a WLAN deployment. Tropos [117] and Strix Systems [113] appear to provide a similar architecture as we report here; however details are unclear. In our knowledge there is no reported performance evaluations about hand-off performance.

There have been several community initiatives for mesh networking. Examples include Locust World [63] (uses transparent mobile IP [116]), Bay Area Research Wireless Network [23] (routing solution unknown), Seattle Wireless Network [102] (can use Internet routing protocols such as OSPF or RIP), and Champaign-Urbana Community Wireless Network [30] (uses *Hazy Sighted Link State* routing protocol [100]). However, none of such initiatives report any performance data or experience report.

On the other hand, there is some documentation on building and using mesh networking concepts in the research community. The Roofnet project at MIT [72] and mesh networking research in Microsoft Research [81] are the most prominent examples of work in this area. Roofnet is a collection of wireless routers that are stationary PCs running Linux with 802.11-based interfaces in the ad hoc mode. The network runs a routing protocol, SrcRR [19], that is based on the well known Dynamic Source Routing (DSR) protocol [50]. Mesh networking research in Microsoft Research uses a testbed similar to Roofnet with 802.11 interfaces running in ad hoc mode. They addressed the question of appropriate routing metrics [37] and use of multiple radios on each node [38]. Recently, Raniwala and Chiueh considered routing and channel assignment algorithms for multi-radio mesh networks with a similar testbed [95, 94]. However, these projects only consider the backbone mesh network, and do not directly provide any support for seamless mobility for mobile stations that is the main focus in our current work.

The wireless LAN research community has addressed the question of fast layer-2 handoffs in wireless LANs. The issue here is to save channel probing times to determine the best AP to hand off to. Shin, Mishra and Arbaugh have proposed

schemes to reduce probing latency by scanning only a subset of channels. The probe wait time is optimized by exploiting the knowledge of operating channels of APs in the neighborhood [106]. Shin, Rawat and Schulzrinne have proposed a new handoff procedure which reduces the probe time by using a selective scanning algorithm and a caching mechanism [107]. Cleyn et. al. [34] and Sharma et. al. [104] independently have considered efficient layer-3 hand-offs for wireless LANs, but only in the context of standard-compliant mobile IP [89]. Our layer-3 hand-off latencies are much superior relative to these reported studies. This is in spite of the fact that our APs are connected via a wireless network while wired links are considered in these studies. However, it is fair to state that conformance to standard mobile IP imparts some inefficiency to these implementations. This also happens with transparent mobile IP in our work.

Recently, the IEEE 802.11 committee has started a new working group (802.11s) for the so-called *ESS mesh* networking [1]. The purpose of this new 802.11s ESS mesh working group is to provide a protocol for auto-configuring paths between APs over self-configuring multi-hop topologies to support both broadcast/multicast and unicast traffic. This solution also uses WDS links.

5.6 Conclusions

We have presented an 802.11-based "infrastructure-mode" mesh networking architecture called *iMesh*. The goal of our design is client-side transparency, so that existing mobile clients can seamlessly use such a mesh network *in lieu* of a wireless LAN. The fundamental design concept is the use of a flat routing protocol in the mesh network where association of a mobile station at an access point triggers a route update. This ensures that the optimal path to the mobile is maintained at all times. We analyzed the performance of the *iMesh* architecture in a six node 802.11b-based mesh network. We presented detailed experimental results involving measurements of handoff latencies at both layer-2 and layer-3. The flat routing demonstrates excellent latency performance relative to a more traditional layer-3 handoff technique using a mobile IP like scheme, called transparent mobile IP. The layer-3 latency for the routing scheme is faster by a factor of about 3–5. If absolute performance is of concern, the routing scheme provides a combined layer-2 and layer-3 handoff latency of less than 50–100ms (depending on the layer-2 technique used) even when the route change involves route updates over five hops. This measurement includes certain layer-2 optimizations reported in literature. We have also showed experimentally that the improvement in handoff latency over transparent mobile IP translates to a superior transport protocol (UDP and TCP) performance in terms of losses, delay and throughput.

Chapter 6

Conclusion

In this dissertation work, we critically analyzed the impact of wireless interference and client mobility in WLANs and mesh networks. We designed mechanisms to mitigate the effects of wireless interference in three different scenarios. With regard to the mobility issue, we designed an architecture for supporting seamless client mobility.

In our first work, we address the issue transient degradation in wireless link quality, which is not handled by existing end-to-end routing protocols for mesh networks. We developed Deflect, a path adaptation mechanism that complements the routing protocol to achieve quick detection of link losses and rerouting of traffic via an alternative node in the neighborhood of the faulty link using only one-hop signaling messages. Performance evaluation on the testbed shows that Deflect can estimate the presence of interference within 200 ms and adpat the route immediately with very low overhead.

In our second work, we explored the use of fine grained transmit power control to reduce co-channel interference and increase the network capacity for dense WLAN deployments such as campus-wide or city-wide deployments. We developed a novel centralized transmit power control scheme called Contour for the APs in the network, which significantly reduces inter-AP interference while avoiding starvation due to use of asymmetric transmit power levels. Our results obtained from our WLAN testbed shows that Contour improves the network throughput by about 40%.

Our third work investigated the impact of extreme forms of interference caused by a malicious user who is jamming all communication on a channel. We proposed an optimal channel hopping protocol that improves the security and resilience of 802.11 networks to jamming attacks. We show through analysis and prototype experiments on existing conventional 802.11 hardware that it is possible to achieve up to 60% of the achievable throughout in presence of an intelligent jammer who is aware of the use of a channel hopping protocol.

Finally, we analysed the mobility issue from the infrastructure side of the network without the need to modify the software on the clients. When APs serving clients are connected over a mesh networks, the routing protocol quickly reroutes the client traffic to the new point of attachment. We developed a mobility management architecture that tracks the location of clients to achieve very fast network layer handoffs in the order of few milliseconds.

Bibliography

[1] Amendment to the current 802.11 standard to provide Extended Service Set Mesh Networking. http://standards.ieee.org/board/nes/projects/802-11s.pdf.

[2] Atheros communications. `http://www.atheros.com`.

[3] CRAWDAD data on campus wireless deployments. `http://crawdad.cs.dartmouth.edu/data.php`.

[4] Google Wi-fi Coverage. `http://wifi.google.com/city/mv/apmap.html`.

[5] hrtimers - High-resloution timer subsystem. `http://www.tglx.de/hrtimers.html`.

[6] National Marine Electronics Association. `http://www.nmea.org/`.

[7] The netfilter/iptables project: the Linux 2.4.x/2.5.x firewalling subsystem. http://www.netfilter.org.

[8] Recommended Practice for Multi-Vendor Access Point Interoperability via an Inter-Access Point Protocol Across Distribution Systems Supporting IEEE 802.11 Operation. http://csl.ee.iastate.edu/ cpre543/80211f.pdf.

[9] SHM PPS driver. `http://time.qnan.org/shmpps.tar`.

[10] *Skype*. http://www.skype.net.

[11] T-Mobile Hotspots. `http://hotspot.t-mobile.com/`.

[12] Using a Garmin GPS 18 LVC as NTP stratum-0 on Linux 2.6. `http://time.qnan.org/`.

[13] IEEE 802.11b/d3.0 Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification, August.

[14] IEEE Standards Department, Draft Standard IEEE 802.11, Wireless LANs (P802.11D3), 1996.

[15] P. S. A. Akella, G. Judd and S. Seshan. Self management in chaotic wireless deployments. In *ACM MobiCom 2005*.

[16] S. B. A. Mishra, V. Shrivastava and W. Arbaugh. Partially overlapped channels not considered harmful. In *SIGMETRICS '06*.

[17] F. B. Abdesslem, L. Iannone, M. D. de Amorim, K. Kabassanov, and S. Fdida. On the feasibility of power control in current ieee 802.11 devices. In *PERCOMW '06*.

[18] A. Adya, P. Bahl, J. Padhye, A. Wolman, and L. Zhou. A multi-radio unification protocol for IEEE 802.11 wireless networks. In *Proc. of Broadnets*, 2004.

[19] D. Aguayo, J. Bicket, and R. Morris. SrcRR: A high throughput routing protocol for 802.11 mesh networks. Technical Report, 2004.

[20] A. Akella, G. Judd, P. Steenkiste, and S. Seshan. Self management in chaotic wireless deployments. In *Proc. of ACM Mobicom*, 2005.

[21] M. Aron and P. Druschel. Soft timers: efficient microsecond software timer support for network processing. *ACM Trans. Comput. Syst.*, 18(3):197–228, 2000.

[22] R. Baird and M. Lynn. Airjack Driver, Version 0.6.2-alpha. http://sourceforge.net/projects/airjack.

[23] Bay Area Research Wireless Network. http://www.barwn.org.

[24] Y. Bejerano and S. Han. Cell breathing techniques for load balancing in wireless lans. In *INFOCOM '06*.

[25] J. Bicket, D. Aguayo, S. Biswas, and R. Morris. Architecture and evaluation of an unplanned 802.11b mesh network. In *MobiCom '05*.

[26] S. Biswas and R. Morris. Opportunistic routing in multi-hop wireless networks. In *Proc. SIGCOMM*, 2005.

[27] M. Burkhart, P. von Rickenbach, R. Wattenhofer, and A. Zollinger. Does Topology Control Reduce Interference? In *MOBIHOC*, 2004.

[28] J. Byers, M. Luby, M. Mitzenmacher, and A. Rege. A Digital Fountain Approach to Reliable Distribution of Bulk Data. In *Proceedings of SIGCOMM'98*, Vancouver, Canada, September 1998.

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

[30] Champaign-Urbana Community Wireless Network. http://www.cuwireless.net.

[31] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris. Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. In *Proceedings of Intl. Conference on MobiCom*, 2001.

[32] T. Clausen and P. Jacquet. Optimized link state routing protocol (OLSR). RFC 3626, October 2003.

[33] R. Cle and J. Rosenbluth. Voice over ip performance monitoring. In *ACM Computer Communication Review*.

[34] P. D. Cleyn, N. V. den Wijngaert, L. Cerda, and C. Blondia. A smooth handoff scheme using IEEE802.11 triggers: design and implementation. *Computer Networks*, 45(3):345–361, 2004.

[35] D. S. J. D. Couto, D. Aguayo, J. Bicket, and R. Morris. A high-throughput path metric for multi-hop wireless routing. In *Proceedings of the 9th ACM International Conference on Mobile Computing and Networking (MobiCom 03)*, September 2003.

[36] S. Desilva and S. R. Das. Experimental evaluation of a wirelessad hoc network. In *Proceedings of the 9th. IEEEInternational Conference on Computer Communications andNetworks (IC3N)*, pages 528–534, Oct. 2000.

[37] R. Draves, J. Padhye, and B. Zill. Comparison of routing metrics for static multi-hop wireless networks. *SIGCOMM Comput. Commun. Rev.*, 34(4):133–144, 2004.

[38] R. Draves, J. Padhye, and B. Zill. Routing in multi-radio, multi-hop wireless mesh networks. In *Proc. of ACM MOBICOM*, pages 114–128. ACM Press, 2004.

[39] J. Eriksson, S. Agarwal, P. Bahl, and J. Padhye. Feasibility study of mesh networks for all-wireless offices. In *MobiSys 2006*.

[40] Firetide. Wireless instant networks. http://www.firetide.com.

[41] M. Garetto, J. Shi, and E. W. Knightly. Modeling media access in embedded two-flow topologies of multi-hop wireless networks. In *ACM MobiCom 2005*.

[42] Garmin. GPS-18 Technical Specifications. `http://www.garmin.com/manuals/425_TechnicalSpecification.pdf`.

[43] http://www.gnu.org/software/gnuradio.

[44] R. S. Gray, D. Kotz, C. Newport, NikitaDubrovsky, A. Fiske, J. Liu, C. Masone, SusanMcGrath, and Y. Yuan. Outdoor experimental comparison of four ad hoc routing algorithms. In *Proceedings of the ACM/IEEE International Symposium on Modeling,Analysis and Simulation of Wireless and Mobile Systems (MSWiM)*, pages 220–229, October 2004.

[45] C. Gui and P. Mohapatra. Short: self-healing and optimizing routing techniques for mobile ad hoc networks. In *MobiHoc '03: Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*, pages 279–290, 2003.

[46] N. Gupta and S. R. Das. Energy-aware on-demand routing for mobile adhoc networks. In *Springer Lecture Notesin Computer Science: LNCS2571: Proceedings of the 4th InternationalWorkshop on DistributedComputing (IWDC 2002)*, pages 164–173, Kolkata, India, Dec 2002.

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

[48] S. Jain and S. R. Das. Exploiting path diversity in the link layer inwireless ad hoc networks. In *Proc. of IEEE WoWMoM Symposium*, June 2005.

[49] Z. Ji, Y. Yang, J. Zhou, M. Takai, and R. Bagrodia. Exploiting medium access diversity in rate adaptive wireless lans. In *Proc. ACM MobiCom Conference*, 2004.

[50] D. Johnson and D. Maltz. Dynamic source routing in ad hoc wireless networks. In T. Imielinski and H. Korth, editors, *Mobile computing*, chapter 5. Kluwer Academic, 1996.

[51] D. B. Johnson, D. A. Maltz, and J. Broch. DSR: The dynamic source routing protocol for multi-hop wireless ad hoc networks. In C. E. Perkins, editor, *Ad Hoc Networking*, chapter 5, pages 139–172. Addison-Wesley, 2001.

[52] E.-S. Jung and N. Vaidya. A power control mac protocol for ad hoc networks. In *ACM MobiCom*, 2002.

[53] V. Kawadia and P. R. Kumar. Principles and protocols for power control in ad hoc networks. *IEEE JSAC*.

[54] V. Kawadia and P. R. Kumar. Power control and clustering in ad hoc networks. In *INFOCOM*, 2003.

[55] T.-S. Kim, H. Lim, and J. C. Hou. Improving spatial reuse through tuning transmit power, carrier sense threshold, and data rate in multihop wireless networks. In *Proc. ACM MobiCom Conference*, Sept. 2006.

[56] A. Kochut, A. Vasan, A. Shankar, and A. Agrawala. Sniffing out the correct physical layer capture model in 802.11b, 2004.

[57] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. The click modular router. *ACM Transactions on Computer Systems*, 18(3):263–297, August 2000.

[58] R. R. Kompella, S. Ramabhadran, I. Ramani, and A. C. Snoeren. Cooperative packet scheduling via pipelining in 802.11 wireless networks. In *E-WIND 2005*, pages 35–40, New York, NY, USA, 2005. ACM Press.

[59] R. R. Kompella and A. C. Snoeren. Practical lazy scheduling in sensor networks. In *SenSys '03*.

[60] M. Krunz, A. Muqattash, and S.-J. Lee. Transmission power control in wireless ad hoc networks: Challenges, solutions, and open issues. *IEEE Network*, 18(5):8–14, Sept/Oct 2004.

[61] P. Kyasanur and N. H. Vaidya. Selfish mac layer misbehavior in wireless networks. *IEEE Transactions on Mobile Computing*, 4(5), September/October 2005.

[62] P. Larsson. Selection diversity forwarding in a multihop packet radio network with fading channel and capture. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5:79–282, October 2001.

[63] Locust World. http://www.locustworld.com.

[64] MADWiFi. Multiband ATHEROS driver for WiFi. http://www.madwifi.org.

[65] J. Malinen. Host AP driver for Intersil Prism2/2.5/3. http://hostap.epitest.fi.

[66] D. Maltz, J. Broch, J. Jetcheva, and D. Johnson. The effects of on-demand behavior in routing protocols for multi-hop wireless ad hoc networks. *IEEE Journal on Selected Areas in Communication*, 17(8), August 1999.

[67] D. Maltz, J. Broch, and D. Johnson. Lessons from a full-scale multihop wireless ad hoc networktestbed. *IEEE Personal Communications Magazine*, 8(1):8–15, Feb. 2001.

[68] M. Marina and S. R. Das. On demand multipath distance vectorrouting in ad hoc networks. In *Proceedingsof the International Conference on Network Protocols (ICNP)*, Dec. 2001.

[69] MeshNetworks. http://www.meshnetworks.com.

[70] Microsoft Corp. Native 802.11 framework for IEEE 802.11 networks. http://www.microsoft.com.

[71] A. Mishra, M. Shin, and W. Arbaugh. An empirical analysis of the IEEE 802.11 MAC layer handoff process. *SIGCOMM Comput. Commun. Rev.*, 33(2):93–102, 2003.

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

[73] A. K. Miu, H. Balakrishnan, and C. E. Koksal. Improving Loss Resilience with Multi-Radio Diversity in Wireless Networks. In *11th ACM MOBICOM Conference*, Cologne, September 2005.

[74] A. K. Miu, G. Tan, H. Balakrishnan, and J. Apostolopoulos. Divert: Fine-grained Path Selection for Wireless LANs. In *2nd International Conference on Mobile Systems, Applications and Services (Mobisys 2004)*, Boston, June 2004.

[75] J. P. Monks, V. Bharghavan, and W. mei W. Hwu. Power controlled multiple access protocol for wireless packet networks. In *IEEE Infocom*, 2002.

[76] T. Moscibroda and R. Wattenhofer. Minimizing Interference in Ad Hoc and Sensor Networks. In *DIALM-POMC, Cologne, Germany*, 2005.

[77] A. Muqattash and M. Krunz. A distributed transmission power control protocol for mobile ad hoc networks. *IEEE Transactions on Mobile Computing*, 3(2):113–128, April/June 2004.

[78] A. Muqattash and M. Krunz. Powmac: A single-channel power-control protocol for throughput enhancement in wireless ad hoc networks. *IEEE Journal on Selected Areas in Communications*, 23(5), May 2005.

[79] S. Narayanaswamy, V. Kawadia, R. S. Sreenivas, and P. R. Kumar. Power control in ad hoc networks : Theory, architecture, algorithm and implementation of the COMPOW protocol. In *Proc. European Wireless Conference*, 2002.

[80] A. Nasipuri, R. Castaneda, and S. R. Das. Performance ofmultipath routing for on-demand protocols in ad hoc networks. *ACM/Kluwer Mobile Networks (MONET) Journal*, 6(4):339–349, 2001.

[81] Networking Research Group, Microsoft Research. http://research.microsoft.com/mesh.

[82] D. Niculescu, S. Ganguly, K. Kim, and R. Izmailov. Performance of VoIP in a 802.11b wireless mesh network. In *Proc. of the IEEE INFOCOM'06 Conf.*, 2006.

[83] http://csrc.nist.gov/CryptoToolkit/tkrng.html.

[84] R. Ogier, F. Templin, B. Bellur, and M. Lewis. Topology broadcast based on reverse-path forwarding (TBRPF). IETF Draft, October 2003. Work in progress.

[85] Packethop. http://www.packethop.com.

[86] M. Pearlman, Z. Haas, P. Scholander, and S. Tabrizi. On the impact of alternate path routing for load balancing in mobile ad hoc networks. In *Proceedings of ACM MobiHoc 2000 Workshop*, August 2000.

[87] Pebble Linux. http://www.nycwireless.net/pebble.

[88] C. Perkins, editor. *Ad Hoc Networking*. Addison Wesley, 2001.

[89] C. Perkins. IP mobility support, RFC 2002, October 1996.

[90] C. Perkins, E. Royer, and S. R. Das. Ad hoc on demand distance vector (AODV) routing. RFC 3561, July 2003.

[91] C. E. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In *Proceedings of the ACM SIGCOMM '94 Conference*, pages 234–244, August 1994.

[92] R. Ramanathan and R. Rosales-Hain. Topology control in multihop wireless networks using transmit power adjustment. In *Proceedings of the IEEE INFOCOM 2000 Conference*, 2000.

[93] I. Ranani, R. R. Kompella, S. Ramabhadran, and A. Snoren. Efficient cooperative scheduling in 802.11 wireless networks. In *UCSD Technical Report CS2005-0830*, 2005.

[94] A. Raniwala and T. Chiueh. Evaluation of a wireless enterprise backbone network architecture. In *Proc. of Hot Interconnects*, 2004.

[95] A. Raniwala, K. Gopalan, and T. Chiueh. Centralized channel assignment and routing algorithms for multi-channel wireless mesh networks. *SIGMOBILE Mob. Comput. Commun. Rev.*, 8(2):50–65, 2004.

[96] A. Rao and I. Stoica. An overlay mac layer for 802.11 networks. In *MobiSys 2005*.

[97] M. Raya, J. Hubaux, and I. Aad. Domino: A system to detect greedy behavior in ieee 802.11 hotspots. In *Proceedings of Mobisys*, Boston, MA, June 2004.

[98] Routerboard. Routerboard. http://www.routerboard.com.

[99] R. RoyChoudhury and N. Vaidya. Mac-layer anycasting in ad hoc networks. *Computer Communication Review*, 34(1), 2004.

[100] C. Santivanez and R. Ramanathan. Hazy sighted link state (HSLS) routing: A scalable link state algorithm. Technical report, BBN technical memo BBN-TM-I30I, BBN technologies, Cambridge, MA, 2001. Available at http://www.ir.bbn.com/documents/techmemos.

[101] A. Scaglione and Y. W. Hong. Opportunistic large arrays: Cooperative transmission in wireless multihop ad hoc networks to reach far distances. *IEEE Transactions on Signal Processing*, 51(8), 2003.

[102] Seattle Wireless. http://www.seattlewireless.net.

[103] A. Shamir. On the Generation of Cryptographically Strong Pseudorandom Sequences. *ACM Transactions on Computer Systems*, 1(1), February 1983.

[104] S. Sharma, N. Zhu, and T. cker Chiueh. Low-latency mobile ip handoff for infrastructure-mode wireless lans. *IEEE Journal of Selected Areas in Communications*, 2004.

[105] A. Sheth and R. Han. Shush : Reactive transmit power control for wireless mac protocols. In *WICON '05*.

[106] M. Shin, A. Mishra, and W. A. Arbaugh. Improving the latency of 802.11 hand-offs using neighbor graphs. In *Proc. of ACM MOBISYS*, pages 70–83. ACM Press, 2004.

[107] S. Shin, A. G. Forte, A. S. Rawat, and H. Schulzrinne. Reducing MAC layer handoff latency in IEEE 802.11 wireless lans. In *Proc. of ACM MOBIWAC*, pages 19–26. ACM Press, 2004.

[108] S. Singh and C. S. Raghavendra. PAMAS: Power aware multi-access protocol with signalling for ad hoc networks. *ACM Computer Communications Review*, July 1998.

[109] B. Sirkeci-Mergen and A. Scaglione. Continuum approach to dense wireless networks with cooperation. In *IEEE Infocom*, 2005.

[110] H.-S. W. So, G. Nguyen, and J. Walrand. Practical synchronization techniques for multi-channel mac. In *MobiCom '06*, 2006.

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

[112] M. Soroushnejad and E. Geraniotis. Probability of capture and rejection of primary multiple access interference in spread spectrum networks. *IEEE Transactions on Communications*, 39(6), 1991.

[113] Strix Systems. www.strixsystems.com.

[114] M. Subbarao. Dynamic power-conscious routing for manets: An initial approach. In *IEEE Vehicular Technology Conference '99*.

[115] F. A. Tobagi and L. Kleinrock. Packet switching in radio channels: Part II - the hidden terminal problem in carrier sense multiple-access modes and the busy-tone solution. *IEEE Transactions on Communications*, 23(12), 1975.

[116] Transparent Mobile IP.
http://www.slyware.com/projects_tmip_intro.shtml.

[117] Tropos Networks. http://www.tropos.com.

[118] U. Varshney. The status and future of 802.11-based WLANs. *IEEE Computer*, 36(6), June 2003.

[119] Vivato. http://www.vivato.net.

[120] J. Wang, H. Zhai, and Y. Fang. Opportunistic packet scheduling and media access control for wireless lans and multi-hop ad hoc networks. In *Proc. of IEEE Wireless Communications and Networking Conference (WCNC'04)*, 2004.

[121] K. Xu, M. Gerla, and S. Bae. How effective is the ieee 802.11 rts/cts handshake in ad hoc networks. In *IEEEE GLOBECOM*, 2002.

[122] W. Xu, W. Trappe, Y. Zhang, and T. Wood. The Feasibility of Launching and Detecting Jamming Attacks. In *6th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC)*, Urbana-Champaign, IL, 2005.

[123] W. Xu, T. Wood, W. Trappe, and Y. Zhang. Channel surfing and spatial retreats: defenses against wireless denial of service. In *Proceedings of the ACM Workshop on Wireless Security (WiSe'04)*, Philadelphia, PA, October 2004.

[124] Y. Xu, J. S. Heidemann, and D. Estri. Geography-informed energy conservation for ad hoc routing. In *Proceedings of the Intl. Conference on MobiCom*, 2001.

[125] X. Yang and N. Vaidya. Spatial backoff contention resolution for wireless networks. In *Proc. IEEE WiMesh Workshop*, 2006.

[126] M. Zorzi and R. R. Rao. Geographic random forwarding (geraf) for ad hoc and sensor networks: Multihop performance. *IEEE Trans. Mob. Comput.*, 2, 2003.