ABSTRACT

Title of Dissertation: SUPPORTING SECURE AND TRANSPARENT MOBILITY IN WIRELESS LOCAL-AREA NETWORKS

Arunesh Mishra, Doctor of Philosophy, 2005

Dissertation directed by: Assistant Professor William Arbaugh
Department of Computer Science

Wireless Local Area Networks (WLANs) are experiencing unprecedented growth as the last mile connectivity solution. Mobility is an important feature of any wireless communication system. Handoffs are a crucial link level functionality that enable a mobile user to stay connected to a wireless network by switching the data connection from one base station or access point to another. Conceptually the handoff process can be subdivided into two phases: (i) Discovery - wherein the client searches for APs in vicinity and (ii) Authentication - the client authenticates to an AP selected from the discovery phase.

The handoff procedure recommended by the IEEE 802.11 standard and closely implemented by various wireless vendors is an intrusive and a brute-force approach. My testbed based study of these algorithms showed that they incur high latencies varying between 400ms to 1.3 seconds depending on the security settings in effect. Such inefficient handoff mechanisms can have a detrimental impact on applications especially synchronous multimedia connections such as Voice over IP.

In my dissertation, I have proposed and evaluated the notion of locality among APs induced by user mobility patterns. A relation is created among APs which captures this locality in a graph theoretic manner called *neighbor graphs* – a distributed structure that autonomously captures such locality. Based on this, I have designed and evaluated efficient mechanisms to address the two different phases of this handoff process. Through a rigorous testbed based implementation, I have demonstrated the viability of the concept of mobility induced locality through good performance improvements. Through extensive simulations I have studied the performance of proposed handoff mechanisms over various different topologies. This work has shown that a topological structure which captures the locality relationship among APs is fundamental to designing mechanisms that make user mobility transparent from the higher layers of the networking stack.

SUPPORTING SECURE AND TRANSPARENT MOBILITY IN WIRELESS LOCAL-AREA NETWORKS

by

Arunesh Mishra

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2005

Advisory Committee:

Assistant Professor William Arbaugh, Chairman/Advisor
Professor A. Udaya Shankar
Associate Professor Bobby Bhattacharjee
Professor Raymond Miller
Professor Mark Shayman, Dean's Representative

DEDICATION

To my late Father who believed strongly in discipline, systematic approaches to work and a quest for perfection in everything you do. To my Mother who has provided constant encouragement during good and bad times and who has always supported my decisions.

ACKNOWLEDGEMENTS

My graduate study has lasted about $5\frac{1}{2}$ years at the University of Maryland. During this period a lot of people have helped me through with various aspects of this work. I would like to acknowledge the efforts of Mike van Opstal who was always available and helpful with the wireless testbed and related issues at any time, be it a night or a weekend. I would like to acknowledge the efforts of my colleague Min-ho Shin, who has helped extensively with the implementations, the simulations and staying up in the night for the experiments apart from fruitful discussions that we had. I wish him the best with his research. I have to mention the constant enthusiasm boosters from my friends notably Chirag Kathrani, Arunchandar Vasan, Srinivasan

Parthasarathy, Vijay Gopalakrishnan and Yuan Yuan who have always helped immensely during the tough periods of my graduate study. Without them it would have been very hard to finish this long journey. I would like to also mention Adithya Nagarajan who has been a good take-a-break-buddy. He is currently programming away at Microsoft.

I would like to acknowledge the constant help offered by Dr. Bobby Bhattacharjee on various aspects during my study. Most importantly I would like to acknowledge the hard work and efforts of my advisor Dr. Bill Arbaugh. Without his guidance and support this whole effort would have been impossible.

Finally, I would not have made it this far in my doctoral work if not for the constant affection, encouragement and support from a close friend of mine, Maya Palem.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

The radio music box has no imaginable commercial value. Who would pay for a message sent to nobody in particular? – David Sarnoff's associates in response to his urgings for investment in the radio in the 1920s.

Wireless communications is enjoying its fastest growth period of the last 100 years, due to the enabling technologies of today. In 1897, Guglielmo Marconi first demonstrated the radio's ability to communicate wirelessly among ships sailing in the English channel. Since then growth in the mobile communications field was slow but closely coupled with technological advancements. The notion of providing mass wireless communications was not conceived until Bell Laboratories developed the cellular concept in the 1960s and 70s [1]. The development of the miniature solid-state radio frequency hardware in the 1970s has

matured as a technology today bringing about an explosive growth in the form of data and multimedia services in the wireless communications era.

Mobility is an important aspect of a wireless communication system. The early mobile radio telephony systems (first introduced in 1946) used a single powerful transceiver called a *base-station* to cover large distances (30 miles). Thus, the mobility of users was transparent within the large region of coverage. This approach became inefficient in crowded areas such as the New York City market. With such coverage, the available 12 independent channels could serve only 543 users [2]. Bell Labs developed the theory and techniques of cellular radiotelephony – the concept of breaking a coverage zone into small cells, each of which reuse portions of the spectrum to increase spectrum usage at the expense of greater system infrastructure. Some of today's cellular architectures use a micro-cell concept where the cell sizes are greatly reduced to increase frequency reuse. This also enables the user radios to be power efficient – an important consideration for today's battery powered phones. This cellular approach, however, inevitably exposes the user to the effects of mobility limited by the granularity of the cell sizes. Such a design brought about the need for low level primitives which provide for a smooth transition of the phone call across the

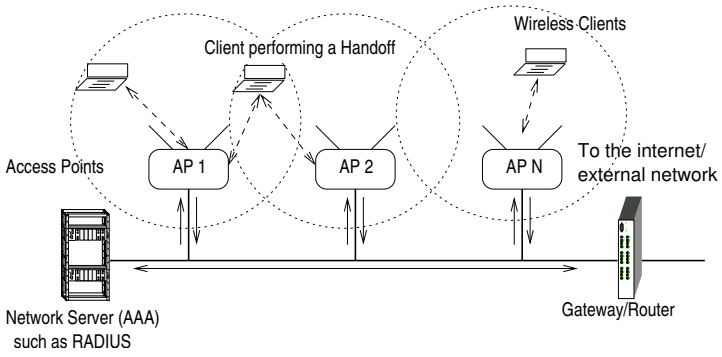


Figure 1.1: Architecture of a typical IEEE 802.11 wireless LAN.

cell boundaries. This is when the first *handover or handoff* primitives were designed in the context of wireless networks to provide for the transition of a phone call across cell boundaries.

Wireless Local-Area Networks (WLANs)

In the late 1980s, the FCC allocated license free bands for low power spread spectrum devices in the 900 MHz, 2.4. Ghz and the 5.7 Ghz bands to facilitate private computer communication in the workplace and the household. The IEEE 802.11 Wireless LAN working group was founded in 1987. 802.11 was finally standardized by 1997 gaining impetus from the popularity of the Internet and portable laptop computers. The original IEEE 802.11 standard uses direct sequence spread spectrum to provide data-rates of up to 2 Mbps in the 2.4 Ghz band. The higher rate extension, called IEEE 802.11b, provides data-rates

of up-to 11 Mbps in the same spectrum. During the last few years, the IEEE Wireless LAN working group has standardized the 802.11a and 802.11g technologies which use orthogonal frequency division multiplexing (OFDM) to provide data-rates up-to 54 Mbps in the 5 Ghz and the 2.4 Ghz bands respectively.

The basic communication model used in an IEEE 802.11 wireless LAN is very similar to that of a cellular network. Designated devices, called *access points* (APs) perform the function of a base-station in the cellular counterpart. Each individual user obtains network service from a single access point and is said to be *associated* to it. The user communicates all data traffic through its associated AP. An access point together with its associated clients form what is called a *basic service set* (BSS), much like a base-station forms a cell in a cellular network. Multiple APs are used to extend coverage beyond a single BSS. A set of two or more APs which collectively form one logical wireless network is called an *extended service set* (ESS). An extended service set is associated with an identifier (typically a 8-10 character string) which is used by clients to search and associate to APs belonging to the correct wireless LAN. Such a mechanism allows co-existence of different extended service sets in the same physical space.

Typically, an access point *bridges* data traffic over a wired ether-

net interface. Thus, installing a wireless network requires proper positioning of APs along with the ethernet wiring that forms the *backbone* of the wireless network (DS). Figure 1.1 shows the backbone architecture typically used to engineer a wireless LAN. The backbone network connects the APs to a server which performs various management and accounting functions for the network as a whole. The backbone also connects to a gateway/router which connects to an external network such as the Internet.

Since multiple APs are used to extend the coverage of a single BSS, a mobile user's data session might suffer disconnections as it moves out of the coverage of one AP into the coverage of another. To facilitate such handoffs, the IEEE 802.11 standard defines certain MAC level functions by which the user can maintain higher layer connectivity while the link layer switches the APs.

In contrast with wireless LANs, cellular networks have very regular structures. Each cell has a fixed number of neighbors and regular coverage contours so a regular hexagon is used to approximate a base-station's cell. Wireless LANs have seen popular use in the indoor environment. Due to the complex nature of radio frequency behavior in such environments, it becomes difficult to predict coverage and overlap among APs. Also a single wireless LAN can be composed of AP

devices from different vendors. This makes them loosely coupled in nature and in fact the former standardized protocol used for inter-AP communication is was only a recommended practice document rather than a requirement. Thus it is not surprising that the handoff (or a *re-association*) technique presented in the standard is not much different from a fresh association. Because of these differences, the problem of designing efficient handoff mechanisms becomes very different from that of cellular networks.

Mobility Induced Locality

Handoff is the crucial link level function that enables users to move while staying connected to a wireless network. This applies equally well to the two types of networks we discussed before namely, cellular networks and wireless LANs. As users move within a wireless network, they induce a relationship among base-stations that they associate with in succession. We illustrate this with an example shown in Figure 1.2. Here a user is moving in the direction shown by an arrow. The user first associates to Base-station 1 and moves toward Base-station 2. Once the link quality degrades due to the user's mobility (and the resultant signal loss), the user performs a handoff to Base-station 2. By doing this, the user has created what we call *locality* among base-

stations 1 and 2. Base-stations 1 and 2 are considered local in the sense that users can obtain service from Base-station 1 and Base-station 2 in succession (or handoff in succession). This locality between Base-station 1 and 2 was *induced* by the mobility pattern of the user in this example. If we aggregate such locality created by all users over the entire set of base-stations that comprise the network under question, we observe an interesting locality based relation among the base-stations. We call this concept, *locality in mobility* or *mobility induced locality* among base-stations or access points. This locality can be thought of as a set-theoretic 'relation' over the set of base-stations. Two base-stations $\langle B_1, B_2 \rangle$ are set to be local if users can perform a handoff from B_1 to B_2. We refer to this as the *locality relation* R_L.

In essence, such a notion of locality captures the mobility patterns of all users in an aggregate manner to the granularity of the coverage of a base-station. For example, if a large number of low power base-stations (and thus small coverage region) are used to provide network coverage to a certain area, they would exhibit good diversity in the locality relation as opposed to using a few high power base-stations. Also this locality is a function of how users 'select' base-stations. In a tightly coupled system such as a cellular network where all devices belong to the same manufacturer and hence use the same algorithm to

determine the next base-station during a handoff, the locality relation would not vary from one user to another. However, in the scenario of a wireless LAN, the overall locality observed by the network would be an aggregate of the various individual decisions made by users executing different handoff algorithms.

It is important to note the asymmetric nature of the R_L relation. In the example of Figure 1.2, the occurrence of a handoff from B_1 to B_2 does not imply the definitive occurrence of a handoff from B_2 to B_1. This can be seen as follows. We are given that B_2 was the best base-station according to a certain criteria when the user was at the edge of the coverage of Base-station 1. Now, this does not imply that there exists a location where Base-station 1 would be the best base-station assuming that the user was on the edge of the coverage of Base-station 2. That is, the locality relation is asymmetric in nature.

It is trivial to see that for two base-stations to be considered local according to user mobility, they need to have an overlapping area of signal coverage. This is because the user needs to be able to perform a handoff between the two base-stations. If they do not overlap, the user might loose the network connection and initiate a new session with another base-station. Thus, the notion of the user maintaining a single session while moving within the network is important to defining the

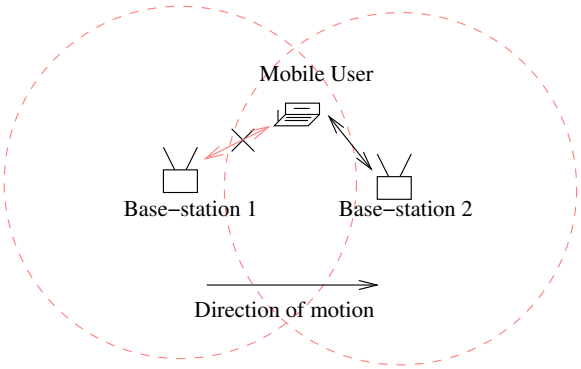


Figure 1.2: User mobility induces locality among base-stations.

concept of mobility induced locality.

1.1 The Research Problem

The concept of mobility induced locality among base-stations is very basic to user mobility. This applies in a general fashion to all infrastructure wireless networks where most communication is performed through a fixed infrastructure such as base-stations or access points. Representing such a locality (given by R_L) can have very interesting ramifications for the network designer. For example, this can be used to build better network management schemes. It can allow for quality of service provisioning among base-stations so as to provide a certain service guarantee to users. This can also allow for the design of ingenious security and network monitoring techniques to detect congestion and attacks.

In this dissertation, the first question we address is the following: Does there exist an efficient method of representing the mobility induced locality among base-stations or access points in such a manner that diverse network applications can take advantage of it ? To answer this, we first define the semantics of what 'an efficient method' should be.

1.1.1 Representing Mobility Induced Locality

Before network designers can take advantage of the locality among base-stations, we need a good structure that represents it. This structure needs to exhibit certain important properties discussed below, which will make it practically usable for diverse applications.

1. *Distributed Representation:* This structure should be representable in a distributed fashion. This is because access points, base-stations or users who desire to find other base-stations that are local (according to user mobility) should be able to do so by using local communication. That is, they should not be required to contact a central server to find local information. Of course, applications can trivially centralized a distributed representation if the need arises, and thus a distributed representation gives immense flexibility to the network designer.

2. *Robustness to failures/changes:* The structure should have methods associated with it that allow for resilience to failures at the network side or at the user. Also there could be changes in the physical environment that can alter the locality relations. For example, construction of new walls, streets, or changes to an office environment can bring about changes in the radio propagation which translates to changes in the locality relation. The locality structure needs to exhibit mechanisms that can let it adapt to such changes and converge to a more accurate form as quickly as possible.

3. *Independence from complex RF models:* The structure should not depend on the validity or computations resulting from the usage of any simple or complex radio propagation models. For example, radio map based techniques should not be used. This is important as it is not possible to contemplate all environments in which one might apply this structure and thus one cannot guarantee the validity of any radio propagation model. Also for any given environment, a propagation model is only approximate and thus errors can result.

4. *Autonomous Nature:* This structure will have various methods associated with it, such as for resilience, adaptation to environ-

ment changes and so forth. These methods should not require any human intervention such as a human dependent trigger or a decision model. This allows for autonomous operation of diverse applications which use such a structure to capture mobility induced locality.

In this work, we have proposed and evaluated a distributed graph theoretic structure called *Neighbor Graphs* to represent mobility induced locality. This structure applies to any wireless network in general and exhibits all of the above properties. We evaluate how well this structure represents locality by applying it to a real world problem in the context of wireless LANs. We examine the problem of high handoff costs in WLANs and design efficient schemes to eliminate such high costs. The success of the schemes depend on how well neighbor graphs capture this locality and our testbed based evaluations and simulation based study throw ample light on this subject. Next, we discuss specifics of the handoff problem in the context of wireless LANs.

1.2 The Cost of Mobility in Wireless LANs

Handoff can be defined as the procedure or process which results in the *transfer* of an existing *association* relationship ($\langle A, B \rangle$), typically symbolizing network connectivity, across two different entities ($\langle A, B \rangle \rightarrow \langle A, C \rangle$). Here, the common entity (A) is usually the client which changes connectivity at the link layer across two different base stations (B, C). The exact implication of this *transfer* would depend on the specifics of the wireless communication system used.

Handoff became an important problem in cellular networks in the late 1980s and the early 1990s. This was primarily because of the increase in an average user's mobility due to the various cost-effective means of transportation combined with the explosive growth of the cellular phone industry. Consequently, there is a large body of research on the handoff problem in cellular networks so much that the handoff primitive has today become an integral part of the underlying communication technology (for example UMTS, W-CDMA etc., refer [3] for further details).

While the handoff primitive allows for a user to be mobile and stay connected to a wireless network, nevertheless this incurs a cost

– the handoff latency. We define this as the duration of time during which the user stays disconnected to a network in a logical sense. That is, the user cannot send or receive voice or data signals during this period. Thus, while handoffs provided greater mobility to users they nevertheless brought about a cost associated with user mobility.

We first examine the process of handoffs in wireless LANs and discuss the cost associated with it in terms of the handoff latency. Recall that a wireless LAN consists of a set of access points (APs) or base-stations which are connected over a backbone wired network usually an ethernet. Multiple APs are used to expand the coverage of a single AP. Below we discuss the various logical steps involved in obtaining network service or performing a handoff as directed by the IEEE 802.11 standard for the wireless LAN.

Handoffs in Wireless LANs

Figure 1.3 shows the IEEE 802.11 state machine. This diagram conceptualizes the *state* of a client's connection to the 802.11 based wireless network. Any client desiring to connect to a given wireless network starts at State 1, where it is not associated and not authenticated. After performing authentication with an particular AP, the client transitions to State 2. Here, the client associates to the AP and enters

State 3 where it gains network access. The client can also re-associate from State 2 to State 3 depending on whether the client had a prior association to another AP.

We first discuss the specific functions that are available at the link level which are implemented by every wireless interface manufacturer that conforms to the IEEE 802.11 standard.

1. *Association:* This primitive allows a client to associate to an AP. After this the client can send and receive wireless traffic through the AP.

2. *Re-association:* Performing a re-association with an AP terminates the client's prior association with a different AP apart from creating a new association. This can result in transfer of client's *context* information from the old-AP to the new-AP.

3. *Scanning:* A client can perform an active scan, which is an intrusive and reliable method of determining availability of APs. The client sends explicitly *probe* messages to search for APs on specific channels. Another method available in the standard, is a passive scan, wherein the client passively monitors a channel for activity and attempts to associate based on collected information. It being very opportunistic and slow in nature, few wireless vendors use this method.

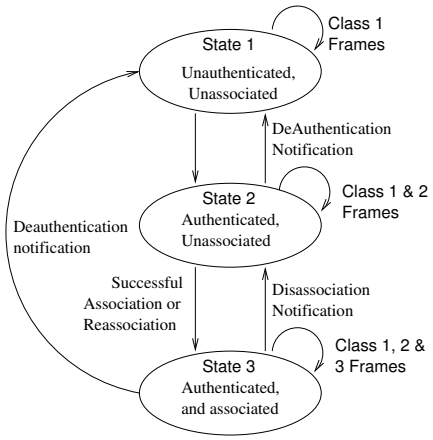


Figure 1.3: The IEEE 802.11 state machine reflecting a user's state of connectivity at the link layer.

Most handoff vendors implement a very simplistic handoff technique wherein they just perform an active scan of all available channels and associate to an AP that has the strongest signal as obtained by the scan function. This is essentially a brute force technique of searching for the next AP to handoff to. Hence, not surprisingly such handoff algorithms incur a heavy cost in terms of the latency, during which the client is unable to send or receive data packets. In Chapter 3, we study such algorithms in detail through testbed experiments. We found that the scan process incurred a latency of approximately 400 ms on average.

The scan phase returns a candidate set of APs organized by certain criteria typically the signal strength of the transmission from the

AP. Depending on the security policy in effect, the client would perform an IEEE 802.11i [4] based authentication to authenticate itself to the network and derive session level keys for per-packet confidentiality, authenticity and integrity. The authentication is performed based on the IEEE 802.1X [5] framework which uses the Extended Authentication Protocol (EAP) [6] to encapsulate a wide variety of authentication methods. This gives the network users flexibility in choosing an authentication method in a standardized and inter-operable fashion. The most commonly used authentication method is based on the Secure Socket Layer's Transport Layer Security protocol (SSL-TLS) [7, 8] which uses a public-key certificate based authentication.

We implemented the IEEE 802.1X framework as a part of the founded *Open1x* [9] effort. The authentication latency using the SSL-TLS authentication method was measured to be around 800 *ms* on average. This furthers the already high 500 *ms* latency of the scan phase bringing the handoff costs to around 1.2 seconds. Clearly, such high handoffs are a deterrent to a variety of applications, specifically voice over IP and more generally synchronous multimedia connections. However, the authentication latency comes into effect only if the network enforces strong IEEE 802.11i standard based authentication. Given the vulnerability of wireless networks to the so-called *parking lot attack*[10],

a greater number of wireless networks are enforcing tougher security measures on their served clients.

In addition to the above, if the network implements the Inter-Access Point Protocol (IAPP) [11] an additional latency might be incurred. The IAPP is used by the APs as a vehicle to perform the transfer of a roaming client's context information (quality of service parameters, billing, etc.) to the AP to which the client has re-associated. The IAPP 802.11f Draft Best Practice achieves this transfer in a secure and inter-operable manner. We have implemented IAPP over a wireless testbed and measured this latency to be around 15.3 *ms* on average. This latency, albeit small when compared to the other dominating components, still furthers the overall handoff latency.

Impact of High Handoff Costs

Based on this discussion, it is apparent that the handoff algorithms used by 802.11 wireless NIC vendors cause the applications using the wireless network to suffer due to the mobility of users. They experience intermittent periods of dis-connectivity which can last between 400 *ms* to 1.2 seconds depending on the security policy used. Many of the current handoff techniques are essentially brute-force as they perform a full scan of each channel during every handoff. Also the security

component of the handoff essentially implements a full-authentication with each new AP that the client associates to, thus discarding any trust and security relationship built with past APs.

For client applications especially those that are voice over IP driven, there is a need to design handoff algorithms that make mobility transparent from the higher layers. The core problem in this thesis is to research and study the viability of mechanisms that reduce the effect of user mobility on higher layers of the networking stack. Optimally, such mechanisms would totally eliminate the mobility effects thus making client mobility transparent to any of the upper layers (network, transport etc) and most importantly to the applications utilizing the network. That is, if a higher layer were to perform a test using an engineered sequence of packet transmissions, it should be unable to distinguish user mobility.

In a cellular network due to the regular structure of the cells [12], smooth and seamless handoffs are much easily achieved. Figure 1.4 shows an example of the regular structure used in a cellular network. Also shown is a typical channel assignment using three channels. There are a number of reasons for this. Firstly, the base-stations are fixed at a particular location for a very long duration of time. Combining this with the regular structure of cell neighborhood and a very deterministic

channel assignment used (example Figure 1.4) it is possible to pre-program deterministic handoff patterns which can be downloaded into the client device [13]. Secondly, all base-stations and compatible client devices are typically bought from a single manufacturer (or more than one with close collaboration) and thus they can be tightly coupled using proprietary protocols to pass back and forth information which can aide in smooth handoff decisions. Thirdly, because of this regular structure even dynamic handoff decisions can be made in real time owing to the regularities and tight coupling between the base-station devices [14, 15].

Wireless LANs pose an interesting and difficult problem because of the above differences. Access points are typically placed in an irregular fashion which is mostly an artifact of the building structure and the unpredictability of radio frequency (RF) behavior in indoor environments. Thus, it is hard to impose any specific regular structure, for example an hexagon, on the neighborhood of the BSSs. Also since the AP and client equipment are expected to be inter-operable with any manufacturer conforming to the standards, the system becomes very loosely coupled with regard to information sharing between the participating entities.

Because of the above observations, wireless LANs become an excellent playground to evaluate the impact of using a structure like neigh-

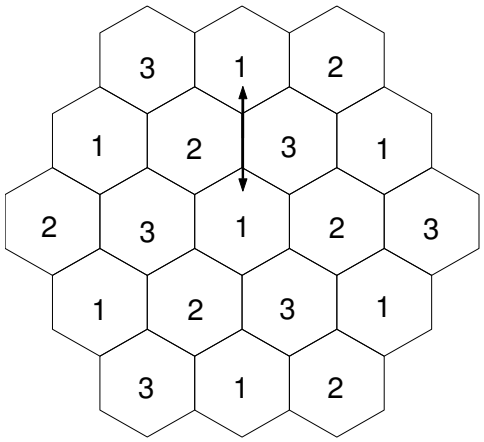


Figure 1.4: The regular structure of a cellular network.

bor graphs to represent mobility induced locality. The primary reason for this comes from the observation that current handoff algorithms recommended by the IEEE standard do not take advantage of the locality among access points or base-stations induced by user mobility patterns. A structure that accurately captures the locality among base-stations will lead to significant cost reductions during the handoff process. This is primarily because a handoff is essentially a local computation and the latency is attributed to the time required to collect information on the local neighborhood of a user. Thus, by designing new handoff algorithms that address each phase of the handoff process and studying their performance, we will be able to understand the real world impact of using neighbor graphs as a representation method for mobility induced locality among base-stations.

1.3 Key Contributions

The primary contribution of this work is the distributed graph theoretic representation of mobility induced locality (relation R_L) which applies to any wireless network in general. We discuss the notion of Neighbor Graphs which capture this locality in a distributed fashion. We present various methods that provide robustness to failures and adapt to changes in the physical environment. These methods do not require any human intervention and thus allow for autonomous maintenance of neighbor graphs.

Vertices in a neighbor graph represent access points. Directed edges reflect handoff relationships between APs. The graph can be constructed in an autonomous manner by observing handoffs performed by clients. Thus, using a neighbor graph, the set of APs that form the local neighborhood of an AP in a graph theoretic sense, captures the locality in a client's association patterns.

We evaluate the efficiency of this structure by applying it to a real-world problem in the context of wireless LANs. We examine the handoff problem in WLANs due to the high latency costs. We study various algorithms that address how the different phases of the handoff process take advantage of the locality in the wireless network. Specifically we leverage neighbor graphs to proactively transfer a roaming station's

context to the potential set of candidate APs (which store them in a cache) to eliminate the context-transfer latency due to the IAPP [11] communication. We use neighbor graphs to design a re-keying scheme called *proactive key distribution* which takes advantage of the initial full-authentication and the established trust relationship between neighboring APs to generate session key material when a client roams to the neighbor AP. This method incurs negligible overhead and virtually eliminates the need for a full and costly IEEE 802.1X authentication while providing the same security guarantees.

We study algorithms that improve the active scan latency using this locality information represented as a graph. We study the various standard and opportunistic optimizations that can improve the scan latency without compromising on the quality of the candidate set of APs returned to the client. We evaluate the performance of all the various mechanisms through rigorous testbed based implementations. Throughout this dissertation, we implement and study the mechanisms over an in-building testbed spanning four floors of an office building and with a capacity of up-to 40 APs in operation. Through simulations we study the performance of the techniques over various randomly generated topologies. This case study shows that neighbor graphs are an efficient and practically usable structure to represent mobility induced

locality in wireless networks.

1.4 Organization of Thesis

This dissertation is organized as follows: The next chapter presents the concept of locality in user mobility in greater detail. We present some analysis to show how this locality manifests itself into the associations performed by roaming wireless clients. From this we derive the concept of neighbor graphs as a structure that captures locality for a wireless network.

In Chapter 3, we present a detailed study of the handoff process based on observations over an in-building testbed network. In particular we note the high cost and the high variation in the costs involved with the scan phase. We also study the factors that contribute to this high latency and the variations observed. We performed this study with the handoff algorithms implemented by three different popular wireless interface vendors.

Chapter 4 presents the first evaluation of our hypothesis and the concept of neighbor graphs as a structure that captures the locality among APs in a wireless network. We discuss methods of constructing and maintaining the neighbor graph structure in an autonomous and a light-weight fashion. We evaluate the proactive caching technique

which keeps a roaming station's context one-hop ahead, thus eliminating the need for an intrusive IAPP communication during the handoff. We study this approach through a full-fledged testbed based implementation and examine its asymptotic properties through simulations over various randomly generated topologies.

Chapter 5 presents the fast re-keying technique called proactive key distribution. This mechanism leverages existing security material generated during a full IEEE 802.1X authentication to securely generate key material when the client roams to a neighboring AP. This key distribution technique virtually eliminates the need for a full IEEE 802.1X authentication which would have happened otherwise as stated in the IEEE 802.11i standard. This reduces the authentication costs of around 800 *ms* to about 2-3 *ms*. We study the performance of this algorithm through a testbed based implementation.

Chapter 6 discusses algorithms to improve the latency of an active scan. The locality information is represented as an overlap graph, an extension to the above discussed neighbor graph structure. We discuss various algorithms that perform standard and opportunistic optimizations to slice the scan latency and bring it under the 50 *ms* barrier for voice over IP[16] applications. As always we study the performance through a client-side implementation and show the reduction in the

latencies. We study various performance properties through rigorous simulations.

The research presented in this dissertation which specifically includes the representation of locality using neighbor graphs, its autonomous construction and maintenance methods along with the handoff mechanisms, is a crucial step toward making mobility transparent to the higher layers of the networking stack. As a lesson from this work, I strongly believe that the concept of locality induced by user mobility and the methods of capturing locality as a theoretical structure (as a graph or otherwise) are fundamentally important to developing mechanisms that make mobility transparent to user applications. I hope that the discourse through the remainder of this prose shows the path to this observation.

Chapter 2

Locality in Mobility and Neighbor Graphs

The great tragedy of Science – the slaying of a beautiful hypothesis by an ugly fact.

– Thomas Husley

Locality is a well studied concept in Computer Science. Locality gave birth to a very fundamental technique known as caching [17]. The principle behind caching can be stated as follows: Demand for a certain object A at time T suggests that with high probability there would be a similar demand for an object B at time $T + \delta$, where object B is *related* to A in the following manner. Either $A = B$ or there exists a distance function D, which defines the 'distance' between two objects in an abstract manner, such that $D(A, B)$ is bounded.

Caching is a general concept that takes advantage of this locality. When a request for the object A originates, the cache stores the object A for future and also potentially prefetches all such *local* objects B such that $D(A, B)$ is within a certain bound. The bound on $D(A, B)$ essentially quantifies the notion of locality in this case. The underlying assumption here is that there is locality in the pattern of requests. Since the cache prefetches such local objects, subsequent requests can be served directly from the cache. This brings about valuable optimizations in the turnaround time for the requests.

Locality exists in various arenas in computer systems. Some examples are as follows. Locality of reference [18] commonly means that there is locality in references to memory locations, disk sectors, file requests etc. Locality exists in program execution [19] which is used by various caches present in central processing units. There exists both spatial and temporal locality in streams of requests arriving at web servers [20]. Thus, caching based on locality has been extensively used to speed up system operations, be it file, web or memory requests etc.

As the core concept in this thesis, we explore locality among basestations or access points that form a wireless network. We focus on locality created by user mobility. There has been no prior research that explicitly states and explores this form of locality in wireless en-

vironments. Handover techniques in cellular networks indirectly take advantage of this locality by 'reserving' channels at an impending cell boundary. However, the regular structures of a cell enable very precise determination of the next base station and this has prevented systematic study of locality in an algorithmic fashion. We now state this principle of locality in a formal manner.

Locality Principle: Locality in user mobility can be stated formally as follows. Consider the example shown in Figure 2.1. A user is U moving with a constant speed s. We allow for the user to change direction, but assume for simplicity that his speed is constant. He is known to be at location X at time $t = T_1$. Now at time $t = T_2$ such that $|T_2 - T_1| \leq \delta$ where δ is bounded, the location of user U is expected to be accordingly bounded to a certain region Y as shown. The radius of this circular region is equal to $\delta * s$. The user might be located anywhere in this circular region depending on how frequently he changes direction. A more involved analysis can show a similar bound on this region Y, allowing for the user's speed to vary while placing realistic bounds on his acceleration. In an in-building wireless LAN environment, this locality is constrained by the building structures. That is, the physical barriers in an in-building environment will skew the distribution of the users location with the region Y shown in Figure 2.1 making certain

locations more probable than others.

A direct method of 'observing' this locality could be, for example, to employ a location determination system such as Horus [21] or Radar [22] which periodically measure a user's location. As the user moves through a wireless network, this locality manifests itself into the user's observed *association pattern*, i.e., the patterns of APs it associates with during this motion. We refer to this as *mobility induced locality*. We explore this further in Section 2.1. In Section 2.2 we discuss the notion of neighbor graph and how they capture this locality in a theoretical sense.

In this dissertation, we develop efficient handoff algorithms that improve the handoff costs by taking advantage of the locality induced by user mobility much like the way caching works. Similar to the concept of prefetching related items and storing them in a cache, the techniques of proactive caching and proactive key distribution (discussed later) prefetch and perform certain precomputations to reduce the handoff latency. In Section 2.3 we discuss such applications of locality based on neighbor graphs.

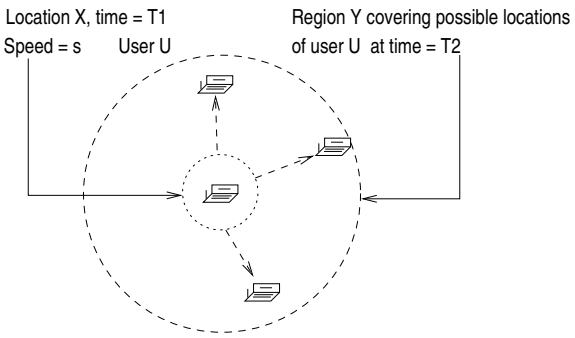


Figure 2.1: Example illustrating the locality in mobility.

2.1 Locality in Association Patterns

A Wireless LAN consists of multiple APs to provide extended coverage. A network user associates with an AP to become a part of the wireless network. The IEEE 802.11 standard defines association as a service that establishes an access point/station (AP-STA) mapping and results in the invocation of the distribution system services for the STA. Mobility of the user can cause it to move out of the coverage region of its associated AP and move into the region of one or more APs operating on non-overlapping channels. At this point, the user associates with a different AP and dis-associates with the previous one through a single procedure termed re-association.

If we look at the *sequence* of APs that a user associates to in succession, we can observe the locality manifesting itself in this sequence. That is, if a user re-associates between two APs ap_1 and ap_2, these

APs cannot be too far apart. That is, they have to be bounded in their physical separation. This follows directly from the principle of locality stated in the previous section. We call this sequence an *association pattern* as defined below:

Association Pattern: Define the *association pattern* $\Gamma(c)$ for client c as $\{(ap_1, t_1), (ap_2, t_2), \ldots, (ap_n, t_n)\}$, where ap_i is the AP to which the client re-associates (new-AP) at time t_i and $\{(ap_i, t_i)\ ,\ (ap_{i+1}, t_{i+1})\}$ is such that the handoff occurs from ap_i to ap_{i+1} at time t_{i+1}; the client maintains continuous logical network connectivity from time t_1 to t_n.

The effect of the locality principle can be reflected onto this association pattern in the following manner: Consider the example shown in Figure 2.2. Here, the user is associated to an AP ap_j at time $t = T_j$. Thus, its association pattern has an entry (ap_j, T_j). Now suppose that the user is associated to an AP ap_{j+k} at time T_{j+k}, that is , the k^{th} entry after (ap_j, T_j) in the association pattern is (ap_{j+k}, T_{j+k}).

The locality principle discussed earlier places a bound on the maximum distance traveled by the user in a given time frame. This, combined with the fact that an AP has a finite communication range essentially suggests that the physical distance between ap_j and ap_{j+k} is also bounded indirectly because of the locality principle. This is seen as follows: The maximum distance that the user could have traveled

during $\delta = T_{j+k} - T_j$ is given by $\delta * s$, where s is the speed of motion. As Figure 2.2 shows, D_{user} denotes the spatial separation between user's locations at times T_j and T_{j+k}. Thus, $D_{user} \leq \delta * s$. Since, the user is in communication range with ap_j at time T_j and with ap_{j+k} at time T_{j+k}, the maximum distance between ap_j and ap_{j+k}, denoted by D_{ap} is bounded by $D_{ap} \leq 2 * R + \delta * s$, where R is the maximum range of an AP. This result follows directly by applying the triangle inequality.

The above analysis shows how the two APs ap_j and ap_{j+k} are related. They can be thought of as being related via a parameter k, the 'distance' in the association pattern. Smaller values of k indicates a stricter bound. For the special case where $k = 1$, it shows the basic locality relation between successive associations and the corresponding APs. This relationship between the APs ap_j and ap_{j+1} will appear in multiple association patterns belonging to different users. In other words, ap_j and ap_{j+1} have a high probability of appearing as successive associations in any association pattern attributing to user mobility (such as walking patterns). This *defines* a relationship between ap_j and ap_{j+1}.

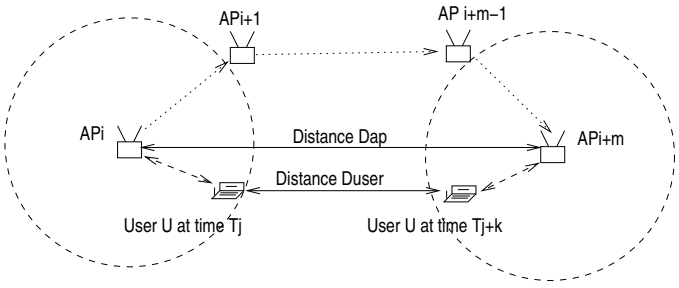


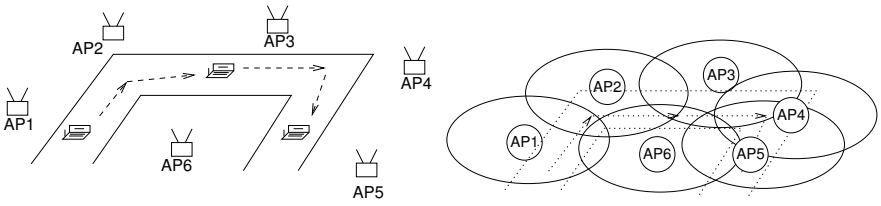
Figure 2.2: Example illustrating the locality in association patterns.

2.2 Neighbor Graphs

We now create a formal model that captures such relationships between APs attributing to locality. We define a 're-association relation' R as a formal relation between APs. This relates APs that have a high chance of occurring together in an association pattern. We define this formally as follows:

Re-association Relationship: Two APs, say, ap_i and ap_j are said to have a re-association relationship R_L if it is possible for an STA to perform an 802.11 re-association through some path of motion between the physical locations of ap_i and ap_j, for example as shown using the dashed lines in Figure 2.3(a). In a set theoretic manner, the tuple $\langle ap_i, ap_j \rangle \in R_L$.

We note two important properties of R_L:Firstly, R_L is asymmetric in nature. That is, $\langle ap_i, ap_j \rangle \in R_L \not\Rightarrow \langle ap_j, ap_i \rangle \in R_L$. This means that while it is possible to re-associate from ap_i to ap_j, the other way



(a) The physical Environment. (b) The coverage map.

Figure 2.3: Example of a wireless network and a user moving through the indicated path.

around need not be true. This asymmetry comes from the nature of RF in indoor environment and also depends on the criteria used by the wireless cards to make handoff decisions. Also as a note the relation R_L is non-transitive.

Relations can also be expressed in a graph theoretic manner, which allows us to build algorithms that take advantage of such structure. We thus, define a neighbor graph in the following manner:

AP Neighbor Graph: Define a undirected graph $G = (V, E)$ where $V = \{ap_1, ap_2, \ldots, ap_n\}$ is the set of all APs (constituting the wireless network under consideration), and there is an edge $e = (ap_i, ap_j)$ between ap_i and ap_j if they have a re-association relationship. Define $Neighbor(ap_i) = \{ap_{i_k} : ap_{i_k} \in V, (ap_i, ap_{i_k}) \in E\}$, i.e. it is the set of all *neighbors* of ap_i in G.

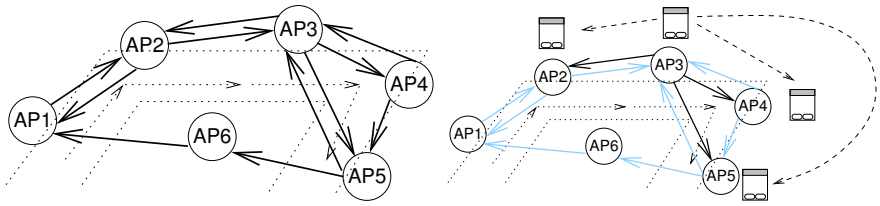
Given a wireless network, one can construct and maintain a neigh-

bor graph structure in a dynamic manner. The edges in the neighbor graph represent the relation R_L and have a finite lifetime. That is, if no user re-associates along a particular edge for a suitably long period of time, that edge is removed from the neighbor graph. Edges are added by observing handoffs performed by users. Such autonomous methods of edge-addition and deletion keep the neighbor graph *fresh* and in sync with changes to the network topology.

The neighbor graph can be implemented either in a centralized or a distributed manner. In this dissertation, we implement it in a distributed fashion for one set of applications. The construction and maintenance of this data-structure (in a distributed fashion) is discussed further in Chapter 4. We also implement it in a centralized fashion for the proactive key distribution algorithm which addresses the problem of fast authentications during handoff, discussed further in Chapter 5.

Consider an example shown in Figure 2.3(a). A user is moving within an in-building wireless environment as shown by the dashed arrows. There are six APs as shown. Figure 2.3(b) shows the coverage areas of the respective APs. As the user moves around, it performs handoffs between these APs. Figure 2.4(a) shows the neighbor graph constructed by observing a large number of handoffs. As can be seen

more edges are bidirectional indicating symmetry however, a few edges are unidirectional. Later in Chapter 4, we present the neighbor graph constructed over a realistic in-building testbed network.



(a) View from the 802.11 layer.

(b) Localized propagation of station's context and localized scanning.

Figure 2.4: View of the locality as captured by the neighbor graph.

2.3 Discussion

The neighbor graph structure captures the locality in mobility to the granularity of STA-AP associations. Thus, two APs that have an edge between them are considered 'local'. This structure in essence defines locality at a level that can be used by the wireless networking layer. An edge in the neighbor graph relates two APs as being 'local'. Thus, with a collection of such edges forming a graph, we have a structure that specifies what locality means at the network level. As a note, any user movements below the granularity of an 802.11 association are

'invisible' to the network and are hence not captured. Thus, neighbor graphs provide the network designer with a well defined algorithmic structure on top of which various mechanisms can be built. They lay the ground work for mechanisms that deal with user mobility. We now summarize how the concept of locality in mobility has been employed to address the problem of handoff in wireless LANs.

Proactive Caching: This technique propagates a station's context information to a localized set of APs such that the context information is readily available at an AP prior to a re-association. For example, in Figure 2.4(b), the context information stored at ap_3 is propagated to neighboring APs ap_2, ap_4 and ap_5 via the proactive caching method. This algorithm refers to the underlying neighbor graph to discover the APs ap_2, ap_4 and ap_5 as being local to ap_3. We study this technique fully in Chapter 4. Since mobility has locality, this technique performs well as long as the designated APs can store a user's context in their limited memory.

Proactive Key Distribution: This mechanism generates key material for fast re-authentication of a station during handoff. This key material is generated prior to the re-association and propagated to a candidate set of APs. This set is constructed using the neighbor graph which returns the localized set of neighboring APs. By restricting the propa-

gation of the key material to a limited set of APs that are considered 'local' to the user's current AP, the probability that a generated key will actually be used is greatly increased. Also this places strict bounds on the computation, communication and storage costs since this local neighborhood is bounded in size. Also a localized propagation reduces the chances that a randomly compromised AP can pose a threat to a moving user. We evaluate this approach in detail in Chapter 5.

Fast Active Scanning: Prior to re-associating with an AP, a station has to perform a scan of all channels to determine the set of APs along with their channel mapping. This information is used to create a priority list of APs according to the signal strength of the transmission from the AP. The results of this scan essentially gives the user a localized view of the network topology. This localized view is created by using structures such as neighbor graphs which define locality for the network. Using this local information, a client can reduce the number of channels to scan and the time spent on each channel waiting for responses from APs. This optimization is possible with the knowledge of the exact set of APs to scan along with their channel mapping. We shall observe in Chapter 6 that the currently used brute-force scan algorithms can be greatly optimized by taking advantage of locality in mobility to restrict the scan to a smaller set of APs and corresponding channels.

Chapter 3

The MAC Layer Handoff Process

Every great advance in science has issued from a new audacity of imagination.

- John Dewey.

The IEEE 802.11 standard does not recommend any specific *handoff algorithm* for client vendors to implement. The standard, however, defines certain MAC-level functions such as association, scanning, etc., which aide the mobile clients in making handoff decisions. In general, the goal of a handoff algorithm would be to provide current information about in-range APs to the client. This information could consist of the signal-to-noise ratio of the transmissions from each AP, the capabilities (data-rates, QoS, etc.) and potentially also load information in some form. Essentially the information about the set of APs in range gives a local view of the network to the user. As we shall ob-

serve from the analysis presented in this chapter, the current handoff algorithms essentially use a brute-force approach where they perform a full scan of each channel during the handoff. Using our concept of mobility induced locality as discussed in Chapter 2, such algorithms stand to benefit from prefetched information about the local view of the network and performing a localized search on specific channels for specific APs. This local view can be obtained from a neighbor graph which captures locality among APs in a graph theoretic manner.

The goal of this chapter is to evaluate the costs of performing handoffs in current wireless LANs as driven by the IEEE 802.11 specification. This would establish a baseline against which we can later compare the schemes that we build based on a representation of mobility induced locality as discussed earlier in Chapter 1. Specifically, we perform a study of the MAC level handoff process based on empirical measurements. The cost of this process is the latency incurred by the handoff algorithm during which the client could potentially be disconnected from the data service. Thus, a good handoff algorithm would provide accurate and concise information about potential 'next-APs' and incur minimum possible latency during this process. We conduct a study of the handoff process based on measurements performed on indoor wireless testbed networks. The goal of this study is threefold:

1. To study the distribution of handoff latencies incurred by mobile clients.

2. To observe the various 802.11 management messages exchanged during handoff and perform a logical analysis by categorizing the messages as belonging to various MAC-level primitives defined in the standard. This would lead to an outline of a tentative *handoff algorithm*.

3. To study the various factors, such as density of APs, response time, etc, which affect the handoff latency and to quantify such effects.

The measurements were done on three co-existing wireless networks (utilizing APs from different vendors) using three wireless NICs from different vendors. We analyze the handoff latencies by breaking down the handoff process into phases to assess the contribution of each phase to the handoff latency. Our results show that the *probe* phase is the significant contributor to the handoff latency. We performed an analysis of the probe phase and conclude with an evaluation of possible optimizations.

This chapter is organized as follows. Section 3.1 discusses the design of the experiment setup. Section 3.2 presents the observed handoff process divided into various phases and corresponding latencies. Sec-

tion 3.3 shows the distribution of the handoff latencies with the detailed breakup into various contributing phases. Section 3.4 presents a detailed analysis of the probe phase which is the primary contributor to the MAC level handoff latency. We summarize the key insights from this study in Section 3.5.

3.1 Design of the experiments

The experimental setup consisted of three in-building wireless networks, a mobile wireless client, and a mobile sniffing system. As shown in Figure 3.1 , the basic methodology behind the experiments, is to use the sniffer (in *close* proximity to the client) to capture all packets of interest related to the client for the analysis. This section describes the testbed networks, the sniffing system, and the client setup in detail.

3.1.1 The Wireless Network Environment

All the experiments were done in the A.V. Williams Building at the University of Maryland, College Park campus. This building hosts three co-existing wireless networks namely *cswireless*, *umd* and *nist* spanning four floors in all. The three networks have overlapping coverage, with *umd* covering the whole building, *nist* having half the coverage of *umd*, and *cswireless* covering one floor of the building. They are described

below :

1. The *umd* network: This network has 35 Cisco 350 APs distributed over four floors. This network uses open authentication. The gateway does a MAC address based access control for the data packets, and this does not have any effect on the MAC layer handoff process. The APs are configured to use channels 1, 6 and 11 only.

2. The *nist* network: This network has 17 APs covering roughly half of the building. The APs are built using a Soekris board [23], each using a Demarctech Prism 2.5 200mW wireless card, running OpenBSD 3.1 and using the *hostap* driver [24] for the AP functionality on the wireless interface. This network uses open authentication without access control. Again the APs use channels 1,6,11 only.

3. The *cswireless* network: This network has 8 Lucent APs in total. It uses a static shared key for WEP encryption. The APs are present on 8 different channels.

Methodology of each Experiment

The experiments were done in the following manner. A person with a mobile station walks through the building following a fixed path of

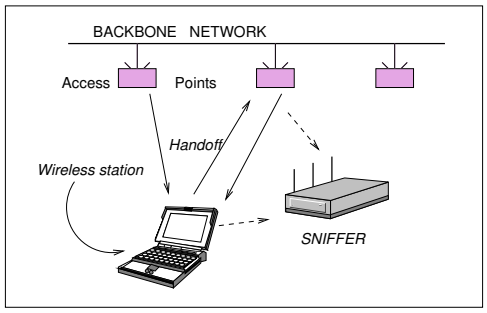


Figure 3.1: The Handoff Measurement Setup

travel (to minimize effects from the layout of APs) during each run. The duration of the walk, which is the duration of a single run of the experiment is approximately 30 minutes. Each experiment is characterized by the (i) wireless NIC used at the mobile station and (ii) the wireless network used. The mobile client sends negligible periodic ICMP messages to the network to maintain and display connectivity. Thus as the station moves, it performs handoffs as it leaves a BSS and enters another.

During the experiment on one wireless network, the other two coexisting networks were shutdown *i.e.*, the environment had absolutely no RF interference apart from the entities taking part in the experiment. Also the experiments were done when there was negligible user activity (during the early morning hours of weekends). This was done in order to minimize the effects of channel contention on the various latencies measured. Thus, the results in this work reflect zero con-

tention, and we reason that channel contention will only worsen the handoff latencies.

The mobile station is accompanied by a sniffing system which is designed (see Section 3.1.2) to capture packets of 'interest' i.e. the management frames constituting the handoff process. The sniffer is always in *close proximity* to the client, *i.e.* as close as physically possible and also moves with the station during the experiment. This helps in validating the sniffing system in the following manner :

1. Since the sniffer and station are in close proximity, any packets received by the client, will also (accuracy measured in the next section) be captured by the sniffer.

2. Frames sent by the station on a neighboring channel with respect to the sniffer, will be captured with high probability.

3.1.2 The sniffing process

In this section, we describe the sniffing system used, the accuracy of our sniffing method and its limitations.

Capturing packets on one channel

The wireless NICs based on the *Prism 2.5* chipset from Intersil [24] have a monitor mode in which the NIC captures all traffic (management and

data) on one particular channel and passes it to the driver. The wlan-ng linux driver [25] provides the functionality to capture the traffic, the *ethereal* sniffing program was used to capture and filter the traffic. A laptop with a PCMCIA prism 2.5 based wireless NIC was used to sniff one channel.

We measured the accuracy of this setup by having a source machine send sequenced UDP packets over a wireless network to a sink machine on the wired segment. The experiment was done in an RF free environment (i.e. no other STAs or APs were present in the RF medium). The sniffer was placed in close proximity to the source to reflect the sniffing setup in our later experiments. We observed a loss percentage of 0.2% averaged over five experiments, each sending 1032 packets in an interval of 10 seconds.

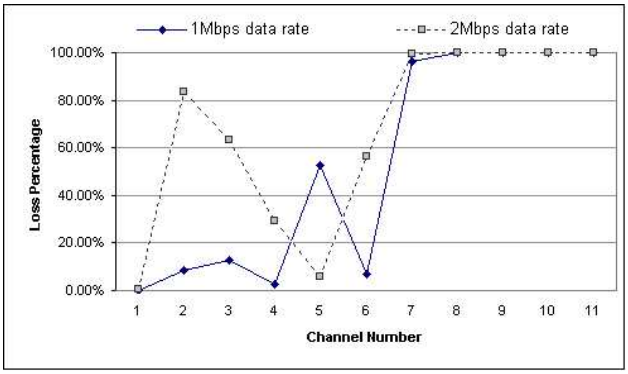


Figure 3.2: The loss percentage of the sniffer on neighboring channels. The traffic was sent on channel one.

Capturing packets on neighboring channels

Depending on the data rate, packets sent on one channel can be captured by the above sniffer on a neighboring channel. We performed an experiment to empirically observe the accuracy of sniffing traffic on neighboring channels.

The experiment consists of a source machine transmitting sequenced UDP packets through the wireless network on channel 1. The sniffer is progressively moved from channel 1 through 11. Figure 3.2 shows the loss ratio for various data rates. As can be seen, for packets sent at a data rate of 1Mbps, the sniffer can capture packets up to three neighboring channels (in either direction) with a maximum loss of 12% [1]. As a note, handoffs with missing packets (which can be detected using missing sequence numbers), were not taken into account during the analysis in this work. Hence although the loss effects the number of handoffs that can be studied, it does not compromise the accuracy of the latency measurements.

We reason to our best knowledge that because of poor *selectivity* of the radios used in the wireless NICs employed for sniffing, a strong signal on an adjacent channel is treated as a weak signal on the sniffing channel (since the transmitting client is always in close proximity to

[1] At 5.5Mbps and 11Mbps, no traffic was observed on any neighboring channel.

the receiver). This phenomenon, also known as *adjacent channel interference* [26], is being exploited by our sniffing mechanism to capture packets transmitted by the client on adjacent channels.

Design of the sniffer system

Based on the above observations we design the sniffing system in the following manner. The frames of interest are the *Probe Requests* and *Responses*, the *Re-association* and the *Authentication* frames. These frames are sent at the lowest data rate allowed i.e. 1Mbps (for maximum range and compatibility).

1. For frames sent by the STA (at 1Mbps), the sniffer (which is in close proximity) has to be capturing packets in a neighboring channel (or on the same channel), as discussed in Section 3.1.2.

2. For frames sent by an AP on a particular channel, the sniffer has to be capturing packets on the same channel for high accuracy. We require this constraint because the AP is *not* in close proximity to the sniffer.

Based on the above principles, we designed the sniffing system for the three networks in the following manner:

1. For the *umd* and the *nist* networks: Here the APs are on channels 1, 6 and 11. Thus the sniffer needs one NIC capturing packets on

each of these channels. We use two Linux machines, one with a single wireless NIC and the other with two wireless NICs (PCMCIA based) which sniff the three channels independently. The captured data is then merged using the timestamp on each packet. To minimize the inaccuracy caused by the inconsistencies of the system clock in the two machines, they were synchronized using the *Network Time Protocol* (NTP) through a point-point ethernet connection between the machines. Throughout the experiment, we maintained a clock accuracy of 80 μs or better between the machines (an error of less than 0.08% for latency of 1 ms). The GNU/Linux machines we used were IBM ThinkPad laptops with Pentium III 866 MHz and 256 MB RAM. The software for sniffing on each NIC was as discussed in Section 3.1.2.

2. For the *cswireless* network we used six Linux machines, sniffing all eleven channels. Five machines had two wireless NICs and one had a single wireless NIC. The machines were synchronized using NTP in a similar manner. The traces were combined and the duplicate packets were removed using the 802.11 sequence numbers.

3.1.3 The clients

The wireless NICs studied for the handoff process were from three different vendors, namely, Lucent Orinoco, Cisco 340, and ZoomAir prism 2.5[2]. The mobile station performing the handoff was an IBM Thinkpad T30 with Pentium IV and 512 MB RAM. The machine was running RedHat Linux 8.0 as the operating system.

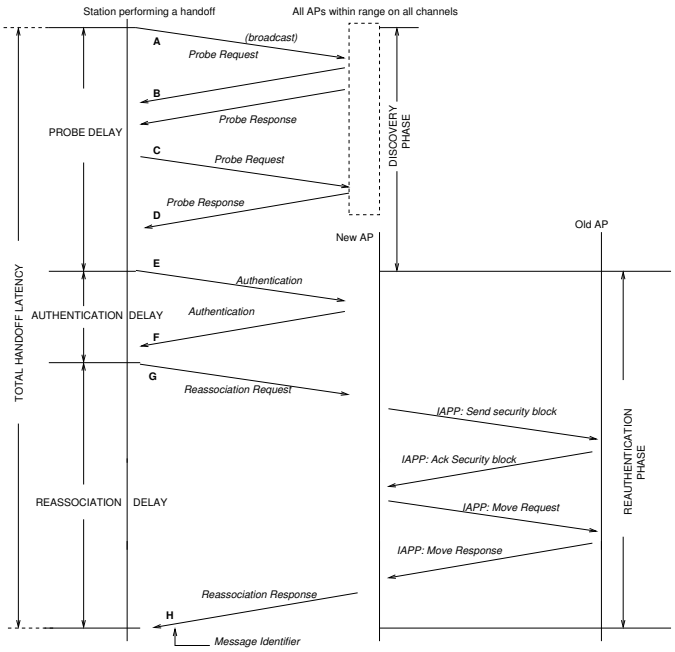


Figure 3.3: The IEEE 802.11 Handoff Procedure (followed by most cards)

[2]The secondary firmware versions on these NICs were as follows : Lucent Orinoco – 7.28.1, Cisco 340 – 4.25.10 and ZoomAir – 0.8.3.

3.2 Logical steps in a handoff

Here, we discuss the handoff process as observed in our experiments. Based on an observation of the sequence of messages, we first discuss the logical steps involved, and we analyze the latencies later in Section 3.3.

3.2.1 Logical steps in a handoff

The complete handoff process can be divided into two distinct logical steps:(i) *Discovery* and (ii) *Re-authentication* as described below.

1. Discovery: Attributing to mobility, the *signal strength* and the *signal-to-noise* ratio of the signal from a station's current AP might degrade and cause it to begin to loose connectivity and to initiate a handoff. At this point, the client might not be able to communicate with its current AP. Thus, the client needs to *find* the potential APs (in range) for a new association. This is accomplished by a MAC layer function: *scan*. During a scan, the card listens for beacon messages (sent out periodically by APs at the default rate of 10 ms), on assigned channels. Thus the station can create a candidate set of APs prioritized by the received signal strength.

There are two methods of scanning defined in the standard : *active* and *passive*. As the names suggest, in the active mode, apart from

listening to beacon messages (which is passive), the station sends additional probe broadcast packets on each channel and receives responses from APs. Thus, the station actively probes for the APs.

2. Re-authentication: The station attempts to *re-authenticate* to an AP according to the priority list. The re-authentication process typically involves an authentication and a re-association to the posterior AP. The re-authentication phase involves the transfer of credentials and other state information from the old-AP. As mentioned earlier, this can be achieved through a protocol such as IAPP [27]. In the experiments detailed in this paper, we do not utilize the standard IAPP communications, but we do permit the proprietary inter-access point communications (between APs of the same vendor). Thus, the authentication phase is just a *null* authentication in our experiments.

Figure 3.3 shows the sequence of messages typically observed during a handoff process. The handoff process starts with the first probe request message and ends with a re-association response message from an AP. We divide the entire handoff latency into three delays which we detail below.

1. **Probe Delay:** Messages A to E are the probe messages from an active scan. Consequently, we call the latency for this process, *probe delay*. The actual number of messages during the probe

process may vary from 3 to 11.

2. **Authentication Delay:** This is the latency incurred during the exchange of the authentication frames (messages E and F). Authentication consists of two or four consecutive frames depending on the authentication method used by the AP. Some wireless NICs try to initiate re-association prior to authentication, which introduces an additional delay in the handoff process and is also a violation of the IEEE 802.11 [28] state machine.

3. **Re-association Delay:** This is the latency incurred during the exchange of the re-association frames (messages G and H). Upon successful authentication, the station sends a *re-association request* frame to the AP and receives a *re-association response frame* and completes the handoff. Future implementations may also include additional IAPP messages during this phase which will further increase the re-association delay.

As a note, according to our analysis presented above, the messages during the probe delay form the discovery phase, while the authentication and re-association delay form the re-authentication phase. Apart from the latencies discussed above, there will potentially be a *bridging* delay caused by the time taken for the MAC address updates (using

the *IEEE 802.1d* protocol) to the ethernet switches forming the distribution system (the backbone ethernet). The results in our experiments do not reflect this latency.

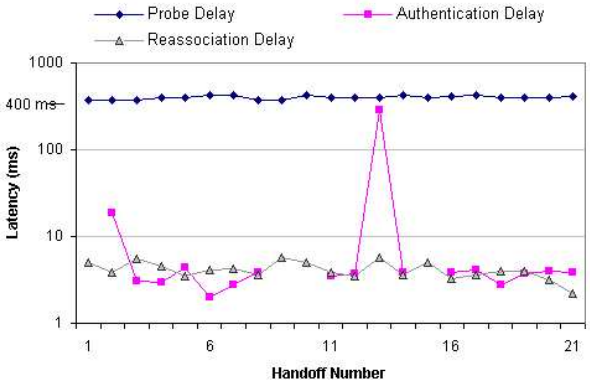


Figure 3.4: Handoff Latencies - Cisco 340 STA on *umd* (Cisco AP) network. Zero values are not plotted on the log-scale.

3.3 Experiment Results

Three wireless NICs and three different networks give nine experiments to run. Each experiment is characterized by the wireless NIC and wireless network being used. The experiments were performed as discussed in Section 3.1.

As a representative set, Figures 3.4,3.5 and 3.6 show the raw-breakup of the handoff latencies on the *umd* network for the three client NICs used. Figure 3.4 shows the handoff latencies for the Cisco

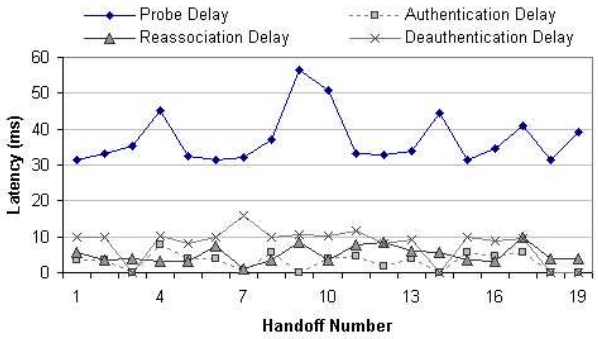


Figure 3.5: Handoff Latencies - Lucent STA on *umd* (Cisco AP) network.

340 STA, Figure 3.5 shows the latencies for the Lucent STA and Figure 3.6 shows the latencies for the ZoomAir STA. Figures 3.5 and 3.6 show a fourth delay, namely, *de-authentication* delay which comes into picture because of a different sequence of handoff messages followed by these NICs (Figure 3.9), discussed later.

Figure 3.7 shows the average values of the total handoff latency for the nine experiments along with the standard deviation (shown graphically on the bars). Figure 3.8 shows the average values of the four delays in the nine experiments.

Based on these results, the following direct conclusions can be drawn.

1. **Probe delay is the dominating component:** From Figure 3.8 it is clear that the probe delay accounts for more than 90%

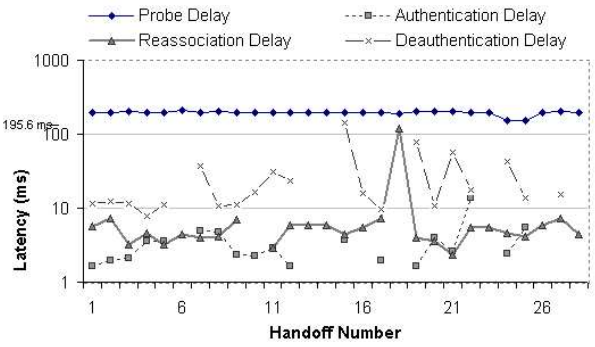


Figure 3.6: Handoff Latencies - ZoomAir Prism 2.5 NIC on *umd* (Cisco AP) network. Zero values are not plotted on the log-scale.

of the overall handoff delay, regardless of the particular STA, AP combination. Also even in the number of messages exchanged between the STA and the APs involved, the probe phase accounts for more than 80% of these in all cases. Thus any handoff scheme that uses techniques/heuristics that either cache or deduce AP information without having to actually perform a complete active scan clearly stand to significantly improve the handoff process.

2. **The wireless hardware used (AP,STA) affects the handoff latency:** We can infer this by observing two facts. Firstly, keeping the AP fixed, we can see that the client wireless card affects the latency. Figure 3.7 compares the average values of the latency among all nine configurations. Keeping the AP fixed, we can see a maximum average difference of 367.5 ms (Lucent STA

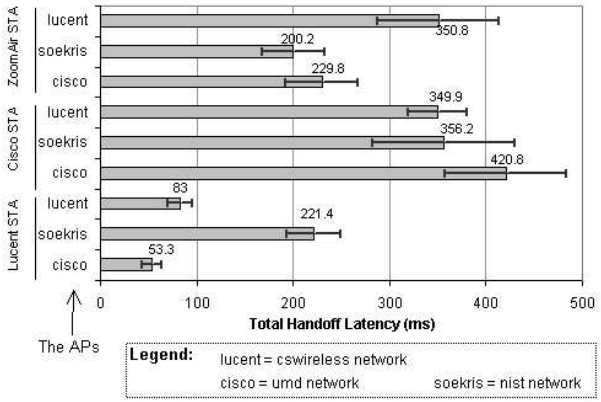


Figure 3.7: Handoff Latencies - Average values and standard deviation shown for all nine experiments.

and Cisco STA with Cisco AP). This is a huge variation by just changing the client card being used. Secondly, keeping the client card fixed, the AP also affects the latency but to a much lower extent (around 60% less): The maximum average difference (between the two APs for any fixed client) is 150.2 ms (Lucent AP vs Cisco AP for ZoomAir STA). This supports our previous result that the probe function is the dominating component since it is a firmware function (and implemented differently by different vendors) of the NIC cards. Refer to Section 3.4.2 for further analysis and reasoning on this observation.

3. **There are large variations in the handoff latency:** Apart from the variations in the latency with different configurations,

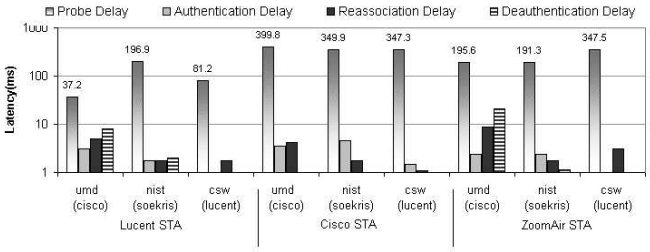


Figure 3.8: Handoff Latency Breakup - Comparison of the nine experiments.

we find significant variations in the latency from one handoff to another within the same configuration. This is also supported by the high standard deviations (Figure 3.7). Cisco STA on Cisco AP (*umd* network) has the largest standard deviation of 63.2ms. Also, we observe that the larger the handoff latency, the higher the variation.

4. **Different wireless cards follow different sequence of messages:** This is an observation from looking at the traces offline. We found that the ZoomAir and the Lucent NICs follow a slightly different procedure from the Cisco NIC, as shown in Figure 3.9. The figure shows that the card sends a *re-associate* message prior to *authentication* which it performed when the AP sends a *de-authentication* message. The figure also shows the modified semantics of the *re-association delay* and the *authentication delay*

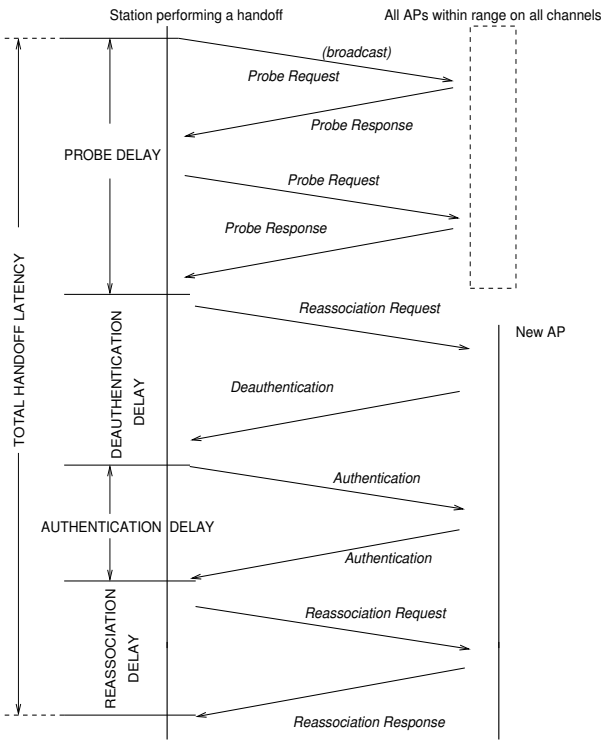


Figure 3.9: The Handoff Procedure as observed on the Lucent and ZoomAir wireless NICs.

for the ZoomAir cards. In order to attribute the delay caused by this modified sequence, we call the latency between the first *re-association* and the first *authentication* message as the *de-authentication* delay. This latency includes the *de-authentication* message as shown in Figure 3.9.

Thus the probe delay is accountable for the high handoff latency and also the variations in some cases. We present a detailed analysis of this phase based on the traces collected in the above experiments.

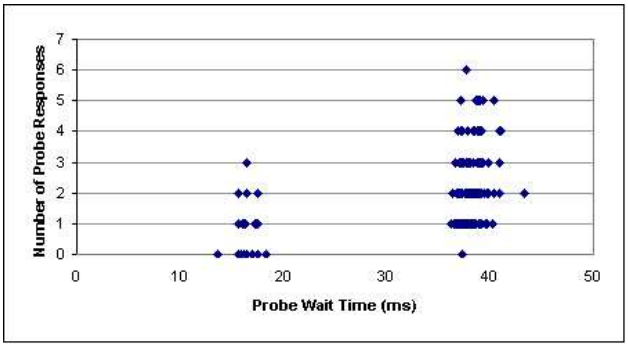


Figure 3.10: The distribution of the probe-wait times with respect to the number of probe responses received for the Cisco STA.

3.4 Analysis of the Probe Phase

In this section, we present a detailed analysis of the probe phase based on the experiment data. Presented first is the specification of the active scan algorithm from the standard ([28]), a discussion of the observations, and suggestions for improvement of the probe latencies.

3.4.1 The Probe Function Specification

The probe function is the IEEE 802.11 MAC active scan function and the standard specifies this as follows (modified for brevity):

For each channel to be scanned,

1. Send a *probe request* with broadcast destination, desired SSID, and broadcast BSSID.

2. Start a *ProbeTimer*.

3. If medium is *not busy* before the ProbeTimer reaches *MinChannelTime*, scan the next channel, else when ProbeTimer reaches *MaxChannelTime*, process all received *probe responses* and proceed to next channel.

As can be seen from the algorithm, *MinChannelTime* and *MaxChannelTime* are two parameters that determine the duration of scan for each channel. Figure 3.11 shows the messages in a probe phase. The STA transmits a probe request message and waits for responses from APs on each channel. Let *Probe-Wait latency* be the time an STA waits on one particular channel after sending the probe request. We measure this as the time difference between subsequent probe request messages. Thus the STA waits on one channel for *MinChannelTime*, and if any traffic (data or management frames) was observed or a probe response was received, the STA further extends the probe-wait period to *MaxChannelTime*. Thus according to the above procedure, the traffic on the channel and the timing of probe response messages affects the probe-wait time, i.e. the probe-wait time should be expected to be distributed between a *MinChannelTime* and a *MaxChannelTime* value. Hence the total probe delay, say t, for probing N channels, would be bounded by:

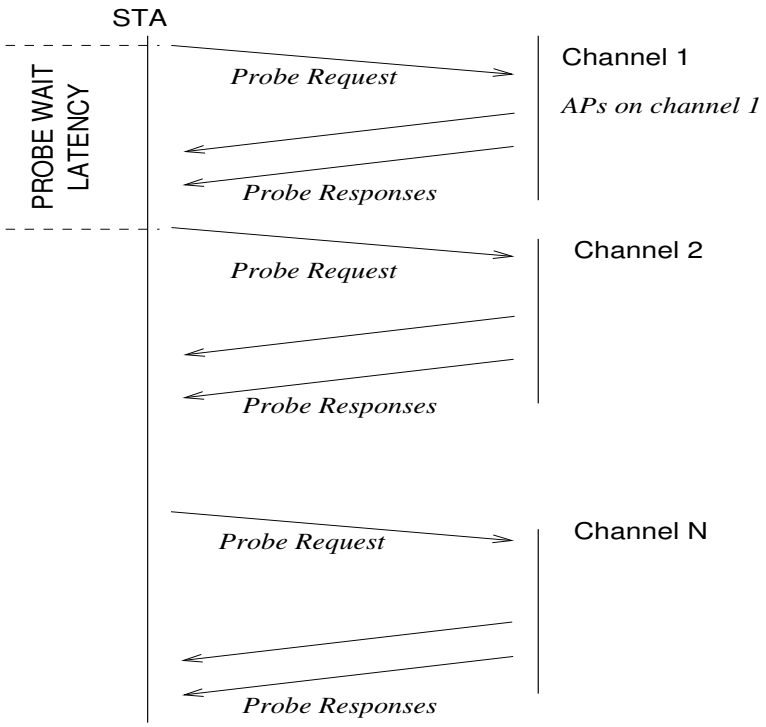


Figure 3.11: The messages in an active scan.

$$N * MinChannelTime \leq t \leq N * MaxChannelTime$$

In the next subsection, we present the empirical observations on the probe-wait time. We contrast this with the expected behavior according to the standard.

3.4.2 The Probe-Wait time: Observations

We study the probe-wait times for the three wireless NICs under study over the *umd* network.

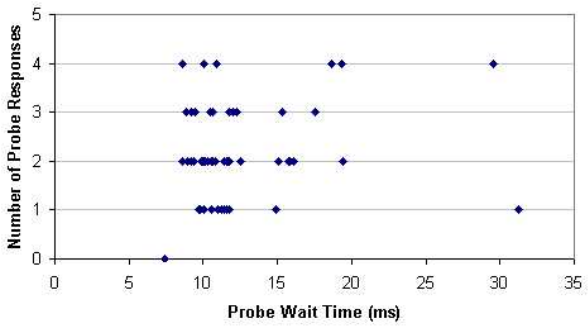


Figure 3.12: The distribution of the probe-wait times with respect to the number of probe responses received for the Lucent STA.

1. *Cisco 340 STA:* Figure 3.10 shows the various probe-wait times with respect to the number of probe response messages received by the Cisco STA on the *umd* (Cisco AP) network. The scatter-plot shows two clusters being formed, which more-or-less correspond to the *MinChannelTime* and *MaxChannelTime* values from the active scan algorithm. When no probe responses are received, (and the channel has no traffic, probably since there were no APs present) the probe-wait time is equal to the *MinChannelTime* which is around $17ms$ for the Cisco NICs. When there are responses (or traffic) on the channel, the NIC spends *MaxChannelTime* on the channel which is around $38ms$. The Cisco STA sends 11 probe requests in all, one on each channel.

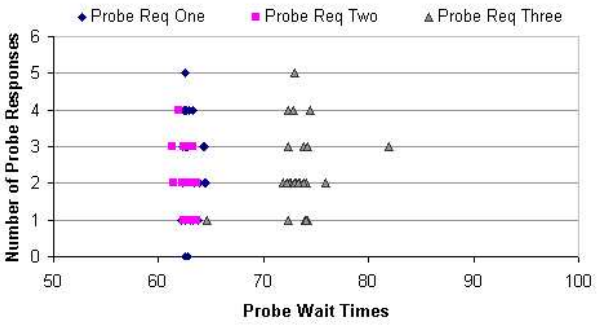


Figure 3.13: The distribution of the probe-wait times with respect to the number of probe responses received for the ZoomAir STA.

2. *Lucent STA:* Figure 3.12 shows the distribution for the Lucent STA on the *umd* network. Here the probe-wait times do not have significant correlation with the number of probe responses. Also the Lucent STA sends only 3 probe requests, one each on channels 1, 6 and 11. The probe requests are sent at 1Mbps, and can be received by the APs on the neighboring channels. Also the variation is not much, having a standard deviation of $4.2ms$.

3. *ZoomAir STA:* Figure 3.13 shows the probe-wait times for the ZoomAir STA on the *umd* network. Like the Lucent STA, ZoomAir also sends only 3 probe requests on channels 1, 6 and 11. The probe-wait times for the first two requests cluster around $63ms$ while the times for the third probe-wait clusters around $73ms$. The third (i.e. the last) probe wait time is measured as the

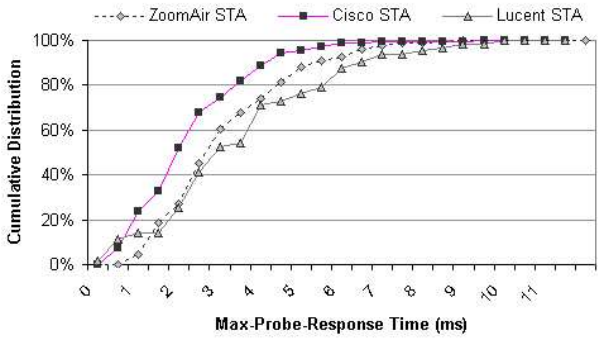


Figure 3.14: Cumulative distribution of the maximum probe response times observed by the three wireless NICs under study on the *umd* network.

difference between the last probe request and the re-association request frame sent by the STA. Thus the additional $10ms$ (on average) potentially goes into the processing of the probe results, making a decision about the AP to re-associate to and sending the re-association request. However, we did not observe a similar difference in the probe-wait times for the other STAs.

From the above analysis, it is clear that vendors implement different probing methods which reflect the large variation in the probe delays from one STA to another.

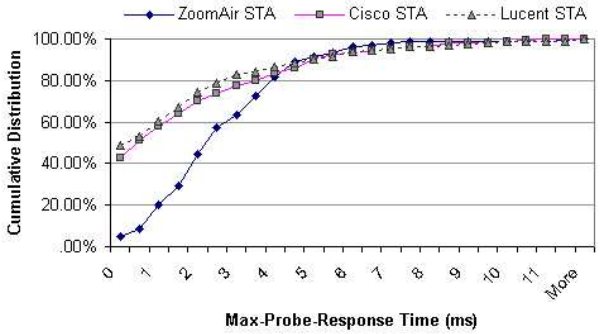


Figure 3.15: Cumulative distribution of the maximum probe response times observed by the three wireless NICs under study on the *cswireless* network.

3.4.3 Probe-Wait Optimizations

In this section, we discuss some optimizations on the probe-wait time based on the observations. In particular, we analyze the probe responses to determine a *good* value for the *MinChannelTime* and *MaxChannelTime* parameters which effect the probe-wait time.

Let AP ap_i be on channel L and assume that STA C is performing a probe. We define the *probe response time* from ap_i to be the time between the probe request message sent by C on L and the corresponding probe response sent by the AP ap_i. The *maximum* probe response time is the time between the probe request and the last probe response received by the STA C from any AP on channel L. Ideally the probe-wait time on every channel should be no larger than the

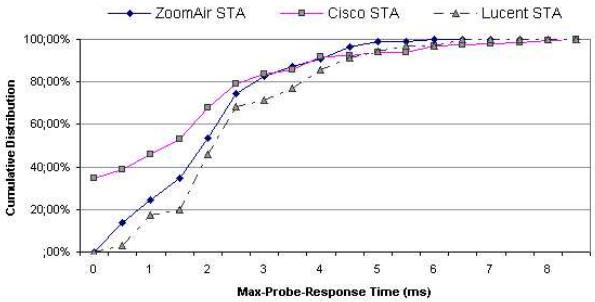


Figure 3.16: Cumulative distribution of the maximum probe response times observed by the three wireless NICs under study on the *nist* network.

maximum probe response time on that channel.

Figure 3.14 shows the cumulative distribution function of the probe response times for all three wireless NICs (on the *umd* network). From the graph it can be seen that all probe responses are received by a station within (approx.) $11ms$ for the *umd* network. Also in 90% of the cases all probe responses are received within (approx.) $6.5ms$. Hence a direct conclusion is that a *MinChannelTime* of around $6.5ms$ would a very good indicator of the presence of APs on the channel. And a *MaxChannelTime* of around $11ms$ would be sufficient to capture all the probe responses. This optimization can bring about a drastic reduction in the overall handoff latency. Even using a pessimistic probe-wait time of $11ms$, brings the handoff latency for the Cisco STA to around $121ms$

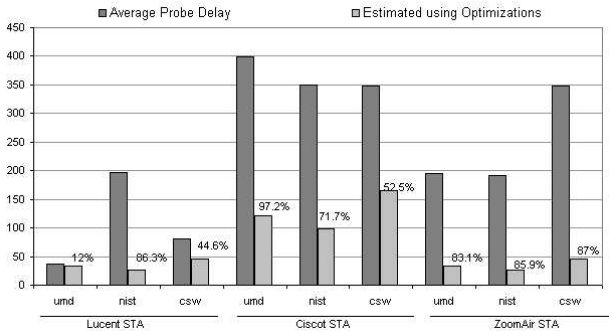


Figure 3.17: The average probe delay values, and the estimated probe delay values based on the pessimistic calculation of the *MaxChannelTime* for the nine scenarios. Also shown is the percentage improvement next to the estimated probe delay values.

for the 11 probe requests (from an average of 399.8ms, a reduction of around 70%), for the Lucent and ZoomAir to 33ms for the 3 probe requests (reduction of around 12% for Lucent and around 83.2% for ZoomAir).

Figures 3.15 and 3.16 show the same distribution calculated for the experiments done on the *cswireless* and the *nist* networks respectively. For the *nist* network, all responses are received within 9ms, while for *cswireless* the value is around 15ms. Using these pessimistic estimates (*MaxChannelTimes*), Figure 3.17 shows the expected probe-wait time improvements for the nine scenarios.

3.4.4 Hints for Fast-Handoff Strategies

Based on the observations from the previous sections, we present some simple heuristics to improve the probe-wait latency and also the overall handoff latency.

Reducing the Probe-Wait Latency

Based on observations from Section 3.4.3, we have the following methods to improve the probe-wait latency. These heuristics basically estimate a better fit for the *MinChannelTime* and *MaxChannelTime* parameters, thereby improving the probe-wait time:

1. An offline empirical analysis of the network as done in this work, can help come up with a good static value for these parameters which could be broadcast in the beacon messages or probe response messages from the APs.

2. The average AP density (i.e. number of APs *visible* per channel) could be broadcast in the beacon or probe response messages. The STA can wait for the corresponding number of probe responses and decide to switch to the next channel.

3. The STA could wait *sufficiently* long on one channel for the last probe response and empirically learn the *maximum* probe re-

sponse time. It can thus dynamically refine the *probe wait* time accordingly.

Other Optimizations

Handoff heuristics that require the least number of active scans will tend to perform the best. The following methods (or a combination of them) might be used to design heuristics and these are all attempts to avoid an active scan:

1. Using a distributed data-structure such as *Neighbor Graphs*. The AP-neighborhood relationship can be captured as a data-structure stored in the APs in a distributed manner. Each AP maintains a list of its neighbors, and using this information the STA performing the handoff can proactively determine its next AP instead of performing the scan process. Since this information it not dynamic (i.e. the AP topology does not change rapidly), it can bring about significant improvements in the overall handoff latency by potentially eliminating the probe phase completely. We discuss this initiative further in Chapters 4 and 6.

2. Query an external agent that provides *hints* on the potential next APs and their channels i.e a *map* of the APs based on the location. Pack et. al. in [29], [30] propose a technique in this category.

3. Interleave scan messages with data during normal connectivity and use that information to perform a partial active scan (or no scan at all) during the handoff. Also passive scanning (listening for beacon messages) might be performed during normal connectivity to build up the list of APs.

4. Since the probe-wait time depends on the number of probe responses, another strategy might be to create an *ordering* among the APs such that a single AP or a small set of APs is responsible for probe requests (i.e. the number of probe responses is a constant).

3.5 Discussion

In this chapter, we performed a detailed analysis of the handoff process, the factors that bring about the high latency and the variation and the various messages/steps involved. We find that out of the three basic functions (probe, authentication and re-association), carried out by the STA, the probe phase has the dominant latency regardless of the AP-STA being used.

We also performed a detailed analysis of the probe phase, and account for the large variation to the *probe-wait* time which essentially

depends on the particular heuristic employed by the wireless client NIC being used. We observed that two specific parameters namely *MinChannelTime* and *MaxChannelTime* have a significant effect on the overall handoff latency.

In our experiments we used wireless PC cards from three vendors, namely Lucent Orinoco, Cisco Aironet, and ZoomAir and the APs from Lucent, Cisco and Demarctech. This provided good diversity in our experiments, and we find that there is large variation in the latency with the particular AP-STA hardware being used. Also we find that the sequence of messages exchanged during the handoff process can also differ with the STA being used.

The probe phase which dominates the MAC layer handoff latency essentially discovers the 'position' of the client in the wireless network relative to the APs. In Chapter 2, we discussed how the locality 'binds' APs that have a re-association relationship. Thus, between successive handoffs the results of the active scans are very much related. This follows directly from the locality principle presented in Chapter 2. Specifically if a user performs a handoff from ap_i to ap_j, this implies that ap_j was the AP with the strongest signal as a result of the active scan, while ap_i was the best AP as a result of the previous active scan. This shows the relationship between ap_i and ap_j as captured by the neighbor

graph.

The current active scan algorithms used by popular vendors are very simplistic in the sense that they do not utilize any information about the local topology. As a result, they have to scan each and every channel regardless of whether APs are present or not. And so they incur very high latencies which will not meet the expectations of the Fourth generation wireless networks. This is because the handoff latencies we measured far exceed guidelines for jitter in voice over IP (VoIP) applications where the overall latency is recommended not to exceed 50ms [16].

Since the results of successive active scans (as a part of successive handoffs) are very much related by locality, this brings about the possibility of interesting optimizations which take advantage of this locality. Neighbor graphs provide a robust and autonomous mechanism of capturing this locality among APs in a graph theoretic sense. In Chapter 6, we design active scan algorithms which employ 'prefetched' information that gives a local view of the network, to perform fast active scans.

Chapter 4

The Proactive Context Caching Algorithm

From a network architecture point of view, a wireless LAN very closely follows the design of a wired network based on the IEEE 802.3 Ethernet standard. In wired ethernet, end-user machines are connected to a bridge/hub which acts as a layer-2 relay. A collection of such bridges are connected to a switch. In a wireless LAN, an access point (AP) acts as a layer-2 bridge between 802.11 based wireless media and the wired ethernet. Although the IEEE 802.11 standard also specifies a Wireless Distribution System (WDS) wherein the backbone network is built using multi-hop link-level connections, the most common architecture followed by today's widely deployed WLANs is similar to the one shown in Figure 4.1. Figure shows a set of N APs connected via ether-

net backbone to a gateway and a central server. Such a backbone could be expanded using a network of switches to support a greater number of APs. The optional gateway routes traffic to an external network (such as the Internet). The central server performs the role of authentication, accounting and authorization (AAA) and is hence called a AAA server. This communicates with the APs using an AAA protocol such as RADIUS [31], for performing the various AAA functions.

The most important distinction of a wireless network over its wired counterpart is the mobility of its users. In the current network architecture which spans the hardware and the various layers of the networking stack, reflections of user mobility or other characteristics which distinguish one deployment from another are absent. We illustrate this with an example. Figure 4.2 shows two different physical environments where the same network consisting of 3 APs is deployed. From an architectural and topological perspective, one can observe that currently both deployments are considered indistinguishable. That is, there is no state information present internally in the network that captures key differences between different deployments.

Lets elaborate on the above issue. In the first situation shown in Figure 4.2, the 3 APs cover a large conference hall. Here, all APs roughly experience similar load and have a similar pattern of roam-

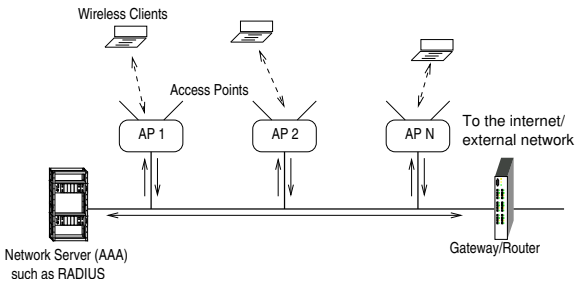


Figure 4.1: Network architecture of a typical 802.11 based WLAN.

ing users. This happens as the user density is typically distributed uniformly across the conference hall. The second situation shows three APs covering a classroom and the corridors around it. Here, AP_1 covers bulk of the classroom where most of the users are physically located. AP_1 serves users which are mostly stationary and tend to load the wireless network, while the other two APs serve users who are mostly roaming and do not tend to create a lot of network load. Thus, the physical environment where a wireless network gets deployed can completely alter the network usage pattern in terms of load, handoffs, etc, as observed by each individual AP. These observations have been reflected earlier in empirical network studies such as [32, 33, 34, 35].

From a topological perspective, both networks of Figure 4.2 have the same network topology, while they experience widely different usage and service patterns. To the network and link level entities, such as APs, backend servers and gateways, both scenarios are logically indis-

tinguishable. This is primarily because the IEEE 802.11 architecture (as discussed earlier) does not contain any means or apparatus for capturing the effects of the physical environment in which the network is being deployed. Lack of this basic ability is the prime hindrance to the development of sophisticated algorithms which can dynamically adapt and *fine tune* the network to the demands and characteristics of the wireless users.

Mobility Induced Locality

We had presented the concept of locality in user mobility in Chapter 2. This principle states that there is strong correlation between successive associations in an association pattern (which is defined as the set of APs that a client associates in succession). This locality is the key characteristic that distinguishes one deployment from another. In the example of Figure 4.2, the users move differently and hence associate differently, thus bringing about wide dis-similarities in the usage patterns although both networks have the same topology from an network standpoint. This difference in locality is a property of the physical environment which manifests itself into the usage patterns.

In Chapter 2, we discussed the notion of *neighbor graphs*. We discussed how neighbor graphs capture this locality as an asymmetric

relation between APs. We construct a relationship between APs based on the mobility characteristics of the users in the wireless environment. In particular, we construct a directed edge between two APs ap_i and ap_j if a handoff occurred from ap_i to ap_j. This signified that the two APs ap_i and ap_j were related due to locality. The graph thus constructed is called a neighbor graph. In essence, neighbor graphs capture the *mobility topology* of the wireless network as discussed earlier and can be constructed by real-time examination of the handoffs occurring in the network in either a distributed fashion at an AP, or in a centralized fashion at the AAA server. A path in a neighbor graph consisting of a set of directed edges represents a mobility path or an association pattern between the vertices, or APs. Therefore, given any edge, $e = (u, v)$, the *neighbors* of v represent the *local neighborhood* of the vertex/AP v. This information can be used to construct a set of potential next-APs for handoff. We present efficient distributed algorithms to construct and maintain the neighbor graph for a wireless network in Section 4.2.

The neighbor graphs constructed for the example of Figure 4.2 would bring out the differences between the usage patterns. The neighbor graph structure is readily available to the network layer in the form of state information at the APs or at a central server. This lays the groundwork for developing algorithms which improve the handoff la-

tencies by taking advantage of this locality information available at the network layer in the form of neighbor graphs.

Proactive Caching

Previous studies of wireless network mobility have shown that users tend to roam in what we call *discrete mobility* where the user utilizes the network while stationary (or connected to the same base station) and before moving the user ceases operation only to continue using the network after moving to a new location [32, 33, 34, 35]. That is, the users do not usually move while using the network because the majority of current network applications and equipment do not easily lend themselves to what we call *continuous mobility* where the user moves while utilizing the network.

Voice based applications are the usual application in *continuous mobility* as seen in the current cellular networks. We expect voice and multimedia applications will serve as the catalyst for *continuous mobility* in Wi-Fi networks much as they did for the cellular networks once multi-mode handsets and end-user applications become more widely available.

Supporting voice and multimedia with continuous mobility implies that the total latency (layer 2 and layer 3) of handoffs between

base stations must be fast. Specifically, the overall latency should not exceed 50 ms to prevent excessive jitter [16]. Unfortunately, the vast majority of Wi-Fi based networks do not currently meet this goal with the layer 2 latencies contributing approximately 90% of the overall latency which exceeds 100 ms [36, 37]. [36] suggests various mechanisms to reduce the layer 2 latency to within 20 to 60 ms depending on the client. Handoffs involve transfer of station *context* [38], which is the station's session, QoS and security related state information, via inter-access point communication. This transfer only furthers the handoff delay by an average 15.37 ms.

One method of reducing the context transfer latency of handoffs is to transfer or cache context ahead of a mobile station in a *pro-active* fashion. Unfortunately, the previous work on context transfer has focused solely on *reactive* context transfers, *i.e.* the context transfer is initiated only after the mobile station associates with the next base station or access router resulting in an overall increase in the latency of the handoff rather than reducing it [37, 39]. The problem with *pro-active* approaches, however, is how to determine the set of potential next base stations without examining the network topology and manually creating the set.

From our discussion on locality, it is evident that this set of po-

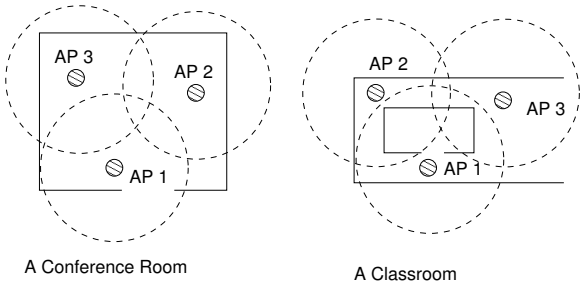


Figure 4.2: Same network deployed in different physical environments.

tential next base stations is essentially the local neighborhood of the client with respect to the APs. Based on our locality principle, neighbor graphs capture this locality in form of a graph theoretic structure readily available at the APs in a distributed manner or at a central server. Thus, it makes algorithmic sense to construct the candidate set of APs proactively by taking advantage of the locality as captured by the neighbor graph. Because of the validity of the locality principle, as discussed in Chapter 2, such schemes are bound to predict this set of candidate APs very accurately.

In this chapter, as an application of neighbor graphs, we develop an algorithm, called *proactive caching* that pre-positions a roaming station's context one-hop ahead of its current AP. We have implemented neighbor graphs using the IEEE Inter-Access Point Protocol ([39]). We find that using neighbor graphs the re-association latency reduces from 15.37 ms to 1.69 ms. We also find through simulations that as users

become more mobile the effectiveness of our solution increases, *i.e.* the context *cache hit ratio* increases to over 98% in most cases with reasonable cache sizes. The *proactive context caching and forwarding algorithm* presented in this work has been included in the IAPP standard [39]. These empirical results strongly confirm the validity of the locality principle which was shown analytically in Chapter 2. It also shows the usefulness of neighbor graphs as a vehicle that captures locality in a graph theoretic sense and makes it readily available to the network layer for possible optimizations.

The rest of this chapter is organized as follows. The next section discusses the context transfer process in detail. Section 4.2 recapitulates neighbor graphs and discusses their construction methods in detail. Section 4.3 discusses the proactive caching algorithm based on neighbor graphs. Section 4.4 provides an analytic evaluation of the performance of the caching mechanism which dictates the handoff latency resulting from the cache hits/misses. Section 4.5 evaluates the proactive caching approach experimentally through a testbed implementation and using extensive simulations. Section 4.6 discusses prior research work related to the proactive caching solution. We summarize the key contributions in this chapter in Section 4.7.

4.1 Context Transfer Process

A roaming client's context consists of state information created during the first association performed by the client. Such state information could consist of service agreement parameters, QoS information, client capability information, etc. The IEEE standard on inter-AP communication [39], specifies such a protocol, called the Inter-Access Point Protocol (IAPP) for the transfer of a station's context *during* re-association. Figure 4.3 shows the messages exchanged during the re-association phase. For a complete description of the handoff process and its constituent phases, the reader is referred to Chapter 2.

The IEEE IAPP standard specifies two types of interaction for completing context transfer [40]. The first form of interaction occurs between APs during a handoff and is achieved by the IAPP protocol, and the second form of interaction is between an AP and the RADIUS server[41].

IAPP plays a significant role during a handoff. The two main objectives achieved by inter-access point communication are : (a) *Single Association Invariant:* Maintaining a single association of a station with the wireless network, and (b) the secure transfer of state and context information between APs involved in a re-association. The client context information [38] can include but is not limited to IP flow

context, security context, quality of service (QOS) information, header compression information and accounting information.

Association and re-association events change a station's point of access to the network. When a station first associates to an AP, the AP broadcasts an *Add-Notify* message notifying all APs of the station's association. Upon receiving an *Add-Notify*, the APs clear all stale associations and state for the station. This enforces a unique association for the station with respect to the network. When a station re-associates to a *new-AP*, it informs the *old-AP* of the re-association using IAPP messages, see Figure 4.3.

At the beginning of a re-association, the new-AP can optionally send a *Security Block* message to the old-AP, each of which acknowledges with an *Ack-Security-Block* message. This message contains security information to establish a secure communication channel between the APs. The new-AP sends a *Move-Notify* message to the old-AP requesting station context information and notifying the old-AP of the re-association. The old-AP responds by sending a *Move-Response* message.

For confidentiality of the context information, IAPP recommends the use of an AAA server such as RADIUS (to obtain shared keys) to secure the communication between APs. The RADIUS server can also

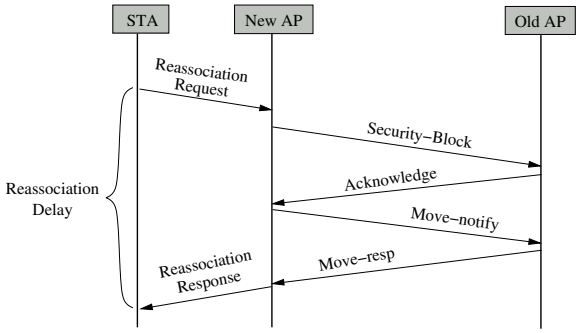


Figure 4.3: IAPP interaction to facilitate context transfer during re-association.

provide the address mapping between the MAC addresses and the IP addresses of the APs, which is necessary for IAPP communication at the network layer.

Although the IAPP communications serve to fulfill the mandatory DS functions, they invariably increase the overall handoff latency because of their reactive nature. In our testbed implementation, we observed the IAPP context transfer latency to be average of $15.39ms$ (Section 4.5.1). In this chapter, we leverage neighbor graphs for optimizations which eliminate this context transfer latency with high probability.

4.2 Neighbor Graphs

In this section, we summarize the definition of neighbor graphs. The reader is referred to Chapter 2 for a more detailed discussion of this concept.

As seen in Figure 4.3, the re-association phase primarily involves the transfer of station context from the old-AP to the new-AP. In order to improve the re-association latency, the context transfer process (using IAPP) must be separated from the re-association process. This can be achieved by providing the new-AP with the client-context prior to the handoff, or pro-actively. Since we are unable to predict the mobile station's movement, we leverage neighbor graphs for determining the *candidate set* of potential new-APs to perform the transfer prior to the handoff. Neighbor graphs provide the basis for identifying this candidate set.

4.2.1 Definitions

Re-association Relationship: Two APs, say, ap_i and ap_j are said to have a re-association relationship if it is possible for an STA to perform an 802.11 re-association through some path of motion between the physical locations of ap_i and ap_j.

The re-association relationship depends on the placement of APs,

signal strength and other topological factors and in most cases corresponds to the physical distance (vicinity) between the APs. Given a wireless network, we can construct and maintain the neighbor graph structure in a dynamic manner.

Association Pattern: Define the *association pattern* $\Gamma(c)$ for client c as $\{(ap_1, t_1), (ap_2, t_2), \ldots, (ap_n, t_n)\}$, where ap_i is the AP to which the client re-associates (new-AP) at time t_i and $\{(ap_i, t_i)\ ,\ (ap_{i+1}, t_{i+1})\}$ is such that the handoff occurs from ap_i to ap_{i+1} at time t_{i+1}; the client maintains continuous logical network connectivity from time t_1 to t_n.

AP Neighbor Graph: Define a undirected graph $G = (V, E)$ where $V = \{ap_1, ap_2, \ldots, ap_n\}$ is the set of all APs (constituting the wireless network under consideration), and there is an edge $e = (ap_i, ap_j)$ between ap_i and ap_j if they have a re-association relationship. Define $Neighbor(ap_i) = \{ap_{i_k} : ap_{i_k} \in V, (ap_i, ap_{i_k}) \in E\}$, i.e. it is the set of all *neighbors* of ap_i in G.

The neighbor graph can be implemented either in a centralized or a distributed manner. In this work, we are implementing it in a distributed fashion, with each AP storing its set of neighbors. The construction and maintenance of this data-structure (in a distributed fashion) is discussed next.

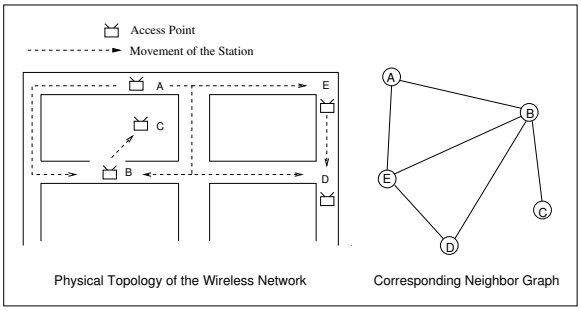


Figure 4.4: Figure shows an example placement of APs and the corresponding neighbor graph.

4.2.2 Construction and Maintenance

The neighbor graph can be automatically generated (i.e. learned) by the individual access points over time. There are two ways that APs can learn the edges in the graph. Firstly, when an AP receives an 802.11 re-association request frame from a STA, the message contains the MAC (BSSID) of the old-AP and hence establishes the re-association relationship between the two APs. Secondly, receipt of a *Move-Notify* message from another AP via IAPP also establishes the relationship. These two methods of adding edges are complementary, and the graph will remain undirected.

Each AP maintains the edges locally with a timestamp. This is necessary in order to eliminate the outliers, i.e. incorrectly added edges, or edges that need to be 'un-learnt' due to changes in the network topology (such as changes in AP positions, etc). For example, an

edge could be added incorrectly as follows: Consider a situation where a client goes into the power save mode and wakes up in a different location to re-associate to an arbitrarily different AP on the wireless network. A timestamp based approach would guarantee the freshness of the neighbor graph and eliminate the outlier edges over time.

The autonomous generation also eliminates the need for any survey or other manual based construction methods. As a result, this also makes the data-structure adaptive to dynamism in the re-association relationship (i.e. changes in AP placements, physical topology changes, etc).

The graph is generated by executing the following pseudo-code at each AP. ap_{host} is the AP on which the algorithm is assumed to be executing:

1. *Receipt of a re-association request:* When a client c *re-associates* to ap_{host} from ap_i, add edge ap_i as a neighbor of ap_{host} (i.e. ap_{host} adds ap_i to its list of neighbors).

2. *Receipt of an IAPP Move-Notify:* When ap_{host} receives a *Move-Notify* from ap_i, add ap_i to the list of neighbors.

4.3 Proactive Caching

We employ neighbor graphs to develop a distributed algorithm that proactively pre-positions a roaming station's context to potential 'next-APs'. This strategy, called *proactive caching*, maintains the context one-hop ahead of the mobile user. The APs, being memory-constrained embedded systems, store the context of such mobile users in a fixed size *context-cache*. The proactive caching algorithm specifies the context forwarding technique and the caching details.

Caching strategies are based on some locality principle, eg: locality of reference, execution etc. In this environment, we have locality in the client's association pattern. In this section we discuss the proactive caching strategy, based on locality of mobility as discussed earlier in Chapter 2.

The following functions/notations are used to describe the algorithm:

1. $Context(c)$: Denotes the context information related to client c.

2. $Cache(ap_k)$: Denotes the cache data-structure maintained at ap_k.

3. $Propagate\_Context(ap_i, c, ap_j)$: denotes the propagation of client c's context information from ap_i to ap_j. This can be achieved by sending a *Context-Notify* message from ap_i to ap_j (as discussed

later in section 4.3.1).

4. $Obtain\_Context(ap_{from}, c, ap_{to})$: ap_{to} obtains Context(c) from ap_{from} using IAPP *Move-Notify* message as discussed in section 4.1.

5. $Remove\_Context(ap_{old}, c, ap_{nghbr})$: ap_{old} sends a *Cache-Invalidate* message to ap_{nghbr} in order to remove Context(c) from Cache(ap_{nghbr}).

6. $Insert\_Cache(ap_j, Context(c))$: Insert the context of client, c, in to the cache data-structure at ap_j. Perform an LRU replacement if necessary.

The Proactive Caching Algorithm: The access points use the following algorithm for proactive caching:

At each AP, the cache replacement algorithm used is a *least recently used* (LRU) approach. The cache can be implemented as a hash table over a sorted linked list (according to the insertion time). This would give a cache lookup of $O(1)$ and a cache replacement of $O(1)$ as well. The method $Propagate\_Context$ requires sending the context to each neighbor and hence would incur an execution cost of $O(degree(ap_j) * propagation\_time)$, where *propagation_time* is the round-trip time for communication between the two APs under consid-

Algorithm 1 Proactive Caching Algorithm (ap_j, c, ap_i)

Require: Algorithm executes on AP ap_j, ap_i is the old-AP, c is the client.

1: **if** client c associates to ap_j **then**
2: **for all** $ap_i \in Neighbor(ap_j)$ **do**
3: $Propagate\_Context(ap_j, c, ap_i)$
4: **end for**
5: **end if**
6: **if** client c re-associates to ap_j from ap_k **then**
7: **if** Context(c) not in Cache(ap_j) **then**
8: $Obtain\_Context(ap_k, c, ap_j)$
9: **end if**
10: **for all** $ap_i \in Neighbor(ap_j)$ **do**
11: $Propagate\_Context(ap_j, c, ap_i)$
12: **end for**
13: **end if**
14: **if** client c re-associates to ap_k from ap_j **then**
15: **for all** $ap_i \in Neighbor(ap_j)$ **do**
16: $Remove\_Context(ap_j, c, ap_i)$
17: **end for**
18: **end if**
19: **if** ap_j received Context(c) from ap_i **then**
20: $Insert\_Cache(ap_j, Context(c))$
21: **end if**

eration.

During the initial learning of the neighbor graph, the clients are expected to incur the usual re-association latency. Specifically, the first client to traverse an edge incurs a high handoff latency. But, the edge is added to the graph, and the cost is amortized over subsequent handoffs. Thus after $O(|E|)$ *high latency* handoffs, the algorithm converges to its expected performance. The algorithm has a $O(1)$ running time per re-association.

4.3.1 Modifications to IAPP

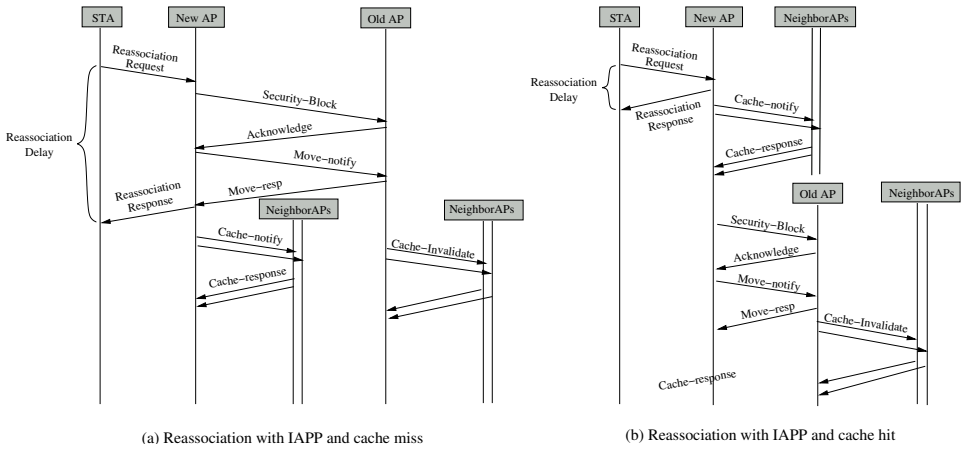


Figure 4.5: Message sequences during a handoff with context caching.

In this section, we discuss the modifications to an early draft of IAPP [27] to incorporate proactive caching using neighbor graphs. The modifications consist of two new messages; *Cache-Notify*, and *Cache-*

Response for the purposes of implementing the *Propagate_Context()* method discussed in Section 4.3. These changes are now included in the IAPP standard [39].

Figure 4.5 shows the modified re-association process (compared to Figure 4.3). For the sake of clarity, the probe and authentication messages are not shown.

1. *Cache-Notify:* This message is sent from an AP to its neighbor and carries the context information pertaining to the client. It is sent following a re-association or an association request.

2. *Cache-Response:* This is sent in order to acknowledge the receipt of *Cache-Notify*. A timeout on this message results in removal of the edge, as the neighbor AP might not be alive.

3. *Cache-Invalidate:* This message is sent from an AP to its neighbor in order to remove the context information from the neighbor's cache. It is sent following a re-association or a disassociation involving an STA leaving the AP.

As can be seen from Figure 4.5, a cache-hit avoids the *Move-Notify* and *Security-Block* communication latency during re-association resulting in a faster handoff.

The knowledge of neighboring APs at each AP is essential for

the effective operation of proactive caching. To avoid the management overhead of manually maintained neighbor graphs, IAPP now includes the algorithms from Section 4.3.

4.4 Performance Analysis

In this section we present an analysis of the proactive caching algorithm based on neighbor graphs. We present two useful properties of the caching algorithm: (i) Firstly, we show that the cache sizes have a upper bound, i.e., with sufficient memory there exists a reasonable cache size which would result in a 100% cache hit,i.e., fast re-association with probability one, (ii) Secondly, we show that the scheme benefits users that are affected by the latency the most, i.e., users that perform higher number of handoffs on average have a higher probability of a cache hit.

Upper bound on the cache size

Assuming there is an upper bound on the number of users associated to any AP, there is an upper bound on the cache size, i.e., the cache would not grow beyond a particular limit. Hence, providing each AP with sufficient memory would guarantee a 100% cache hit ratio. Let $G = (V, E)$ be a neighbor graph. Let $Clientlist(ap_i)$ denote the set of clients associated to ap_i. If client c re-associates from ap_i to ap_j, $Context(c)$

is propagated to $Neighbors(ap_j)$ and removed from $Neighbors(ap_i)$. Hence :

$$Context(c) \in Cache(ap_i) \Longrightarrow c \in \bigcup_{ap_k \in Neighbor(ap_i)} Clientlist(ap_k) \quad (4.1)$$

From Equation 4.1 it follows that :| $Cache(ap_i)$ |$\leq N_{ap_i} * M$ where N_{ap_i} =| $Neighbor(ap_i)$ |= $degree(ap_i)$ and M = maximum number of clients associated to any AP. Summing up Equation 4.1 for all vertices, we get the total memory used by caches over all APs :

$$\sum_{ap_i \in V} \mid Cache(ap_i) \mid \leq M * \sum_{ap_i \in V} N_{ap_i} = M * 2* \mid E \mid \qquad (4.2)$$

Since M is bounded for any AP, the above equation gives an upper bound on the memory requirement at an AP.

Characterizing the Cache Misses

As discussed earlier, the caching algorithm is based on the locality of mobility principle. Since re-association relationships are captured in the neighbor graphs and client-context is forwarded to all neighbor APs, technically we would expect a 100% cache hit ratio for the re-associations. This assumes that the neighbor graph has been learned and the cache size is unlimited (i.e. the cache at each AP is large enough according to Equation 4.1).

The above assumption takes us to the two kinds of cache misses possible during a re-association:

1. *Re-association between non-neighbor APs:* When a re-association occurs between two APs that are not neighbors, the station-context does not get forwarded and results in a cache miss. The edge subsequently is added to the graph through the learning process. Thus when a wireless network is first brought up (or re-booted), the initial re-associations in the network would be cache misses.

2. *Context evicted by LRU replacement:* This happens when the client-context is evicted at the new-AP because of other clients re-associating to neighboring APs.

As discussed in the previous section, the first type of cache miss would occur only once per edge and has a nominal effect towards the performance in the long run. The second type of cache miss depends on mobility of other users, and hence dictates the performance of the algorithm. Presented below is a simplified analysis which calculates the probability distribution of the cache lifetimes of a client's context.

Let the cache size at a particular AP AP_1 be n. Lets divide the time into discrete intervals and assume for simplicity that handoffs are synchronized with these time steps. Lets assume that we are interested

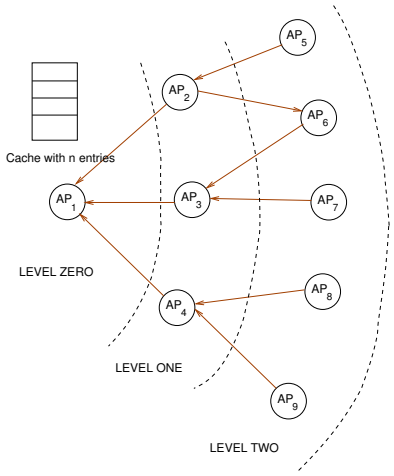


Figure 4.6: Figure shows an AP AP_1 and its immediate (Level 1) and one-hop (Level 2) neighbors with respect to incoming edges.

in the context of a client c which was inserted into the cache of AP_1 at timestep $t = 0$. This is possible only if the client (re-)associated with one of the APs that is one hop away from a neighbor of AP_1, as illustrated in Figure 4.6. For ease of illustration, lets call AP_1 as *Level 0*, neighbors of AP_1 as *Level 1*, and the subsequent neighbors as *Level 2* as shown in Figure 4.6.

Lets assume that any client can perform a handoff with a fixed probability p during any time step. Let M be the number of clients associated to APs at Level 2. Note that implicitly $M \geq n$, else there would be no cache misses at all. These are the potential clients that can handoff to an AP at Level 1, thus having their context inserted into the cache at AP_1. For simplicity of analysis, we assume that the expected

number of clients associated to any set of APs remains constant, i.e., the clients move uniformly at random among APs with constant probability p.

Let $\zeta(t)$ denote the probability distribution of cache lifetimes in discrete timesteps, i.e., the amount of time that a client's context spent in the cache. Say, T timesteps have passed. In timestep i, say x_i clients ($i \in \{0 \dots T\}$) moved to an AP at level 1 thus causing x_i insertions into AP_1's cache. Note that the client c's context will get evicted once n cache insertions have been performed on AP_1. The probability that out of the M clients associated to APs at Level 2, x_i of them performed a handoff given that each client performs a handoff independently with probability p, is given by $\binom{M}{x_i} p^{x_i}(1-p)^{(M-x_i)} = \xi(x_i)$ (say).

Given one particular set of $\langle x_1, x_2, \dots, x_T \rangle$ such that $\sum_{i=1\dots T} x_i = n$ and $x_i \geq 0 \ \ \forall i \in \{1 \dots (T-1)\}$ and $x_T \geq 1$, the probability of that event happening is given by $\Pi_{i=1\dots T}\xi(x_i)$, which simplifies to:

$$\frac{M!}{x_1!x_2! \dots x_T!} p^n p^{(M-n)} \tag{4.3}$$

Thus, the probability that the context gets evicted exactly in T timesteps is given by

$$F(T,n) = \sum_{x_1,x_2,\dots,x_T} \frac{M!}{x_1!x_2! \dots x_T!} p^n p^{(M-n)} \tag{4.4}$$

where $\sum_{x_1,x_2,\ldots,x_T}$ indicates that the sum is carried over all values of $x_i, i = \{1 \ldots T\}$ such that $x_i \geq 0$ and $\sum_{i=1\ldots T} x_i = (T-1)$.

Equation 4.4 can be written as a recurrence relation in the following manner :

$$F(T,n) = \sum_{i=0\ldots n} \frac{F(T-1,n-i)}{i!} \left(\frac{p}{1-p}\right)^i \tag{4.5}$$

The above equation shows that the probability of a cache miss increases rapidly with T. Thus, a faster client (i.e. higher mobility) would spend less time at each AP and hence would have a higher probability of a cache hit. Thus, the performance of the algorithm as perceived by a client would be expected to improve with its mobility. We confirm this theoretical result with extensive simulations and experiments in Section 4.5.

4.5 Experiments and Simulations

We present both simulation and implementation results to demonstrate the performance of proactive caching. Section 4.5.1 discusses the implementation results and the simulation results are presented in section 4.5.2

4.5.1 Experiments

In this section, we discuss the implementation of IAPP with neighbor graphs in a custom wireless testbed. We describe the testbed configuration, the process of the experiments, and the results. In brief, we measured 114 re-associations in the testbed resulting in an average re-association latency of 15.37 ms for a cache-miss without an outlier and 23.58 ms with the outlier (which is the traditional IAPP communication latency) and 1.7 ms for a cache-hit– achieving an order of magnitude improvement in the re-association latency.

The Wireless Testbed

The wireless testbed spans a section of two floors (2nd and 3rd) of an office building. There were five APs on the third floor and four on the second. The geometry of the floors (L-shape and the dimensions) and topology of nine access points are shown in Figure 4.7. The gray circles in the figure represent APs, labeled by an identifier. Three channels, namely 1, 6 and 11 were used by the APs. There were 4 APs on channel 1 and 11 each and one AP on channel 6. These channels were assigned in a fashion to avoid interference with other wireless networks operating in the building resulting in a less than optimal RF design.

The APs used for the experiments were based on a *Soekris* [42]

board NET4521 which has a 133 MHz AMD processor, 64MB SDRAM, two PC-Card/Cardbus slots for wireless adapters and one *Compact-Flash* socket. A 200mW Prism 2.5 based wireless card was used as the AP interface with a 1ft *yagi* antenna. OpenBSD 3.1 with access point functionality was used as the operating system.

The IAPP protocol, neighbor graphs, and the caching algorithm were implemented in the driver (for the wireless interface) along with the AP functionality.

Experiment Process

To preclude possible interference, we shutdown the other wireless networks in the building during the experiments. A *mobile unit* consisting of a client laptop, and a sniffer was used in the experiments. A laptop with Pentium III 750 MHz CPU and 256 MB RAM and a Prism 2.5 based *ZoomAir* wireless card was used as the client. The re-association latencies were measured by capturing management frames on channels 1, 6 and 11. This was done by the sniffer which had a wireless card dedicated to capturing traffic on each channel (1, 6, and 11). Since the APs were configured only on the above three channels, it was guaranteed that the sniffer would capture all management frames destined to or transmitted by an AP in the testbed (with respect to the STA)

(primarily *re-association request* and *response* frames). Three wireless interfaces in two laptops constituted the sniffer.

Two experiments were conducted. The first experiment was conducted with *fresh* APs, i.e. there were no neighbor relationships prior to the start of the experiment. The goal of this experiment was to study the effect of the learning process on the re-association latencies with time. The second experiment (following the first) was to confirm guaranteed cache hits once the neighbor graph had been learnt by the APs. We discuss the detailed setup of each experiment below.

Experiment A: The first experiment consisted of a random walk with the mobile unit, through the physical span of the testbed. There were no neighbor relationships existing among APs prior to the start of the experiment. The experiment started with the client associating to AP-2 (refer Figure 4.7), and a random path of motion covered all APs on third floor. The unit then moved to the second floor, covered all APs, and returned to the initial point of association (AP-2). This was one round of the experiment and nine rounds were conducted for statistical confidence in the measurements. This resulted in one association, and 114 re-associations during the entire experiment.

Experiment B: The second experiment, followed the first, consisted of two short rounds using a different client. The purpose of this

experiment was to verify the existence of neighbor graphs (i.e learned from the first experiment) at each AP by observing a cache hit on all re-associations.

Experiment Results

Figure 4.7 depicts the (3D) neighbor graph created during the experiment. The graph was constructed by observing the *re-association request* frames captured by the sniffer. The directed edges indicate the direction of the re-association (from the old-AP to the new-AP). The solid edges are intra-floor edges and the rest are inter-floor edges. The graph shows 23 distinct pairs of APs, between which the STA could re-associate.

Experiment A: Figure 4.8 shows the re-association latencies at each AP [1]. The Y-axis is the latency in logarithmic scale. The *circular* points represent re-association with a cache-miss and *cross* points are the cache-hits. Most of cache-miss latencies reside around 16 ms except an outlier of 81 ms at AP-8. The cache-hit latencies are clustered around an average of 1.69 ms. There are a few cache-hits with latencies more than 4 ms. We reason that these outliers (involved with AP-4 and 5) are due to poor coverage design with respect to the building

[1]The re-association latency is attributed to the new-AP

topology. AP-4 and AP-5 had a relatively small transmission range when compared to other APs and they were physically close to each other. Since they were the only APs covering a large area, the re-association latencies were effected by packet errors/retransmits. There was another extreme outlier of 2.36 seconds with a cache-hit caused by a sniffing error. This value was excluded from the analysis.

Figure 4.9 shows the re-association latencies observed over time. During the experiment, there was a cache-miss for the first re-association to each AP (except AP-2) as the neighbor graph was built. Figure 4.9 clearly shows how context caching decreases re-association latencies with time. Except the very first re-associations and a few outliers, most re-association latencies lie below 2 ms. In total, there were 8 cache-misses with average of 15.37 ms [2] and 105 cache-hits with average latency of 1.69 ms.

Experiment B: The second experiment, was done with a different client. The APs had learnt the neighbor graph, and hence during the experiment there were no cache misses. Each association/re-association forwarded the context to the neighbors, and hence the client's context

[2]The outlier of 81 ms has been excluded from the average calculation. We eliminated it since it would unfairly distort our result by making it higher, *i.e.* better, than what is clearly the average of 16 ms.

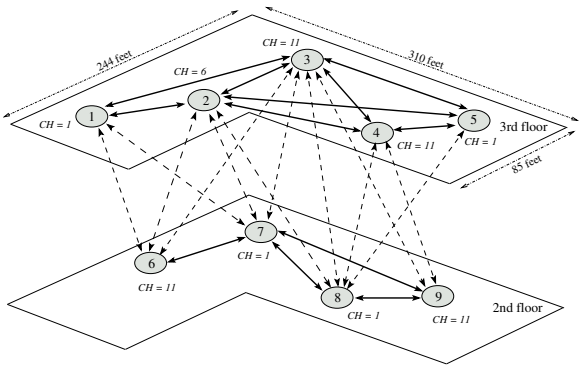


Figure 4.7: Experiment Environment and the Neighbor Graph.

was always found in cache during a re-association [3]. This experiment had 18 re-association, all cache hits, resulting in an average latency of 1.5 ms.

Thus the experiment results show that proactive caching with neighbor graphs reduces the re-association latency by an order of magnitude.

4.5.2 Simulations

Access points, unlike cellular base-stations, are embedded systems with limited resources (computing power and memory) as vendors attempt to lower their costs. A typical access point has around 4MB of RAM and 1 MB of flash. Client context information could potentially consist of security credentials, QoS information etc. Thus an AP can store only

[3]Since we had only one client in the experiment, there were no cache evictions.

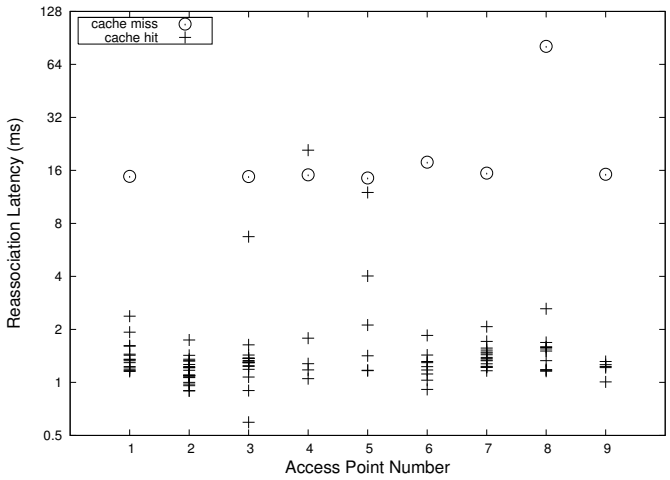


Figure 4.8: Re-association latencies at each access point.

a limited number of contexts in its cache (LRU cache replacement). In this section, we present results on how the algorithm performs while varying the mobility, the number of clients and the number of APs in the network.

Simulation Objectives:

1. To observe the effect of cache size, number of clients and the mobility of clients on the cache hit ratio.

2. To observe the performance of caching with various neighbor graphs.

Each simulation starts with a set of APs, a neighbor graph structure connecting them, a set of clients and their initial distribution on

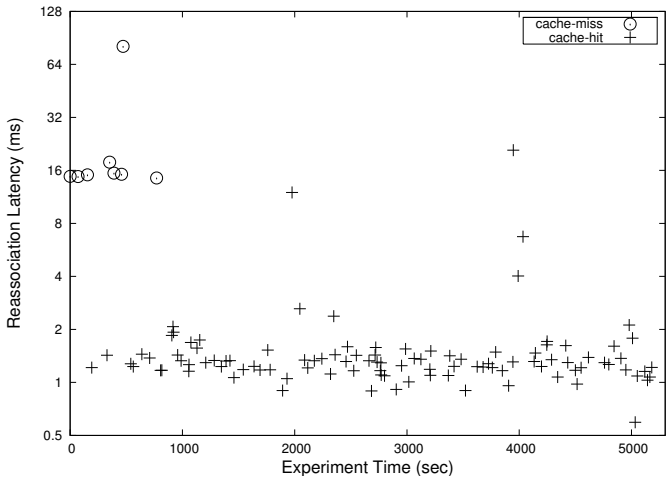


Figure 4.9: Re-association Latencies with Time.

the APs. Each client is assigned a *mobility index* (defined later), which dictates the mobility of the client throughout the simulation. The assumptions and the model we used are:

Simulation Model and Assumptions:

1. *AP Neighbor Graph does not change during the simulation:* As noted earlier, changes in the AP neighbor graph would cost (in the worst case), one high latency handoff per edge, and has a nominal effect on the overall cache performance.

2. *Correctness and completeness of the Neighbor Graphs:* We are assuming that the neighbor graphs are correct and complete, i.e. the simulations do not consider any re-associations which are not cov-

ered as edges in the graph [4]. This makes it sufficient to simulate re-associations according to the neighbor graph without maintaining any correspondence with the physical placement of APs (that would produce the neighbor graph).

3. *Initial User-AP distribution:* We have assumed a uniform distribution of clients across APs to at the start of the simulation. Figure 4.10 shows the distribution of the maximum number of users associated to each AP during a simulation with 100 APs, and 500 clients.

4. *Roaming Model:* The client roams according to the following model:

   (a) Let client c have an association pattern $\Gamma(c) = \{(ap_1, t_1), (ap_2, t_2), \ldots, (ap_n, t_n)\}$. The client c is said to roam from ap_1 to ap_n if (i) the time associated at each $ap_i, (1 < i < n) : t_{i+1} - t_i$ is of the order of a typical re-association latency (around 100 ms, [36]) and (ii) the time the client spends on ap_1 and ap_n is of the order of a typical client session [35]. Thus the client stays for a session-

[4] As discussed earlier, such re-associations would have resulted in the edge being added to the graph

duration with an AP, roams to another AP (according to an association pattern), and stays for another session.

(b) At any given point of time during the simulation, the client is either *roaming* (according to definition above) or staying associated to its current AP.

(c) The association pattern of a roam is decided randomly: If the client c is associated to ap_i, it can move to any one of its neighbors $(ap_{i_1}, ap_{i_2}, \ldots, ap_{i_k})$ with equal probability.

5. *User Mobility:* Define *mobility index* of a client as the probability that the client is *roaming* at any given point of time during the simulation. At the end of the simulation it converges to the (Total time spent in roaming/Total simulation time). Mobility indices are assigned to clients on a scale of $1 \ldots 100$. The distribution of mobility indices on clients is uniform.

Simulation Environment

1. The simulation uses random and connected neighbor graphs with 10, 20, 50 and 100 vertices.

2. *Duration of the Simulation:* The simulation runs for one million re-association events uniformly distributed over the users accord-

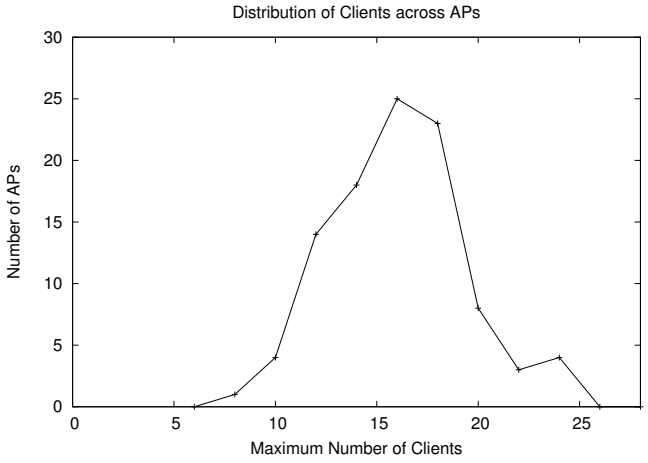


Figure 4.10: Distribution of Maximum number of clients associated to an AP during a simulation with 100 APs and 500 users.

ing to their mobility indices. This makes the duration of the simulation large enough for statistical confidence in the results.

Simulation Results

1. *Mobility Improves Proactive Caching Performance:* Figure 4.11 shows the cache hit ratio achieved by clients according to their mobility index. The figure compares the hit ratio performance over neighbor graphs of size 10, 20, 100 vertices keeping the cache size constant. In all three curves, the hit ratio increases with client mobility as previously discussed in Section 4.4. The relative improvement diminishes with increasing number of vertices in the NG graph, and the prime reason for this being the constant cache

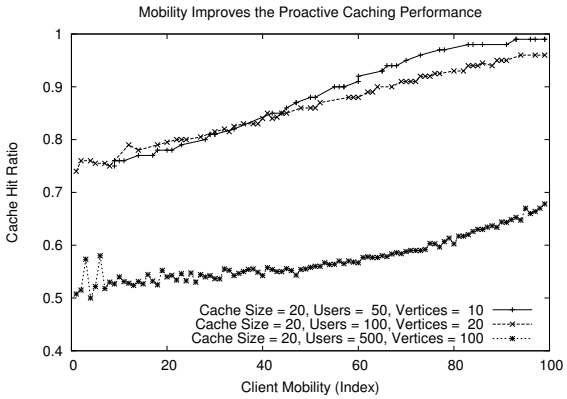


Figure 4.11: Plot of clients mobility and the cache hit ratio achieved.

size. Later plots elucidate this observation.

2. *Effect of Cache Size and Client Mobility on Hit Ratio:* Figure 4.12 shows the effect of cache size on the hit ratio keeping the number of clients, and the NG graph the same. The graph has 100 vertices, and 200 users. Clearly an increase in the cache size has a direct impact on the cache hit ratio, to the extent that for a cache size of 40 (or 20% of the number of users), all clients have a hit-ratio of 98% or better.

3. *Effect of Cache Size and Number of Users on Hit Ratio:* The number of clients in the network has a direct impact on the performance. Figure 4.13 shows the effect of the two parameters on hit ratio. Figure 4.14 shows the effect of the cache size as a percentage of the number of users on the hit ratio. The data points

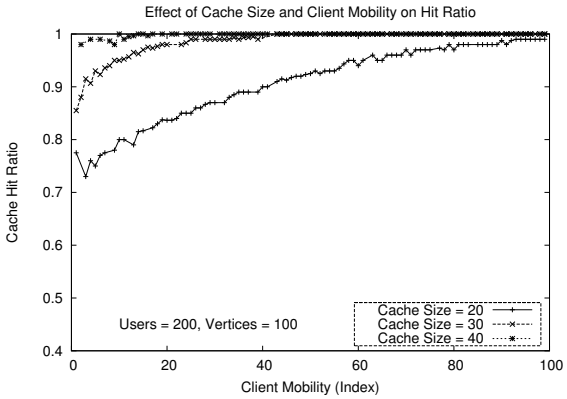


Figure 4.12: Effect of Cache Size and Client Mobility on Hit Ratio.

were taken for cache sizes varying from 20 to 50 and the number of users varying from 200 to 500 in increments of 100. Thus, a 15 percent cache size is sufficient for a hit ratio of 98 % while a cache size of 20 percent gives a hit ratio of 100 %.

4.6 Related Work

The related work is broken into two distinct categories: context transfers, and algorithms that dynamically generate the topology of wireless networks.

The previous work on context transfers has mostly focused on the IP layer using *reactive* transfer mechanisms [37], and general purpose transfer mechanisms without detailing transfer triggers [43]. The only previous work on link layer context caching was also originally *reactive*

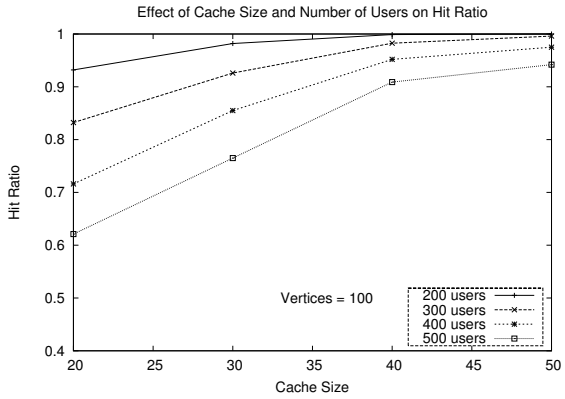


Figure 4.13: Effect of Cache Size and Number of Users on Hit Ratio.

until neighbor graphs were recently added [44].

The IP layer context transfer mechanisms focus solely on the transfer of context from access router to access router, and while Koodli [37] mentions access points briefly– indicating that access routers and access points can be co-located. The context transfer mechanisms are designed solely for access routers and are reactive rather than pro-active as in neighbor graphs [37]. In the case of the SEAMOBY context transfer protocol, the protocol provides a generic framework for either reactive or pro-active context transfers [43]. The framework, however, does not define methods for implementing either reactive or pro-active context transfers. As a result, our approach can easily be integrated into the SEAMOBY protocol providing a pro-active context transfer mechanism as it was with IAPP.

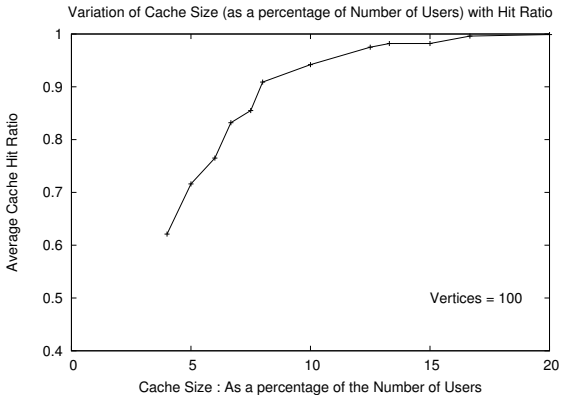


Figure 4.14: Variation of Cache Size (as a percentage of Number of Users) with Hit Ratio.

The previous work on topology algorithms has focused on pre-authentication, automated bridge learning, and sharing of public key certificates [45, 46, 47].

Pack proposes pre-authentication be performed to the k most likely next access points. The k stations are selected using a weighted matrix representing the likelihood (based on the analysis of past network behavior) that a station, associated to AP_i, will move to AP_j. The mobile station may select only the most likely next access points to pre-authenticate, or it may select all of the potential next access points [45, 46]. Pack uses the notion of a frequent handoff region (FHR) to represent the adjacent access points, or neighbors, which is obtained by examining the weighted matrix. The weights within the matrix

are based on an $O(n^2)$ analysis of authentication server's log information using the inverse of the ratio of the number of handoffs from AP_i to AP_j to the time spent by the mobile station at AP_i prior to the handoff. While the FHR notion represents neighboring access points, it requires $O(n^2)$ computation and space, where n is the number of access points in the network, and must be created at the authentication server. Furthermore, the FHR notion does not quickly adapt to changes in the network topology. This is in contrast to neighbor graphs which require $O(degree(ap))$ computation and storage space per AP[5] and which quickly adapts to changes in the network topology. Additionally, neighbor graphs can be utilized either in a distributed fashion at each access point, or client, and in a centralized fashion at the authentication server.

Capkun et. al. leverage station mobility to create an *ad-hoc* public key infrastructure by neighboring stations exchanging public key certificates to create a certificate graph [47]. The idea is that a neighboring station can most likely verify the identity of another station, and after successfully doing so add the certificate to their graph. The resultant graph represents the mobility pattern with respect to other stations. While this mobility graph has a different focus and use than

[5]The cache consumes an $O(1)$ storage and computation.

neighbor graphs, it none-the-less uses the notion of neighbors, and we include a discussion of it for completeness.

In the 1980's to overcome the geographic limitation of a LAN, LANs were connected using *bridges*. In this approach, a bridge connecting two or more links listens promiscuously to all packets and forwards them to a link on which the destination station is known to reside. A bridge also dynamically learns the locations of stations so that it can forward traffic to the correct link. In [48], Perlman proposed a *self-configuring* and *distributed* algorithm to allow bridges to learn the loop-free subset of the topology that connects all LANs, by communicating with other bridges. This subset is required to be loop-free (a spanning tree) to avoid unnecessary congestion caused by infinitely circulating packets. This *Spanning Tree Algorithm / Protocol* [49] is self-configuring because the only apriori information necessary in a bridge is its own unique ID (MAC address). The algorithm requires a very small bounded amount of memory per bridge, and a bounded amount of communications bandwidth for each LAN. Furthermore, there is no requirement for modifications to stations and the algorithm inter-operates with older bridges. Neighbor graphs are also self-configuring and operate in the same manner– examining network traffic, specifically layer 2 management frames or AAA messages, to

create the wireless network topology dynamically. The two algorithms, and their purposes are different however.

4.7 Summary

In this chapter, we have performed an empirical evaluation of neighbor graphs, which captures the locality in mobility of the users by autonomously monitoring the handoffs. This structure abstracts the physical topology of the network into a neighbor relationship which can be used as a vehicle for numerous applications. Neighbor graphs add valuable structure to the distribution system (DS) interconnecting the APs forming the wireless network. This structure, which provides information about the locality among the APs (see Chapter 2), can be leveraged for optimizations on existing algorithms (load balancing, network management, and key pre-distribution) and may lay the foundation for other interesting and novel applications.

As an application for neighbor graphs, we implemented and studied the performance of the *proactive caching algorithm* for faster wireless handoffs. The caching algorithm uses neighbor graphs to send station-context to its neighbors prior to the handoff and hence separates the context transfer process from re-association. We have implemented the approach using an early version of IAPP [40] running on a dedicated

wireless testbed and presented results from experiments conducted on the testbed and as a result of our early experiments proactive caching using neighbor graphs has been added to the final version of IAPP [44].

In our experiments, 114 re-associations occurred with an average re-association latency of 23.58 ms (including the one outlier) and 15.37 ms (without the outlier) for a cache-miss (traditional handoff), and 1.69 ms for a cache-hit, an order of magnitude improvement due to proactive caching. In our simulations, we studied the performance of the algorithm under varying network characteristics : user mobility, the number of users associated to the network, and the number of APs forming the network. We conclude that the performance of the algorithm (hit-ratio) improves as the user mobility increases eventually reaching a 100% hit-ratio under certain network configurations. As expected, we find that the cache size plays an important role in the performance of the algorithm and that a cache size of 15% (of the number of users associated to the network) gives a minimum cache hit-ratio of 98%.

This empirical study has shown the validity of the locality principle as the hypothesis of this dissertation discussed in Chapter 1 and 2. We had performed an analytical verification of this principle in Chapter 2. More importantly, the results of this chapter support the hypothesis

using a real world optimization called proactive caching. The strong success of proactive caching in reducing the context transfer latency as shown through the implementation and simulations stems from the locality present in user mobility patterns.

In the next chapter, we leverage neighbor graphs to devise fast re-authentication schemes that bypass the full-authentication specified by the IEEE 802.11i standard and maintain the same security properties. Later in Chapter 6, we extend neighbor graphs to include edges indicating certain types of overlap in coverage areas of APs, called *Overlap Graphs*. We devise fast scanning algorithms that use such overlap graphs to reduce the scanning latency component of 802.11 handoffs.

Chapter 5

The Proactive Key Distribution Technique

Treat your password like your toothbrush. Don't let anybody else use it, and get a new one every six months. - Clifford Stoll.

In the previous chapter, we studied how locality in user mobility as captured by a neighbor graph could speed up the handoff latency by eliminating the costly context transfer process. We studied a proactive caching algorithm which placed a moving station's context onto a candidate set of APs determined using the locality information. In the previous chapters, we have not considered the role of security in the handoff process. The IEEE 802.11i standard defines a new security architecture called the Robust Security Network Architecture (RSNA), which addresses the ailments suffered by the earlier Wired Equivalent

Privacy (WEP) protocol and provides robust per-packet confidentiality and integrity along with authentication [1].

In this chapter, we study how security and locality in user mobility are inter-coupled. We show how valuable insights and improvements can be gained by applying our locality hypothesis (see Chapters 1 and 2) to the problem of securing handoffs. The current mechanisms require a full authentication upon each handoff. We show that this is wasteful by taking advantage of the locality to develop fast re-authentication schemes which provide the same security guarantees as a full authentication.

Current Authentication Methods

Figure 5.1 shows the network architecture typically used for a wireless LAN. The IEEE 802.11i standard uses the IEEE 802.1X [5] framework for authentication which is based on a three-party model: the supplicant, which requires access; the authenticator, which grants access; and the authentication server, which gives permission. In the 802.11 space, a client is the supplicant, which is authenticated by an AP (the authenticator) to a central AAA (Authentication, Authorization and

[1]For a summary of the problems associated with WEP and how the IEEE 802.11i standard addresses these, the interested reader is referred to [50].

Accounting) server (the authentication server). This authentication process establishes the association mapping between the AP and the client in a secure manner. It also results in creation of key material for per-packet encryption and authentication between the AP and the client. Further details of this process are discussed in Section 5.1.

The IEEE 802.1X standard provides a state machine based framework for the functioning of the supplicant, the authenticator, and the authentication server. The actual communication between these entities takes place through a standardized protocol called the Extensible Authentication Protocol (EAP). EAP, specified as RFC 2716, provides a flexible vehicle to carry authentication traffic independent of the authentication method used. Various authentication methods such as password-based (CHAP), Transport Layer Security (TLS), Tunelled TLS, Kerberos, etc., are supported by EAP as specific authentication *types*. The 802.1X/EAP standards find applicability to a wide variety of networking technologies including wireless and wired LANs, dial-up, Virtual Private Networks (VPNs) and token ring networks. When applied to the wireless domain, authentication based on Transport Layer Security (TLS) (a Secure Socket Layer technology [7]) is the most commonly used. The authentication process in TLS is based on a Server-certificate and hence provides for a robust asymmetric authentication

method.

Because of the active involvement of APs in the security process, the IEEE 802.11i standard mandates a fresh full-authentication when the client roams to a different AP. Thus, from a security perspective there is no difference between a first-association and a handoff (re-association). From a performance perspective, the secure key material generated during a full-authentication is completely discarded as the user roams to a different AP. In order to evaluate this penalty, we constructed a testbed based implementation of the 802.1X setup. As a part of the open-source *Open1x* effort[2], the IEEE 802.1X authentication mechanisms were implemented as client (called *Xsupplicant*) and as an authenticator. The TLS mechanism was implemented as the authentication method. Based on this, we performed experiments to measure the latency incurred by a full 802.1X authentication. We measure this latency to be $800ms$ on average which is a significant additional cost to the handoff latency.

Our Contributions

In this chapter, we design a fast re-authentication/key distribution scheme based on neighbor graphs which takes advantage of the key

[2]See http://www.open1x.org

material generated during the first authentication to perform fast re-authentication at neighboring APs. This scheme takes advantage of the locality present in user mobility (Chapters 1 and 2) by computing and exposing this key material only to a limited set of APs which characterize the local neighborhood into which a user might potentially move. This neighborhood is bounded as shown analytically in Chapter 1 and as a result the computation costs for this scheme are bounded as well. The key distribution scheme provides the same security guarantees as a full TLS authentication in a small fraction of the time. We discuss this scheme in detail and evaluate it through our testbed based implementation.

The rest of this chapter is organized as follows. Section 5.1 provides a brief overview of the IEEE 802.11 Security Architecture (the IEEE 802.11I standard). Section 5.2 discusses the proactive key distribution method using neighbor graphs. We present implementation results from an in-building wireless network testbed in Section 5.3. A detailed description of related techniques to reduce the authentication latency, a discussion on proactive key distribution is provided in section 5.4. Finally we summarize the contributions of this chapter in Section 5.5.

5.1 IEEE 802.11i Authentication Overview

The authentication framework developed by the IEEE Task Group I – Security (TGi) is a complex combination of several different protocols. While a thorough understanding of each of these protocols is not required, basic knowledge of each will assist in understanding the problems we are addressing as well as our solution.

Figure 5.1 shows the architecture of a wireless LAN. The APs are typically connected together to a backend authentication server (AAA server such as RADIUS) over an ethernet (or VLAN). As in any architecture, the trust assumptions are key to the correct operation of the system. TGi makes the following trust assumptions:

- The AAA server is trusted.
- The access point to which a mobile station is associated is trusted– Non-associated AP's are not trusted.

These assumptions, which are different from those in a cellular network, are due to the nature of 802.11 equipment. Access points are low cost devices that are often placed in locations which lack proper physical security. Therefore, it is important to prevent the compromise of a single AP permitting a compromise of the entire network.

5.1.1 IEEE 802.1X

The IEEE 802.1X [5] standard provides an architectural framework to facilitate network access control at the link layer for various link technologies (IEEE 802.11, FDDI, Token Ring, IEEE 802.3 Ethernet, etc.). The standard abstracts the notion of three entities: the *supplicant*, the *authenticator* or the network port, and the *authentication* server. Figure 5.2 shows the typical communication setup. A *supplicant* is an entity that desires to use a service (link layer connectivity) offered via the notion of a *port* on the *authenticator* (such as a switch or an access point). Thus for a single network there will be many ports through which supplicants can authenticate themselves and obtain network access. An *authenticator* is in control of a set of ports, and a network might have multiple authenticators. As an example, an ethernet switch can be an authenticator, which controls network access on multiple physical ethernet ports available on the device. In the IEEE 802.11 scenario, a port corresponds to an association between a supplicant and the authenticator (access point).

The supplicant authenticates via the authenticator to a central *authentication server* which directs the authenticator to provide access after successful authentication. Typically the authentication server and the authenticator communicate using the *Remote Authentication Dial-*

In User Service (RADIUS) protocol. The RADIUS protocol contains mechanisms for per-packet authenticity and integrity verification between the AP and the RADIUS server– although these measures are not as strong as desired.

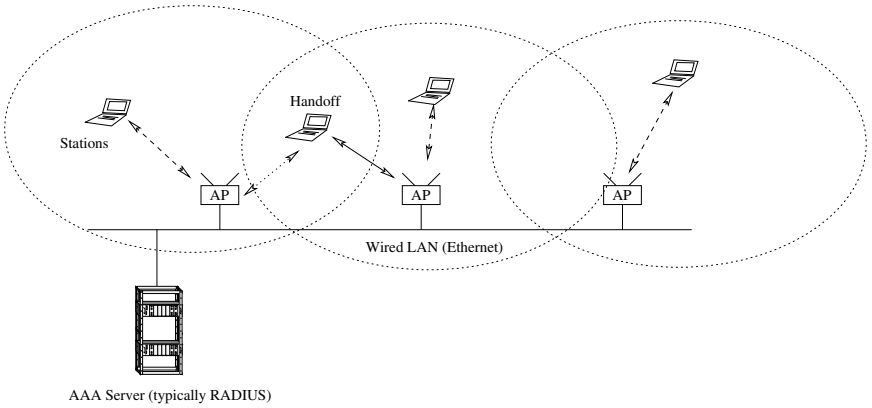


Figure 5.1: Typical topology of a wireless LAN.

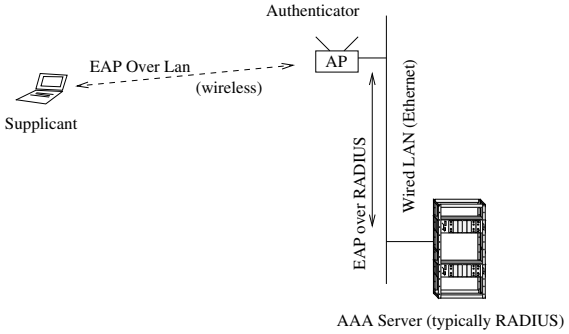


Figure 5.2: The entities in an IEEE 802.1X setup.

The authentication process between the authentication server and the supplicant (via the authenticator) is carried over an Extensible

Authentication Protocol (EAP), which is described in the following section.

5.1.2 Extensible Authentication Protocol

The IEEE 802.1X standard employs the *Extensible Authentication Protocol* to permit a variety of authentication mechanisms. Figure 5.3 shows the protocol layers for communication between the supplicant and the authenticator. EAP is built around the *challenge-response* communication paradigm. There are four types of messages: EAP *Request*, EAP *Response*, EAP *Success* and EAP *Failure*. The EAP Request message is sent to the supplicant indicating a challenge, and the supplicant replies using the EAP Response message. After multiple exchanges of the Request/Response messages the EAP Success/Failure message is used to notify the supplicant of the outcome. A multitude of authentication methods can be encapsulated in the EAP protocol – most notably, EAP-TLS, EAP-MD5, EAP-AKA, EAP-SIM, etc. We discuss the TLS authentication method in further detail in the next Section.

The EAP messages do not have an addressing mechanism and are encapsulated over an *EAP Over Lan* (EAPOL [5]) protocol between the supplicant and the authenticator and are carried as a RADIUS attribute

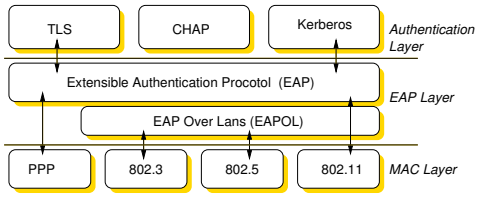


Figure 5.3: The EAP stack

between the authenticator and the authentication server. The EAPOL protocol also provides for a four-way handshake mechanism (discussed later).

5.1.3 Transport Layer Security

The Transport Layer Security protocol as described in RFC-2246, provides strong authentication and encryption at the transport level. It is divided into two protocols : the *handshake* protocol which handles the communication for the authentication and derives strong key material for the data transfer which is carried over the *record* protocol. The authentication part of the TLS has been exported as an authentication mechanism over EAP in the EAP-TLS RFC2716. This is the most commonly used authentication mechanism over EAP within 802.11 based networks, and fits into the IEEE 802.1X model. Figure 5.9 shows the complete set of messages exchanged during a full EAP-TLS authentication.

In the application of TLS to IEEE 802.1X, the supplicant and the authentication server have a certificate from a common trusted certificate authority (CA). The mutual authentication process based on these credentials achieves the following: (i) mutual authentication of the client and the server, (ii) a strong shared secret master key (MK) (iii) an initialized sets of pseudo-random functions (PRFs) which can be utilized for generating further key material. Let TLS-PRF denote the PRFs generated as a result of the authentication. The MK is used to derive a Pairwise Master Key (PMK) by using equation 5.1.

$$PMK = TLS\text{-}PRF(\ MK,\ clientHello.random\ |\ serverHello.random) \tag{5.1}$$

The PMK is used along with certain cipher methods to derive four Pairwise Transient Keys which are used for various purposes as shown in Figure 5.4 [10]. The first key EAPOL-MIC key and the EAPOL-Encr. keys are used to provide data origin authenticity and confidentiality for the four-way handshake discussed later. The other two keys are used for link layer encryption and authenticity depending on the cipher suite being employed.

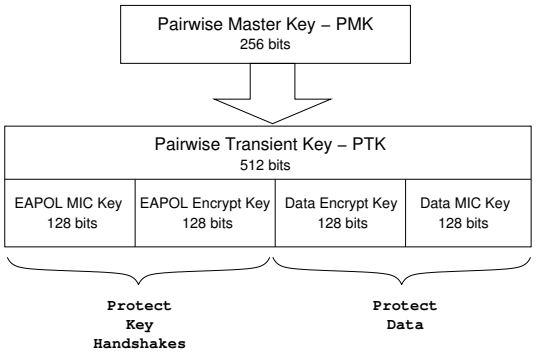


Figure 5.4: The key structure: PMK and the derived PTK.

5.1.4 Four way hand-shake

The IEEE 802.11 Task Group I defines an IEEE 802.1X protocol called a four-way handshake. This protocol is used to confirm the liveness of the AP and the station(STA), guarantees the freshness and synchronizes the shared session key and binds the PMK to the MAC address of the STA. The communication is carried using EAPOL key messages[4].

1. *Message (A) Authenticator* $\longrightarrow$ *Supplicant*: This is the first EAPOL-Key message and is sent from the authenticator to the supplicant. It contains ANonce – a nonce value generated by the authenticator. Once the supplicant has received this message it can compute the four temporal keys.

2. *Message (B) Supplicant* $\longrightarrow$ *Authenticator*: This message contains SNonce – a supplicant generated nonce and a MIC over the message to protect its integrity. The authenticator uses SNonce

to generate the temporal keys, and verifies the MIC.

3. *Message (C) Authenticator* $\longrightarrow$ *Supplicant*: This message includes the earlier ANonce and a MIC check which can be verified by the supplicant proving that the authenticator has a matching PMK.

4. *Message (D) Supplicant* $\longrightarrow$ *Authenticator*: This message signifies the completion of the four-way handshake and signals the installation of the keys by both entities for the data communication.

The four-way handshake protocol is used during a full-authentication and during re-authentication, and hence this cost (i.e. the overhead incurred) will be present in both situations. We also do not include the cost of the hand-shake in the timings of EAP-TLS. In this work, we do not implement the handshake for the above reason, instead we have implemented a simpler two-way handshake mechanism for demonstration purposes.

5.1.5 TGi Trust Relationships

One of the interesting, and disappointing, problems with TGi's new 802.11 security architecture are the trust relationships in an operational

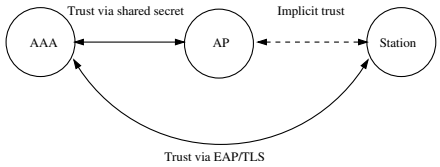


Figure 5.5: The Trust relations in TGi.

network. Many people believe that the access point is a trusted party, and this isn't completely correct.

Figure 5.5 depicts the trust relationships within TGi. The solid arrows represent an explicit mutual trust relationship while the dotted line represents an implicit trust relationship that MUST be created in order to make security claims about the communications path. This trust relationship between the AP and the STA is transitive and derived from the fact that the station trusts the AAA server and the AAA server trusts the AP. This, unfortunately, is not ideal since in many cases the trust relationship between the AAA server and the AP will not exist if shared keys are not used to protect the RADIUS traffic. However, the majority of the AP vendors in TGi had a strong desire for an inexpensive AP which was more of a relay than a participant in the communications.

5.1.6 Properties of a Successful Authentication

After the successful completion of the EAP-TLS authentication phase the following statements hold:

1. The mobile station's identity has been proved.

2. Based on the above identity, the mobile station's access to the network has been granted by the AAA server.

3. The mobile station and the AAA server share a strong master secret, MK.

4. The mobile station, the AAA server, and the associated access point all share a common secret, pairwise master key or PMK, derived from the MK.

5. A session key, PTK, is derived from the PMK using the four-way handshake and is only shared between the mobile station and the associated access point.

5.2 Pro-active Key Distribution

Pro-active key distribution seeks to reduce the latency of the authentication phase by pre-distributing key material ahead of a mobile station. Our approach provides all of the same properties of a full EAP-TLS

authentication, but at significantly less cost in terms of latency and computational power of the mobile station. In this section, we assume that the neighbor graph mechanisms have been implemented by the wireless network in a distributed or centralized fashion. The details of neighbor graphs, their construction and maintenance are discussed in Chapter 4.

5.2.1 PMK Trees

In the current, 802.11i framework the PMK is derived from the MK by Equation 5.1. Pre-distributing this PMK, which is permitted in the current IEEE 802.11 TGi draft as PMK caching, violates the TGi trust assumptions. Rather than pre-distribute this PMK, we change the derivation of the PMK to the recurrence shown in Equation 5.2, where n represents the n^{th} re-association for $n >= 0$.

$$\begin{aligned} PMK_0 &= \textit{TLS-PRF(MK, clientHello.random} \mid \\ &\quad \textit{serverHello.random)} \\ PMK_n &= \textit{TLS-PRF(MK, } PMK_{n-1} \mid \textit{ AP\_MAC} \\ &\quad \mid \textit{ STA\_MAC)} \end{aligned} \tag{5.2}$$

The recurrence shown in equation creates a PMK tree with the re-association pattern, $\Gamma(STA)$, a path within the tree as shown in

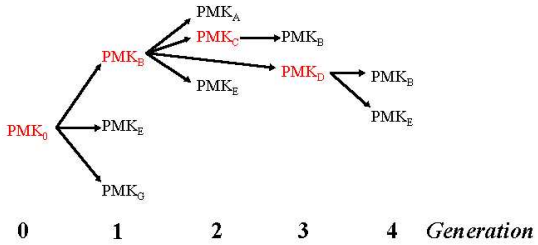


Figure 5.6: PMK tree

Figure 5.6. In Figure 5.6, the re-association pattern is $\Gamma(STA) = A, B, C, D$.

5.2.2 PMK Synchronization

There are two conditions that can exist when a mobile station arrives at an access point with respect to the pre-distribution of the correct PMK: either the AP and the mobile station share the same PMK, or they do not. The handshake (two-way in our case and four-way in the case of TGi) determines which of these cases exist. This also ensures both *liveness* and *freshness* of the key.

5.2.3 PMK Distribution

Once a mobile station completes an initial full EAP-TLS authentication as denoted by the AAA server sending an *ACCESS-ACCEPT* message to the access point indicating successful completion of the authenti-

cation process as well as PMK_0. At this point, the AAA server and the mobile station share the MK, and the AAA server, the access point, and the mobile station all share PMK_0. The AAA server now determines the neighbors of the associated access point and sends a *NOTIFY-REQUEST* that a specific mobile station may roam into the coverage area of each of the neighboring access points. This message is advisory only, and an access point may or may not decide to request the security association, or PMK from the AAA server at this time. If the AP does decide to request the PMK, then the AP sends a *NOTIFY-ACCEPT* message. If not, then the AP sends a *NOTIFY-REJECT* message to the AAA server. Upon receiving the *NOTIFY-ACCEPT* message, the AAA server responds with an *ACCESS-ACCEPT* message which contains the appropriate PMK as well as authorization for the mobile station to remain connected to the network. As a note, RADIUS messages have been designed around a challenge response paradigm and technically we violate it here by introducing the *NOTIFY-REQUEST* message. The IETF is working on DIAMETER (a enhanced AAA server backward compatible to RADIUS), which includes the kind of messages we discussed above and our approach fits nicely into DIAMETER.

5.2.4 Two-way handshake

After the key distribution, the four-way handshake (discussed earlier) confirms the freshness of the keys being used by the AP and the roaming STA. In our implementation, we used a simpler two-way handshake (an EAPOL start message, and an EAP-Success message if the AP has the correct key) for purposes of demonstration. Since the four-way handshake is performed during both – a full authentication and the fast re-authentication, it does not effect the key distribution scheme.

5.3 Implementation

In this section, we present implementation results to demonstrate the performance of the pro-active key distribution scheme. We have implemented the pro-active key distribution method and the standard full-authentication over an in-building wireless testbed network comprised of 9 access points spread over three floors. Since the four-way handshake process appears in both schemes after the key has been delivered, we did not implement the full handshake and we instead implemented a simple 2-way handshake to verify the key freshness. We measured 90 full EAP-TLS authentication latencies with an average of approx. 1.1 seconds. Using the pro-active key distribution scheme

for fast re-authentication we obtained an average latency of 25 ms (a 99.6% reduction).

We also measured the overhead on the wired distribution system added by the two additional messages between the RADIUS server and the authenticator. With eight neighbors to distribute the key, the overhead was approx. 21 ms on average. Note that this overhead plays no role in the re-authentication latency, and just adds to the load on the RADIUS server and the distribution system. We have included it for completeness.

5.3.1 The Testbed

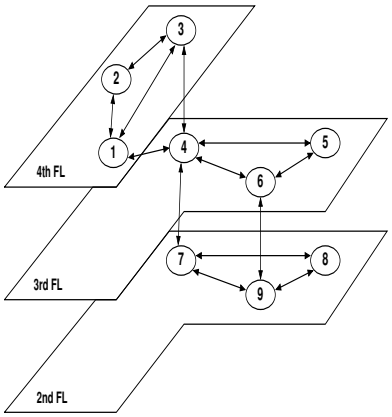


Figure 5.7: Figure shows the topological placement of the APs in our wireless testbed and the resulting structure of the neighbor graph.

The wireless testbed network spans three floors (2nd, 3rd and 4th)

of a university building and consists of nine APs as shown in Figure 5.7. The access point is based on a NET4521 *Soekris* board, which has a 133 MHz AMD processor, 64MB SDRAM, two PC-Card/Cardbus slots for wireless adapters and one *CompactFlash* socket. The board is powered using *Power Over Ethernet* through the ethernet cable. A 200mW Prism 2.5 based wireless card is used as the AP interface with a 1ft *yagi* antenna. OpenBSD 3.3 with access point functionality is used as the operating system.

The supplicant and the authenticator software is based on the *open1x* implementation built at the University of Maryland, College Park (http://www.open1x.org). We also use the *Freeradius* software for the RADIUS server, modified to implement the key distribution scheme and maintain the neighbor graph data structure. The RADIUS server is installed on a backend machine (PIII 551.247 MHz, 128 MB RAM). The *Xsupplicant* and the *authenticator* software was modified to include the simple two-way handshake instead of the four-way handshake for purposes of demonstration.

5.3.2 Results

The experimental setup consisted of a supplicant roaming in the wireless testbed. A laptop with PIII 1.8 GHz, 256 MB RAM and a Prism

2.5 based *DemarcTech* wireless card is used as the supplicant. Three experiments were done to measure three different latencies as detailed below:

1. *Measuring Full-authentication Latency:* The supplicant was made to roam from one AP to another in the wireless network, and a full IEEE 802.1X EAP TLS authentication was performed at each re-association. We measured 90 such authentications resulting in an average latency of 1.1 seconds.

2. *Fast Re-authentication:* Fast re-authentication using proactive key distribution was enabled on the RADIUS and the authenticators. The RADIUS server was initialized with the neighbor graph shown in figure 5.7. We use a static neighbor graph for ease of demonstration. The graph used in our experiments was constructed by human observation of the re-association messages. Autonomous construction methods detailed earlier should be used in order to keep the neighbor graph fresh and dynamic and this has no effect on the performance of the key distribution scheme. Figure 5.8 shows the authentication latencies. The first authentication (which occurs at the start of a session), is a full-authentication and hence incurs a high latency (approx. 800 ms); while all subsequent 18 re-authentications reflect the latency of

the two-way handshake.

3. *Overhead at the RADIUS server:* In this experiment we measured the additional overhead incurred by communication required for distributing the keys proactive using the *Notify-Request*, *Notify-Accept* and the *Access-Accept* messages. We measured 80 authentications and obtained an average latency of 21 ms. This overhead does not increase the handoff latency.

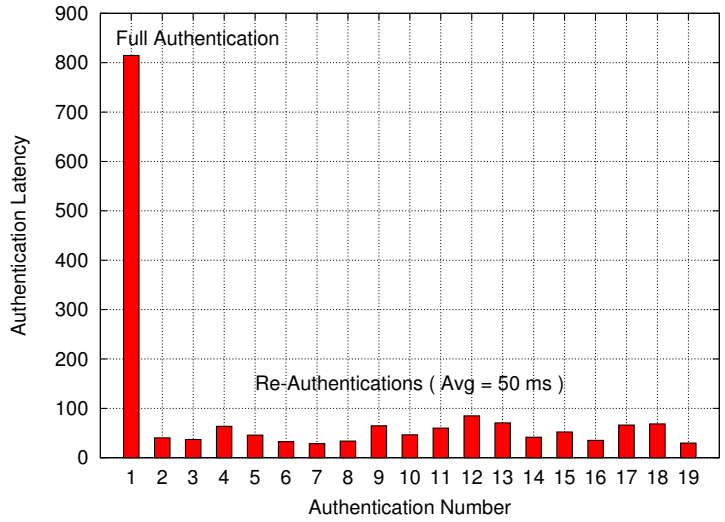


Figure 5.8: Figure shows the authentication latencies as observed by the roaming supplicant in the wireless testbed, with proactive key distribution enabled. As can be seen, the first authentication reflects the full-authentication latency and initiates the key distribution mechanism.

5.4 Related Work

There has been prior work at reducing the authentication latency by doing a predictive pre-authentication to a set of access points. Pack [45, 46] proposes a fast handoff scheme using a *predictive* authentication method based on IEEE 802.1X model. In their scheme, pre-authentication is performed to the k most likely next access points. The k stations are selected using a weighted matrix representing the likelihood (based on the analysis of past network behavior) that a station, associated to AP_i, will move to AP_j. The mobile station may select only the most likely next access points to pre-authenticate, or it may select all of the potential next access points [45, 46]. Pack uses the notion of a frequent handoff region (FHR) to represent the adjacent access points which is obtained by examining the weighted matrix. The weights within the matrix are based on an $O(n^2)$ analysis of RADIUS log information using the inverse of the ratio the number of handoffs from AP_i to AP_j to the time spent by the mobile station at AP_i prior to the handoff. In their paper [45], pre-authentication means the following. When a station authenticates to AP_i, authentication server (AAA server) sends security information not only to AP_i but also to other APs in FHR. As a consequence, the next handoff to one of APs in FHR does not require any message exchanges between the AP and the AAA

server, because the AP already has the security information.

There are several issues with pre-authentication. Firstly, pre-authentication can not occur beyond the first access router due to the fact that EAPOL packets are used to carry authentication information. This severely limits the ability to pre-authenticate to single LANs only and prohibits WAN and Inter-network roaming. Secondly, the cost of a full re-association is prohibitive for a capable device as in the laptop used in our experiments. Imagine the times for a small handset using a low powered processor. In addition, the authentication process must be accomplished to each potential neighbor. Thus, the cost is several seconds rather than milli-seconds. During the authentication time, by the way, the mobile station is on a different channel and unable to process traffic from or from the currently associated access point. Finally, unless there is a significant overlap in coverage pre-authentication will just not work due to the length of times cited earlier.

For the construction of FHR matrix, it requires $O(n^2)$ computation and space, where n is the number of access points in the network, and must be created at the authentication server (AS). Furthermore, the FHR notion does not quickly adapt to changes in the network topology. This is in contrast to our neighbor graphs which require $O(degree(ap))$ computation and storage space per AP and which

quickly adapt to changes in the network topology. Additionally, neighbor graphs can be utilized either in a distributed fashion at each access point, or client, and in a centralized fashion at the AS.

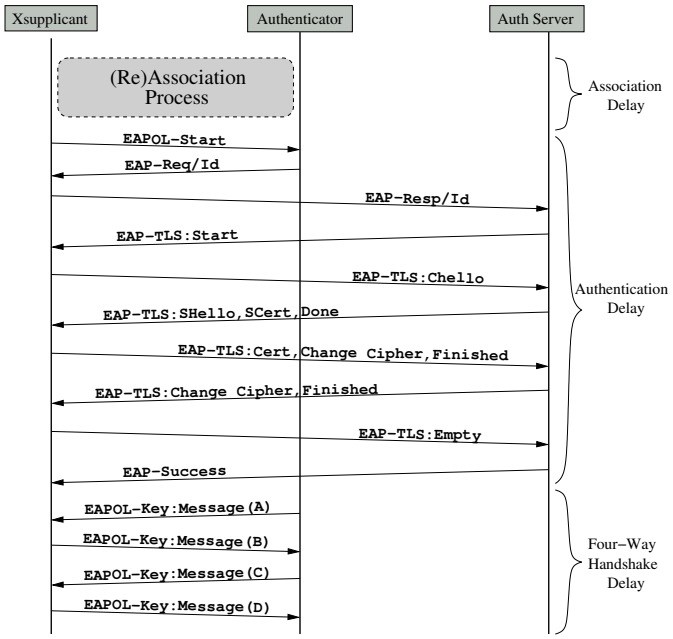


Figure 5.9: Figure shows the complete set of messages exchanged during the (re)association process. In particular, it shows the EAP-TLS authentication messages, and the four-way handshake.

5.5 Summary

In this chapter, we discussed the role played by security in the handoff process. Specifically, we observed that 802.11I based authentication increases the handoff latency by a huge value of 800 ms.

Through this study we showed how our locality hypothesis can improve security in a wireless network. We developed fast re-authentication schemes which generate key material prior to a handoff. This key material is propagated to a limited set of APs determined using the neighbor graph. As noted earlier, neighbor graphs capture locality in user mobility. This locality relates APs that a user associates to in succession. Such a relation places bounds on the key material generation costs for the proactive key distribution scheme. This shows that our scheme achieves the same security properties as a full authentication while at the same time incurs bounded and minimal costs in terms of the handoff latencies, key computation costs and storage. A four-way handshake process, defined in the IEEE 802.11I standard ensures the freshness properties of the distributed keys.

We evaluated our approach through a rigorous testbed based implementation and observed that our approach reduces this authentication latency from 800 ms to 25 ms on average. In the next Chapter, we apply our hypothesis of locality in user mobility to address the scan component of the handoff latency. We discuss extensions to neighbor graphs which provide further details on the local neighborhood of a client to design fast scanning algorithms.

Chapter 6

Fast Active Scanning

A critical component of the handoff process is the active scan function which is also a dominant contributor to the total latency. In Chapter 3, we performed an empirical study based on various experiments in an in-building testbed environment. We had observed that the *probe* or the active scan phase consumes over 90 % of the total handoff latency at the MAC layer (which was measured to be around 400 ms on average). Note that this latency excludes the delay incurred for authentication mechanisms, such as based on the IEEE 802.11i standard. The 802.11i authentication, which happens after the MAC layer handoff takes around 800 ms on average. In Chapter 5, we discussed a key distribution mechanism which uses our locality hypothesis for fast re-keying to eliminate the need for full authentication during handoff. This reduced the authentication latency of 800 ms to the fast

re-authentication latency of around 3-5 ms on average. The focus of this chapter is to address the active scan component of the handoff process.

In this chapter we explore how the locality principle can be employed to design faster scan algorithms. The key insight comes from the fact that the results of successive scans (as part of successive handoffs) are related via locality. This is much like the way two successive APs in an association pattern are related via locality. This is discussed in detail in Chapter 2. In this chapter, we built fast and efficient active scan techniques that take advantage of the local view of the network topology.

The probe (or active scan) returns a set of APs along with signal strength information which aids the client in making handoff decisions. Choosing a good AP to handoff to is an important factor that affects the application throughput. The current active scan algorithm used by most wireless NIC vendors is to scan each channel (Chapter 3 Section 3.4). In IEEE 802.11b, there are 11 channels available and the station would scan each of these channels to search for APs. We call this naive approach, *Full-Scanning* or *Full-Scan* for short. Some wireless vendors (specifically Lucent) implement a slightly modified version, where the station *observes* which channels are used by the wireless network

and scans only those channels during handoff. We call this algorithm, *Observed-Scanning* or *Obs-Scan* for short. This can yield latency improvements if, for example, the wireless network used only 2 or 3 channels out of 11. We note that 802.11b has 11 channels, however, only 3 of them are non-overlapping (channels 1, 6 and 11). Some network administrators decide to use only these non-overlapping channels to avoid unnecessary interference. In such cases, *Obs-Scan* can perform better however, the latencies incurred are still significant.

Recall that a station scans a particular channel in the following manner: The station sends a broadcast *Probe-Request* message to which APs reply with a unicast *Probe-Response* (which is ACK-ed by the station). Since the station has no prior information on how many responses to expect, "how long to wait" on a particular channel becomes a non-trivial question. This is because the total time needed to collect all the responses depends on the number of APs, the contention present in the channel and whether each individual AP received the broadcast *Probe-Response* correctly. The heuristic recommended in the IEEE 802.11 standard specifies that the station wait for a fixed and deterministic period given by the *MinChannelTime* parameter and if it detects activity on that channel, the wait period is extended to *MaxChannelTime* . Through observations, we measured that most vendors

use constants of 17 ms and 34 ms, respectively. With these values, the latency to scan N channels would fall between $17N$ and $34N$, which for $N = 11$, become 187 ms and 374 ms which is significant.

APs bridge traffic between the wired and the wireless segment. These devices form a fixed infrastructure and hence their location is expected to change infrequently relative to user mobility. Also taking advantage of the locality in mobility principle (Chapter 2), obtaining precomputed information that will aid a station in the active scan process is a viable optimization. In the earlier chapters, this locality was captured using neighbor graphs. An edge in a neighbor graph represents a path of re-association. However, the results of a scan contain a list of all APs in range. Thus, two APs might have significant overlap in their coverage areas and might not have any edges between them in the corresponding neighbor graph. To provide detailed information to a client prior to scanning, we extend neighbor graphs by making edges signify the symmetric relation of overlap in coverage.

We build a graph structure, called *overlap graphs*, as an extension to neighbor graphs (Chapter 4), to provide the client with a set of local APs along with their channel information which can be utilized by the scanning algorithms. Here, an edge is added between two APs if their coverage areas *overlap* regardless of their channel of operation.

Overlap is determined by the clients. If a client can communicate with two APs at the same location, their coverage is said to overlap. Symmetric nature of the overlap relation makes the graph undirected. Each AP/vertex in the graph is also marked with its assigned channel to aid in the active scan.

There are two optimizations that can be performed by using precomputed information from overlap graphs. First, a client need not scan a channel if there is no expectation to find any AP on that channel. This information can be obtained from the overlap graph by observing the channel assignment of all APs that are neighbors to the client's current AP in the overlap graph. Second, a client can determine exactly how many responses to expect on a given channel from the overlap graph. This can be actively used in determining how long to wait on a particular channel instead of using the heuristic specified in the IEEE standard. For example, our testbed evaluation in Chapter 3 found that in most cases 4-5 ms of wait time was sufficient to collect all responses. This duration is almost four times less than *MinChannelTime* which would be the minimum time a station would wait on any given channel. Because of the above two reasons, we expect such optimizations to perform significantly better than the existing active scan algorithms, namely *Obs-Scan* and *Full-Scan*.

We first discuss the existing scanning algorithms, namely *Full-Scan* and *Obs-Scan* in further detail in Section 6.1. We discuss the notion of overlap graphs in Section 6.2. We discuss two active scan algorithms, namely, *OG-Scan* and *OGPrune-Scan* which perform the optimizations discussed above in Section 6.3. We present our evaluation of these algorithms using a testbed implementation in Section 6.4 and using simulations in Section 6.5. We summarize this chapter's contribution in Section 6.6.

6.1 Current Scanning Algorithms

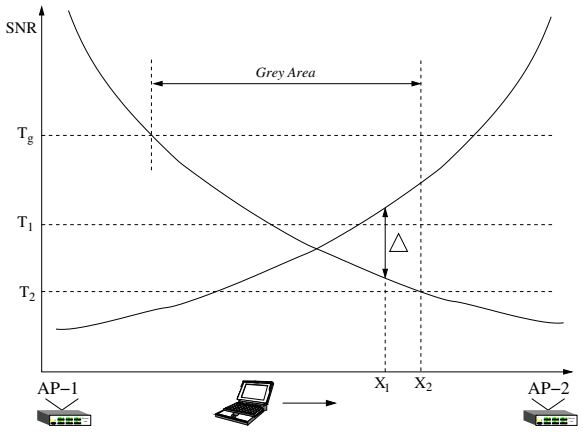


Figure 6.1: Plot of SNR as a station moves from one AP to another. Handoff occurs at X_1 when $T_h = T_1$ and at X_2 when $T_h = T_2$. The *handoff-region* shown is when the handoff threshold $T_h = T_2$.

We present a brief discussion of the active scan functionality in

802.11 for background purposes. The reader is referred to Chapter 3 for further details. We also discuss the two commonly used scanning algorithms, *Full-Scan* and *Obs-Scan* in detail.

A station leaving an access point (AP) initiates the handoff process for finding a candidate set of next-APs to associate with. The station must handoff to maintain service continuity[51]. Also handoff decisions could take into account load balancing constraints, as discussed in [51]. To make this service disruption imperceptible to applications, fast handoff is critical. For example, a handoff completed in less than $50ms$ provides a VoIP user not only continuous conversation but also a smooth transition of the call [52], [53]. [54]. Therefore, when and how to handoff is an important design issue.

When to initiate handoff in the cellular domain has been studied previously [54] [55] [56]. A handoff criteria called the *relative signal strength with hysteresis and threshold* used by Lucent [57] is described below. Figure 6.1 depicts the typical *signal-to-noise ratio*[1] (SNR [58]) changes between two adjacent access points, AP_1 and AP_2. As the station moves from AP_1 to AP_2, the SNR from AP_1 decreases while SNR from AP_2 increases. We denote SNR values from AP_1 and AP_2

[1]Other metrics may be used for hand-off decision, such as Received Signal Strength Indicator (RSSI), Bit Error Rate (BER) or Signal-to-Interference Ratio (SIR). We select SNR in our implementation.

at position x by $S_1(x)$ and $S_2(x)$, respectively. Hand-off initiation is based on two parameters, handoff threshold T_h and hysteresis Δ, both positive. At any position x, the station currently associated to AP_1 initiates hand-off procedure from AP_1 to AP_2 if and only if the following conditions hold :

$$\begin{cases} S_1(x) < T_h \\ S_2(x) - S_1(x) > \Delta \end{cases} \tag{6.1}$$

In Figure 6.1, the station triggers the hand-off process at position X_1 if $T_h = T_1$ and X_2 if $T_h = T_2$. The threshold condition avoids unnecessary hand-offs when the current link quality is sufficient, and the hysteresis condition avoids the *ping-pong* effect [54].

As soon as the hand-off condition holds, the station initiates a hand-off procedure. Below, we discuss the various phases and the respective chapters that address them along with the work presented in this chapter to give the reader an overall picture:

1. *Active Scan or Probe :* Construct a set of candidate APs for hand-off, by performing a scan on each channel of operation (Chapter 3 and 6).

2. *Layer-2 Authentication :* Used for WEP-based shared key authentication. With IEEE 802.11i, this is a dummy exchange, present only as an artifact of the 802.11 state machine and contributes

negligibly to the overall handoff latency. Hence, we ignore this phase for all practical purposes.

3. *Re-association :* Establish layer-2 connectivity with a particular AP. This involves transfer of station *context* according to the IAPP protocol [11] (Chapter 4).

4. *Re-authentication :* Authenticate the user using the IEEE 802.1X [5] framework as described in the IEEE 802.11i Standard [4] (Chapter 5).

5. *Layer-3 Hand-off :* Update binding information and the care of address [59]. This also includes packet forwarding to minimize packet loss. This step is omitted if the station roams to another AP in the same wireless network. In this thesis, we restrict ourselves to such scenarios of mobility where the client stays within a single network domain, also referred to as *micro-mobility*.

The purpose of the probe phase is to construct a set of candidate APs within the client's range. This is achieved by performing an active scan or a probe of each channel of operation. During a probe[2], the client broadcasts a *probe request* messages to which APs respond with

[2]Another method specified in the standard is a passive scan, which few NIC vendors follow.

a *probe response*. Figure 6.2 shows the active scan procedure in detail. As shown in the figure, say N distinct channels are selected for probe. To scan a channel, the client performs the following steps:

1. The client has to first switch to the target channel and wait for a chance to transmit by following the IEEE 802.11 DCF mechanism. We call this latency, the *channel switch and transmission overhead (CST)* as indicated in the figure.

2. The client transmits the *probe-request* frame.

3. The client waits for *probe-responses* from APs that received the probe-request frame. The duration of wait is controlled by the Algorithm 2 and discussed in detail later.

After completing a scan of the selected N channels, the client constructs a list of candidate APs according to a certain criteria – such as strength of received signal, available data rates, etc. An AP is selected from this list for *reassociation*.

We define the amount of time spent by a client waiting for probe-responses, i.e. time spent in step 3 above, as the *probe-wait* time. This variable is critical in determining the total *probe latency* – defined as the total time spent in the probe or the active scan phase.

The above probe process is described in Algorithm 2. This

algorithm takes as input a set S of channels to scan. If this is equal to the set of all available channels, we call it a *Full-scanning* (or *Full-Scan*) algorithm. If this set is restricted to the channels on which APs were found during an earlier probe, we call the algorithm *Observed-scanning* (*Obs-Scan*). *MinChannelTime* and *MaxChannelTime* are two parameters used by the algorithm which affect the probe-wait time in the following manner (steps 6-10 of Algorithm 2) – a client waits for *MinChannelTime* on a particular channel after sending the probe-request. If any transmission is detected during this period (i.e. there are other stations utilizing this channel), the client extends the wait period up-to *MaxChannelTime* (*MinChannelTime* $\leq$*MaxChannelTime*). Thus, if $|S| = N$ and let $MinCT$ =*MinChannelTime* and $MaxCT$ =*MaxChannelTime*, then the total probe latency $\Gamma(N)$ (say) is given by

$$\Gamma(N) = \sum_{i=1...N} \Big[MinCT + F(i) * (MaxCT - MinCT) \Big] \tag{6.2}$$

where

$$F(i) = \begin{cases} 0 & \text{Medium was idle on channel } i, \\ 1 & \text{Otherwise.} \end{cases} \tag{6.3}$$

Clearly, we can upper and lower bound $\Gamma(N)$ as follows:

$$
\begin{aligned}
& N * MinCT \\
& \leq \sum_{i=1...N} \Big[MinCT + F(i) * (MaxCT - MinCT) \Big] \qquad (6.4) \\
& \leq N * MaxCT
\end{aligned}
$$

Figure 6.2 shows an example. Say N channels are to be scanned and APs are present on channels 1 and N. The scan on channel 1 takes *MaxChannelTime* as a probe-response is received before *MinChannelTime* expires, indicating that the medium was not idle. The scan of channel $2 \ldots (N-1)$ takes *MinChannelTime* as there are no APs and stations on those channels, thus no transmission can be detected during *MinChannelTime*. Similarly, on channel N the client has to spend *MaxChannelTime*.

Note that the client did not have to scan channels $2 \ldots (N-1)$ if it had the information that no *neighboring* APs were present on those channels. Also on channel N, the station need not have to wait up until *MaxChannelTime* after receiving the single probe-response message. We shall discuss mechanisms and algorithms in Section 6.3 which will leverage such optimizations to reduce the overall probe latency.

Algorithm 2 Full/Observed-scanning algorithm

S = set of channels to probe.

```
1: for each channel ch ∈ S do
2:     Broadcast probe-request on this channel ch
3:     Start probe timer T = 0
4:     while True do
5:         Wait for probe-responses from APs
6:         if Medium is idle until T ≥MinChannelTime then
7:             break { Move to next channel if medium has been idle
               for MinChannelTime }
8:         end if
9:         if T ≥MaxChannelTime then
10:            break { Wait for MaxChannelTime if medium was not
               idle }
11:        end if
12:    end while
13: end for
```

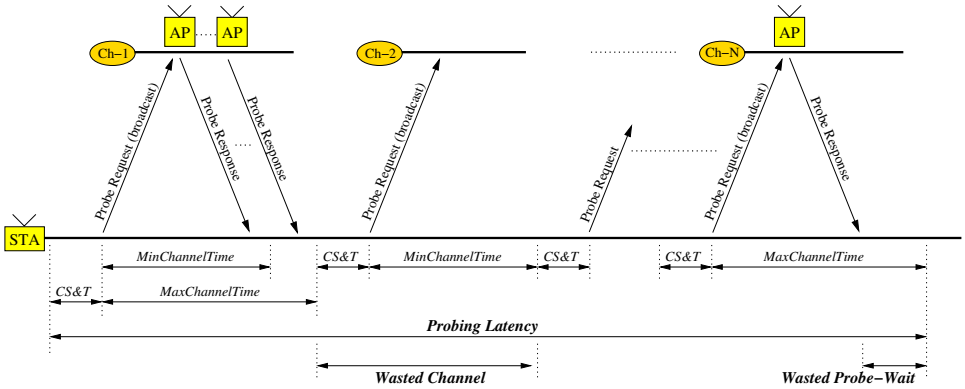


Figure 6.2: Messages during an Active Scan or Probe. CS&T refers to the 'channel switch and transmission overhead'.

6.2 Overlap Graph

The purpose of the active scan is to provide the client with a list of APs along with their signal strength information such that the client can make a good decision on the next access point. From the discussion presented in the previous section, it is clear that the active scan latency depends on (i) the number of channels scanned, and (ii) the duration of wait on each channel in order to receive the probe responses. Both these latencies can be reduced if the client has sufficient information on the neighboring APs and their channel of operation.

Based on this idea, we construct a graph called an *Overlap Graph* where an edge between two APs indicates overlap in their coverage. That is, there exists a location such that a scan of all channels would

result in both APs being reported in the scan results - we call this the *overlap* relation. The symmetric nature of the overlap relation makes the graph undirected in nature.

Neighbor graphs (Chapter 4) capture the asymmetric relation of handoff between two APs. Consider the example shown in Figure 6.3(a). Assume that a client is associated to AP_1. At location Y, the client decides to handoff and thus re-associates to AP_3. Because of circular ranges, a similar re-association from AP_3 to AP_1 is possible. Figure 6.3(b) shows the neighbor graph which has directed edges between AP_1 and AP_3. Now consider location X. Here, the two APs have an overlap in coverage and a scan at location X would include AP_3. However, the signal strength of AP_3 would be weak and very similar to the signal strength of AP_1 at X. Thus, the difference in the signal strengths would be insufficient for the client to handoff between AP_1 and AP_3 (it could possibly handoff to a third AP not shown in the figure). In order to make our active scan accurate, AP_3 should be a neighbor of AP_1 as shown in the overlap graph in Figure 6.3(c). However, the neighbor graph does not have any edges between AP_1 and AP_3 because of the handoff condition not being satisfied.

Formally, the *overlap graph* (OG) is an undirected graph over the set of all access points in the network. An edge of an overlap graph,

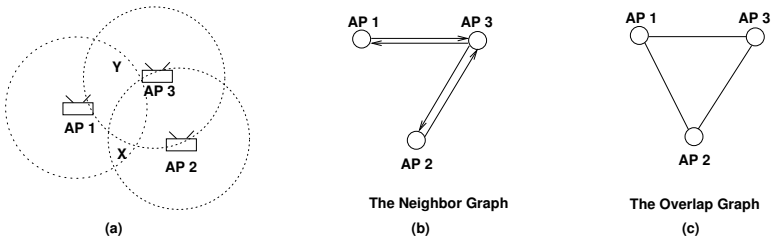


Figure 6.3: Example scenario to illustrate the difference between neighbor graphs and overlap graphs.

$\langle AP_i, AP_j \rangle$ represents the overlap relation between access points. AP_i and AP_j *overlap* if there exists a location where a mobile station can communicate to both of them with 'acceptable' link quality. That is, a probe request at such a location would elicit a probe response from both APs.

6.2.1 Construction

An overlap graph can be obtained by a mobile station's random measurements of signal strengths from various access points. With a very small probability p_{OG}, a station is selected to perform a scan of all channels. The station initiates a full-scan and reports the results to the system. Overlap among APs can be determined from these scan results as per the definition. We call this an *overlap test*. Note that with a sufficient number of stations, generation of the overlap graph using the overlap tests can be completed faster than the generation of

a neighbor graph because it doesn't require user mobility.

Figure 6.5 shows the overlap graph and the neighbor graph for an in-building testbed environment. The testbed used for the experiments in this chapter spanned two floors of an office building comprising of 20 APs. The overlap graph shown in the figure was constructed by performing scans at various points in the building. The arrows indicate the directed edges which are a part of the neighbor graphs. The undirected version of these edges belong to the overlap graph. Undirected edges which belong to the overlap graph and do not belong to the neighbor graph are shown using dashed lines.

6.3 *OG-Scan* and *OGPrune-Scan*

As discussed earlier, the probe phase accounts for a significant portion of the total handoff latency. Thus, for fast handoffs, it is critical to optimize this phase as much as possible. The purpose of the probe phase is to obtain a candidate set of APs for re-association.

We discuss two algorithms, namely, *OG-Scan* and *OGPrune-Scan* . The *OG-Scan* algorithm utilizes the overlap graph to reduce the number of channels to be scanned and to reduce the duration of wait on each channel by knowing exactly how many probe responses to expect. The *OGPrune-Scan* algorithm improves over the *OG-Scan* algorithm

by constructing an additional temporary data-structure, called a *non-overlap* graph using the overlap graph. This is utilized in performing opportunistic pruning of the set of APs to be scanned. We discuss this further in Section 6.3.2. We first present the *OG-Scan* algorithm below.

6.3.1 *OG-Scan* Algorithm

The *OG-Scan* algorithm performs the following optimizations using the overlap graph over the active scan presented in Algorithm 2. Steps 1 and 8 are the improvements over Algorithm 2. The *OG-Scan* algorithm is presented as Algorithm 3.

1. *Reduce the number of channels to be probed:* If there are no neighboring APs present on a particular channel (with respect to the current AP in the overlap graph), that channel is not probed – step 1 in Algorithm 3.

2. *Reduce the probe-wait time:* Probe-wait time is the duration spent by a station on a particular channel waiting for probe responses from the APs. If all APs on a particular channel have responded, the client can move to the next channel instead of waiting for the *MaxChannelTime* timer to expire. Knowledge of the overlap graph gives the client a fine-grained control over the duration of

wait on a particular channel – step 8 in Algorithm 3.

Algorithm 3 : *OG-Scan* algorithm

```
1: for all channel i where any neighbor AP is running do
2:   Broadcast probe request on channel i
3:   Start probe timer
4:   while True do
5:     Receive probe responses
6:     if Medium is idle until MinChannelTime expires then
7:       break
8:     else if all APs on channel i have replied then
9:       break
10:    else if MaxChannelTime expires then
11:      break
12:    end if
13:  end while
14: end for
```

Depending on the density of the wireless network, the *OG-Scan* algorithm can improve the probe latency significantly. We evaluate the performance of *OG-Scan* via simulations in Section 6.5 and via implementation on a testbed network in Section 6.4.

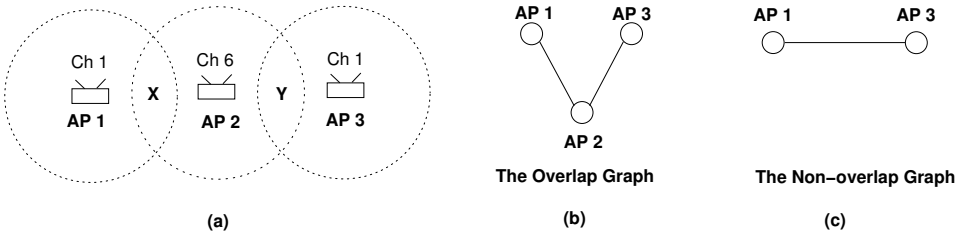


Figure 6.4: Scenario to illustrate the opportunistic pruning possible using the non-overlap graph.

6.3.2 *OGPrune-Scan* Algorithm

The *OGPrune-Scan* algorithm improves over the previous algorithm by performing certain opportunistic optimizations. Consider the example shown in Figure 6.4(a). Here the coverage of AP_2 overlaps with both AP_1 and AP_3. Both AP_1 and AP_3 are on channel 1. Thus, *OG-Scan* would wait for a response from both APs or for *MaxChannel-Time* whichever would come first. However, there is no location where a client can communicate with both AP_1 and AP_2. Thus, the client needs to wait precisely for one probe response, either AP_1 (for example at location X) or AP_2 (at location Y).

We capture the above opportunistic optimization in an algorithmic manner by constructing a non-overlap graph confined to the set of APs that overlap with the client's current AP. Figure 6.4(b) shows the overlap graph for the scenario of Figure 6.4(a). We construct a non-

overlap graph over the set of neighbors of AP_2, which are $\{AP_1, AP_3\}$. The non-overlap graph is shown in Figure 6.4(c). As the client waits for probe responses on channel 1, it can perform opportunistic pruning using the non-overlap graph as given by the following pruning principle :

Non-overlap based Pruning: If the client receives probe response from an AP_i, it implies that the client is within coverage of AP_i. Thus, let $N_i = \{AP_{i_1}, AP_{i_2}, \ldots, AP_{i_k}\}$ denote the set of neighbors of AP_i in the non-overlap graph. This implies that each such edge $\langle AP_i, AP_{i_j} \rangle$ for $j \in \{1 \ldots k\}$ is present in the non-overlap graph. Thus, since the client is within range of AP_i, it is *not* within range of each AP in the set N_i. These APs as denoted by N_i can thus be *pruned* from the non-overlap graph after the client receives a probe response from AP_i.

The above pruning principle is exploited by the *OGPrune-Scan* algorithm in addition to the optimizations performed by *OG-Scan*.

In order to capture the *non-overlap* relation in a more formal manner, we define the following: A *Non-Overlap Graph* (NOG) is the complement of an overlap graph, meaning that $\langle ap_i, ap_j \rangle$ is an edge in the non-overlap graph if and only if $\langle ap_i, ap_j \rangle$ is NOT an edge in the

overlap graph. That is,

$$NOG \stackrel{\text{def}}{=} OG^c. \tag{6.5}$$

When the non-overlap graph is constrained to the set of APs that are a neighbor to the client's current AP, we call it a *local* non-overlap graph. We use the NOG to dynamically prune the set of APs to scan. This may even reduce the number of channels to probe if a pruned-AP was the only AP on its channel. Algorithm 4 describes the *OGPrune-Scan* algorithm.

In Step 1, a local non-overlap graph (NOG) is constructed. This is the induced subgraph of the global non-overlap graph by restricting the APs to the neighbors of the client's current AP in the overlap graph. In Step 3, the degree of each AP in the local non-overlap graph is examined. We select an AP with the maximum degree following the strategy discussed in [60] for the set-cover problem. If a particular AP ap_i responds with a probe response message, this implies reachability to ap_i. Hence, we prune all other APs that are non-overlapping with ap_i – that is, all APs that have an edge to ap_i in the local non-overlap graph. As a result of this pruning, some channels might become devoid of APs to scan, thus resulting in further improvement of the probe latency.

Algorithm 4 : *OGPrune-Scan* algorithm

let NOG be the local non-overlap graph with respect to the client's current AP
while not all APs are probed or pruned **do**
  ch = channel of AP with maximum degree in the local NOG
  Broadcast probe request on channel ch
  while True **do**
    Probe response received from AP_r
    if Medium is idle until *MinChannelTime* **then**
      break
    end if
    Prune all APs that are non-overlapping with AP_r (computed using local NOG)
    if All neighbor APs on channel ch responded or were pruned **then**
      break
    end if
    if *MaxChannelTime* has expired **then**
      break
    end if
  end while
end while

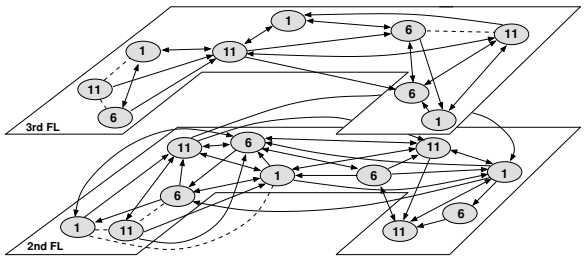


Figure 6.5: The neighbor graph and the overlap graph for the in-building testbed environment. Directed edges show the neighbor graph. These edges are present in an undirected form in the overlap graph. Dashed edges are solely present in the overlap graph.

6.4 Implementation

In this section, we discuss the implementation of the active scan algorithms over a deployed IEEE 802.11b indoor network. We describe the network configuration, implementation, process of performing the experiments and the results. In brief, we implemented four different

Table 6.1: Summary of Experiment Results.

Algorithms	**Probe Count**	**Probe-Wait**	**Latency**
Full-Scan	11.0	10.9 ms	362 ms
Obs-Scan	3.0	11.0 ms	101 ms
OG-Scan	2.5	6.3 ms	70 ms
OGPrune-Scan	2.2	4.4 ms	59 ms

algorithms (*Full-Scan*, *Obs-Scan*, *OG-Scan*, and *OGPrune-Scan*) and measured approximately 250 handoffs on the testbed network.

The *OG-Scan* algorithm reduced the probe latencies of *Full-Scan* and *Obs-Scan* by 80.7% and 30.8%, respectively. The *OGPrune-Scan* algorithm reduced them by 83.9% and 42.1%, respectively. Table 6.1 summarizes the results. In Table 6.1, probe count refers to the average number of probe request messages sent by the client, which equals the number of channels probed. Probe-wait is the duration spent by the client on a particular channel waiting for probe responses. Table 6.1 shows the average probe-wait times as measured. Note that both *OG-Scan* and *OGPrune-Scan* reduce the probe-wait time and the probe count while *Obs-Scan* only reduces the probe count.

6.4.1 Testbed Environment

The deployed wireless network spans two U-shaped floors in a campus building (Figure 6.5). There are nine APs on the third floor and eleven APs on the 2nd floor. Each access point is a Cisco 350 with an omni-directional antenna on the ceiling. Open authentication is used for layer 2 authentication, and the access points are assigned channels 1, 6 and 11, which are known to be non-overlapping in IEEE 802.11b [61]. The geometry of the floors and topologies of the twenty access

points are shown in Figure 6.5. For the mobile station, a laptop with an Intel Pentium 4 Mobile 1.8 GHz and 256 MB RAM, equipped with a Prism 2.5 based *Demarctech* card [62] was used. Linux was used as the operating system.

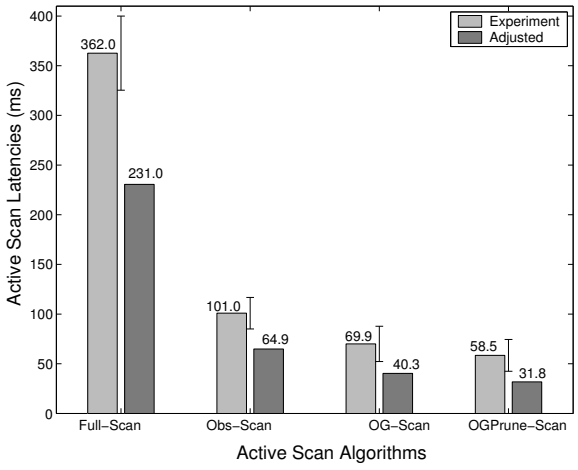


Figure 6.6: Active Scan Latencies for the four scan algorithms. Also shown are the adjusted latencies with the Channel Switch and Transmission overhead = $10ms$. Confidence intervals are also shown.

6.4.2 Software

In most commercial wireless cards, the handoff process is implemented inside the firmware for efficient operation. This includes fast changing of channels for the active scans. Thus, when implemented in the firmware the channel switch and transmission overhead is typically of the order of 1 or 2 ms.

Because the firmware is proprietary and it is hard to obtain firmware source to implement the various scan algorithms, we implemented them as a user-space daemon. The Prism 2 wireless card has a mode which allows the host programs to perform the scan operations. However, when using this interface to do host-based scanning we observed that the channel switch and transmission overhead becomes very high. We measured this overhead to be around 22.2 ms (11.3 ms for channel switch and 10.9 ms for transmission). Thus, we note that when implemented in firmware the scan algorithms would not incur this latency. While reporting the handoff latencies, we report the actual measured latency and also the *adjusted latency* where we reduce the channel switch and transmission overhead to a more realistic value.

We implemented the four scanning algorithms using *Airjack* [63], an open source linux based driver for Prism wireless cards. We used Airjack to implement a roaming daemon in user-space that performs the active scan using the different algorithms. The daemon program, named *roamd*, communicates with the customized Airjack driver so that the mobile station continuously monitors the signal quality with its current AP and initiates the handoff when a certain handoff condition holds. The Airjack driver was customized for the monitoring functionality.

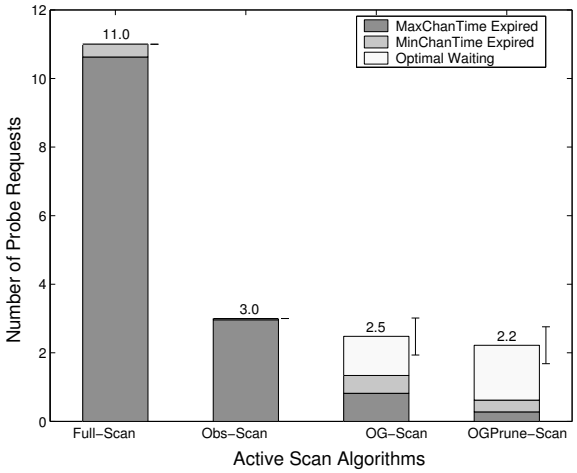


Figure 6.7: Probe count for the four active scan algorithms. Also shown is the distribution of the probe count attributed to either *MaxChannelTime* getting expired, *MinChannelTime* getting expired or optimal waiting.

6.4.3 Measurement Methodology

In our implementation, the probe algorithms are executed as a user-space daemon. We measured the latencies *inside* the daemon for accuracy. To overcome the clock resolution of $10ms$ in Linux, we patched the kernel to get microsecond resolution [64].

A common method used by researchers [65, 66] in the wireless community is to capture all packets over the air using a dedicated system. This approach is called *packet sniffing* and the system is called a packet sniffer. One advantage of packet sniffing is its independence from the system executing the implemented algorithms. However, packet

sniffing fails to measure latencies inside the system. The actual probe latency begins with the internal state transition of the driver/firmware from a normal state (where it is associated to an AP) to a probe state and *not* with the end of successful transmission of the first probe request frame (which will be the only event observed by a packet sniffer). Thus, our approach uses internal measurements from the user-space daemon rather than sniffing.

6.4.4 Experiment Process

Our experiments consists of the following two parts.

1. *Generation of the overlap graph.*

   In order to generate the overlap graph, we issued 475 random overlap tests throughout the area. We used these results to generate the overlap graph offline. This graph structure was stored locally as a file on the mobile station performing the experiments.

2. *Measurement of the probe latencies.*

   For measuring the probe latencies, we induced 250 handoffs for each of the four different algorithms. These handoffs occurred along fixed paths of motion which were carefully chosen to cover all the handoff edges in the neighbor graph shown in Figure 6.5.

To maintain identical conditions across the algorithms, we followed the same exact path for each algorithm during periods of low network activity.

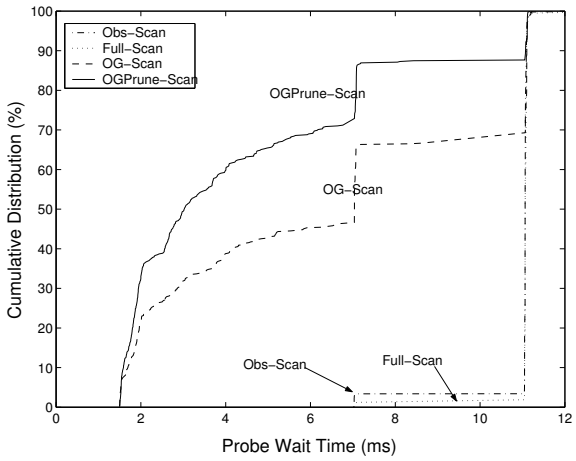


Figure 6.8: Cumulative distribution of probe-wait times for the four algorithms.

6.4.5 Experiment Results

Figure 6.5 shows the neighbor graph and overlap graph constructed by the generation phase. Each circle represents an access point with the assigned channel inside the circle. Arrows with solid lines represent an edge present in both the neighbor graph and the overlap graph. The direction of an arrow shows the direction of the handoff relationship. A dashed line represents an edge present only in the overlap graph. We found that the number of neighbors was 3.15 on average with a

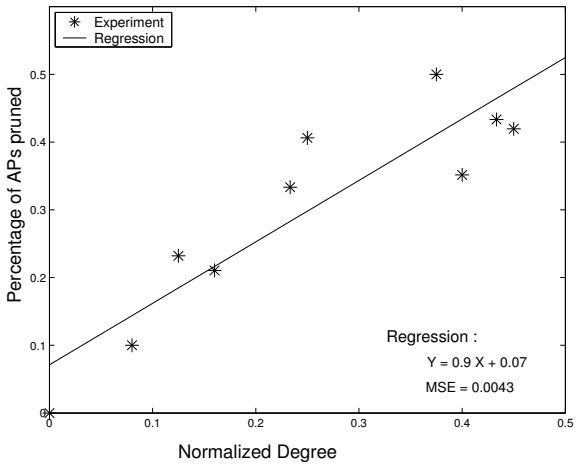


Figure 6.9: Performance of the pruning optimization performed by the *OGPrune-Scan* algorithm as affected by the average normalized degree of an AP in the local non-overlap graph. MSE is the Mean Square Error of the shown regression line.

maximum of 6 while the average neighboring channels that were used was 2.25 (out of 11).

Probe Latencies and Adjusted Latencies

The measured probe latencies are shown in Figure 6.6. The x-axis shows the four different algorithms tested and the y-axis shows the probe latencies in milliseconds. The left bars are the results of our experiments while the right bars show adjusted probe latencies that one could achieve when the channel switch and transmission overhead was $10ms$, which would be a very pessimistic estimate if the algorithm was

implemented in firmware[3]. From the experiments, the *OG-Scan* algorithm reduces the probe latencies relative to *Full-Scan* and *Obs-Scan* by 80.7% and 30.8%, respectively. The *OGPrune-Scan* algorithm reduced them by 83.9% and 42.1%, respectively. The adjusted probe latencies show that this latency can be less than $50ms$ with our algorithms.

Probe Count and Probe-Wait Time

Probe Count refers to the number of channels that were selected for scanning during a single handoff. For example, the *Full-Scan* algorithm would scan all 11 channels, thus resulting in a probe count of 11. This parameter measures the effect of the optimizations performed by the *OG-Scan* algorithm using overlap graphs and the opportunistic optimizations performed by the *OGPrune-Scan* algorithm using the non-overlap graphs. Probe wait refers to the duration spent by the station on a single channel waiting for probe responses. The *OG-Scan* reduces this duration by having a knowledge of how many probe responses to expect. The *OGPrune-Scan* algorithm dynamically decides to wait longer or shorter depending on the probe responses received and the knowledge of the non-overlap graph.

[3]Our measurements indicate channel switch and transmission latencies of around 2-3 ms for the scanning algorithms implemented in firmware.

The analysis of the probe count and the probe-wait times are provided in Figure 6.7 and 6.8. We observe that the *OG-Scan* and the *OGPrune-Scan* algorithms reduce the probe count when compared to *Full-Scan* by 77.45% and 79.82% respectively. When compared to *Obs-Scan* they reduce the probe count by 17.33% and 26.00% respectively.

We use the term *optimal wait* to indicate the *optimal* amount of time needed to wait on a particular channel. This time can be computed offline after observing the probe responses. Ideally the client should wait for the optimal duration (as indicated by optimal wait) on each channel. Information using overlap graphs and pruning using non-overlap graphs aid the client in determining the optimal wait duration. Thus, we can classify each scan as a *MaxChannelTime* expiration (the client probably waited too long), an optimal wait (the client spent the right amount of time) or *MinChannelTime* expiration (the client unnecessarily scanned that channel only to find no activity). Figure 6.7 shows that more than 97% of single channel scans resulted in *MaxChannelTime* expirations for both *Full-Scan* and *Obs-Scan* algorithms. Figure 6.7 also shows that 46 % of the single channel scans turned out to be optimal waits for the *OG-Scan* algorithm. This number was boosted to 72 % for the *OGPrune-Scan* algorithm indicating the success of the opportunistic pruning performed by *OGPrune-Scan*.

In these experiments, *MaxChannelTime* was set at $11ms$ and *MinChannelTime* at $7ms$ as recommended in [65]. The duration of optimal waits was measured to be $2.7ms$ on average with a standard deviation of $1.4ms$.

Figure 6.8 shows the cumulative distribution of the probe-wait times for each algorithm. The graph shows that *OG-Scan* and *OGPrune-Scan* have much shorter probe-wait times than the other two algorithms. For example, scans that take less than $7ms$ occur 46.6% of the time for *OG-Scan* and 71.2% of the time for *OGPrune-Scan* . The peaks at $7ms$ and $11ms$ explain the high density around *MinChannelTime*($7ms$) and *MaxChannelTime*($11ms$).

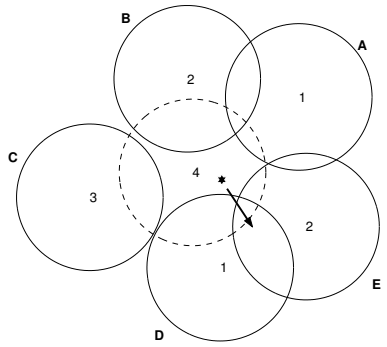


Figure 6.10: Example of a topology generated in simulations. The dashed circle is the current AP; solid circles are the neighbor APs. The number in the circle is the assigned channel. The station, represented by a star, moves as indicated by an arrow.

OGPrune-Scan performance vs the number of neighbors

As discussed earlier in Section 6.3.2, the *OGPrune-Scan* algorithm improves the scan latency by performing opportunistic pruning of the set of APs to be scanned and thereafter also the number of channels to be scanned. This optimization is done by considering the local non-overlap graph. The number of APs that get pruned would depend on the average degree of the APs in the non-overlap graph. This observation was reflected in our measurements.

We observed that higher the average degree of APs in the non-overlap graph, greater was the number of APs that got pruned by the *OGPrune-Scan* algorithm. Figure 6.9 shows this result. The x-axis shows the average degree of an AP in the local non-overlap graph normalized by the size of that graph. The y-axis shows the percentage of APs that were pruned out of all the APs present in the non-overlap graph.

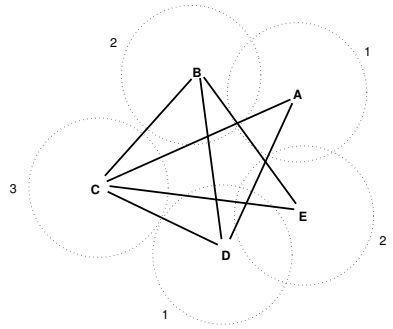


Figure 6.11: Example of a non-overlap graph generated in simulations

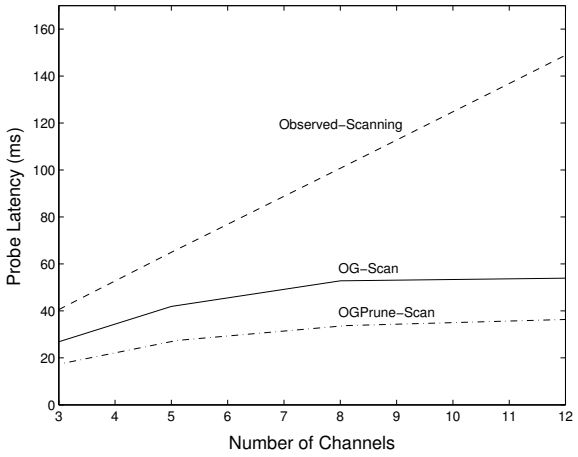


Figure 6.12: Probe latencies of three algorithms vs. the number of channels.

6.5 Simulations

The goal of the simulations is to investigate the performance of the various scanning algorithms under different configurations:

- The number of independent channels from 3 to 12
- The number of neighbors from 2 to 8

We assume optimal channel assignments in which no adjacent APs have the same channel, if possible. We only simulate local topologies, i.e, the current AP and its neighbors.

6.5.1 Simulation Model

The following assumptions are made for simplicity.

1. The radio coverages of all APs are identical circles centered at the corresponding APs.

2. The positions of neighbors, $\{AP_1, AP_2, \ldots, AP_m\}$ where m is the number of neighbors, are randomly chosen around the current access point, AP_c with the following conditions :

$$\begin{cases} \text{For } i = 1, 2, \ldots, m, \\ \qquad R \ \leq \ Distance(AP_c, AP_i) \ \leq \ 2 \times R \\ \\ \text{For distinct neighbors } AP_i \text{ and } AP_j, \\ \qquad Distance(AP_i, AP_j) \ \geq \ R \end{cases}$$

   where R is the radius of the coverage and $Distance()$ is the Euclidean distance.

3. Access points AP_i and AP_j overlap each other if

$$Distance(AP_i, AP_j) \ \leq \ 2 \times R$$

4. The direction of the mobile station is randomly chosen so that there exists at least one neighbor AP to handoff.

5. AP_i is considered to be reachable by the mobile station c if and only if

$$Distance(c, AP_i) \ \leq \ R$$

6. There exists no contention from other mobile stations

The constant values are shown in Table 6.2.

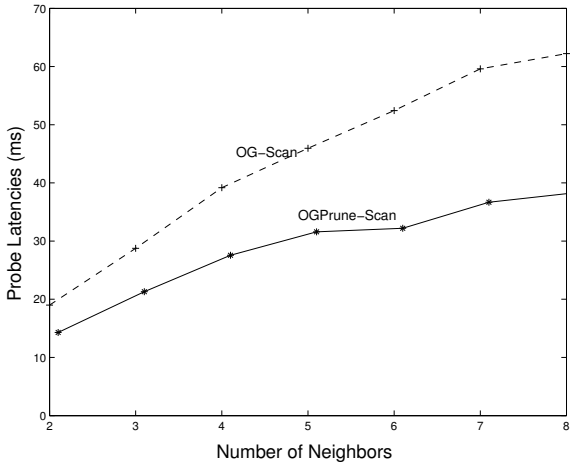


Figure 6.13: Effect of the pruning optimization performed by *OGPrune-Scan* over *OG-Scan* versus number of neighbors.

Table 6.2: Constants used in Simulations

Constants	**Values**
MaxChannelTime	11 ms
MinChannelTime	7 ms
Round Trip Time (RTT)	2 ms
Channel Switch & Transmission	5 ms

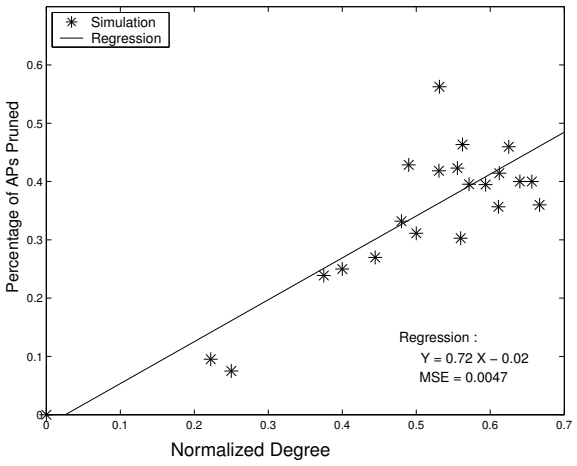


Figure 6.14: Performance of the pruning optimization performed by the *OGPrune-Scan* algorithm as affected by the average normalized degree of an AP in the local non-overlap graph. MSE is the Mean Square Error of the shown regression line.

6.5.2 Simulation Process

Table 6.3 shows the various parameters used in the simulations. For each combination of the parameters, ten different local topologies were randomly generated according to the model. Channels are assigned to the APs according to the following rules :

Table 6.3: Varying Parameters in Simulations

Varying Parameters	**Values**
The Number of Neighbors	$\{2, 3, \ldots, 8\}$
The Number of Channels	$\{3, 5, 8, 12\}$

- The channel used by an AP is not assigned to any of its neighbors.
- If *channel count* > *neighbor count*, then

  assign distinct channels to the neighbors while leaving one channel for the current AP. Here, *channel count* refers to the number of available channels and *neighbor count* refers to the number of neighbors.
- otherwise,

  assign (*channel count* − 1) channels so that no overlapping neighbors have the same channel. When *channel count* = 3, assigning the same channel to overlapping APs may be inevitable

Once topology and channel assignment are determined, the mobile station makes ten different handoffs in randomly chosen directions. The probe latencies are measured using the four different algorithms. Figure 6.10 illustrates an example of a generated topology when the neighbor count is 5 and the channel count is 4. The dashed circle is the current AP and the others are its neighbors. The numbers represent the assigned channels while the letters are for reference purposes. The star and an arrow illustrate the mobility of a station. Note that only access points D and E are reachable to the mobile station at the point of handoff. Also the neighboring cells need not cover the entire

boundary of the current AP.

Figure 6.11 shows the generated local non-overlap graph from the above topology which has an average degree of 2.8.

Table 6.4: Percentage reduction relative to *Obs-Scan*.

Channel Count	***OG-Scan***	***OGPrune-Scan***
3	33.8 %	56.1 %
8	47.6 %	66.5 %
12	63.8 %	75.6 %

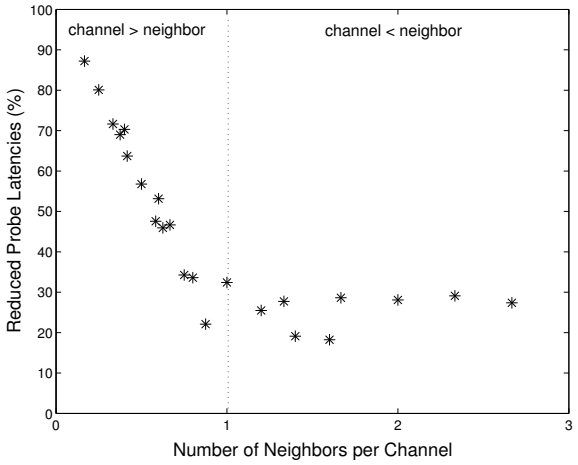


Figure 6.15: Performance improvement of the *OG-Scan* algorithm relative to *Obs-Scan* as a function of number of Neighbors-per-channel.

6.5.3 Simulation Results

Below we present the salient points of our simulation results.

Increasing Number of Channels Improves Performance

Figure 6.12 shows that the performance of our algorithms improve when the number of independent channels increase. Table 6.4 shows the percentage-reduction of the probe latencies relative to the *Obs-Scan* algorithm for three different channel counts (i.e. number of available channels).

Table 6.4 shows that the percentage reduction in the scan latency grows almost linearly with a coefficient of 3.48 for *OG-Scan* and 2.26 for *OGPrune-Scan*, obtained by a linear regression using the least squares method.

Smaller Neighbor-per-Channel Density Helps

Neighbor-per-channel density is the average number of neighbors of a randomly selected AP per channel. The smaller the value, greater is the performance improvement of the *OG-Scan* algorithm over the *Obs-Scan* algorithm as shown in Figure 6.15. But when the neighbor count becomes greater than the channel count (right side of the vertical dotted lines), the local channel reuse increases resulting in a reduction in the performance gain of *OG-Scan* algorithm over *OGPrune-Scan*.

Better Pruning with Increasing Number of Neighbors

Figure 6.13 shows the performance differences between *OG-Scan* and *OGPrune-Scan* versus neighbor count. As shown in the graph, the performance gap between *OGPrune-Scan* and *OG-Scan* increases with greater number of neighbors. Figure 6.14 shows the relationship of pruning performance and non-overlap degree to the neighbor count ratio. Similar but extended results of the experiments are shown in Figure 6.9.

6.6 Summary

In this chapter, we focussed on the scanning latency. We constructed a graph that captures overlap in coverage among APs. This overlap graph positions the client with respect to a localized set of APs. The localized information about the APs and their channel of operation is used by the scanning algorithms. Overlap graph was used to design a fast active scan algorithm called *OG-Scan* that improves the latency by (i) reducing the number of channels to scan and (ii) reducing the duration of wait on each channel being scanned. We also discussed a pruning method to opportunistically shrink the set of APs that need to be scanned by computing a non-overlap graph structure.

We evaluated both algorithms in contrast with the current active scanning techniques using a testbed implementation and through simulations. We observed that both algorithms perform very well relative to the current best known techniques and bring about reductions between 40-80% depending on various parameters. These algorithms performed well in practice and in simulations precisely because they made use of the information available on the local APs. This information proved to be useful because of the locality present in user mobility. The results of this study thus support our hypothesis that there is locality in user mobility.

Chapter 7

Summary

For a successful technology, reality must take precedence over public relations, for Nature cannot be fooled. - Richard P. Feynman.

The primary contribution of this thesis is the development and analysis of the notion of mobility induced locality among base-stations or access points that form a wireless network. We proposed and evaluated an efficient structure called *Neighbor Graphs* that captures this locality in a distributed fashion and provides important properties of resilience and autonomous maintenance. This notion and representation applies to any infrastructure wireless network such as a cellular network or a wireless LAN. We evaluated the practical usefulness of this mechanism by applying it to the real-world problem of handoffs in the context of wireless LANs.

The notion of mobility induced locality simply states that there

	Scanning	Re-association	Re-authentication
IEEE Method	400 ms	15 ms	800 ms
Neighbor Graphs	50 ms	2-3 ms	20 ms

Table 7.1: Overall comparison of the cost improvement for handoffs using algorithms based on neighbor graphs versus the IEEE standard.

is locality in user mobility measured to within any realistic accuracy. When applied to user mobility in a wireless LAN environment, this *relates* access points (APs) through which users perform handoffs. Two APs through which a user can perform a handoff are said to be 'local' to each other. This crucial observation allows us to firstly define what a local neighborhood of a client is with respect to the APs around it. Secondly, this allows us to take advantage of this locality in user mobility by designing schemes that are aware of this local view of the network topology as the user moves. These schemes perform certain pre-computations by prefetching necessary information such that the amount of computation and time spent *during* the handoff is minimized. This is similar to how caching in various areas in Computer Science takes advantage of locality in file accesses, memory and web requests as some examples.

In this dissertation, we performed a detailed study of this hand-

off process in wireless LAN environments. This process as described by the IEEE 802.11, 802.1X and 802.11i standards can be divided into two conceptual phases: (i) Discovery: The client performing a handoff scans the available channels to search for APs belonging to a desired extended service set (ESS) or more loosely a wireless LAN. Through experiments over an in-building wireless testbed, we measured this latency to be around 400 ms on average. (ii) Re-authentication: This (optional) phase depends on the security settings in effect. An IEEE 802.11i compliant network would require a full 802.1X based authentication upon *each* handoff. This essentially being a brute-force approach consequently incurs a heavy performance latency. Our testbed based implementation showed an average latency of about 800 ms per authentication.

From a testbed based evaluation (Chapter 2) we concluded that the current handoff mechanisms were inadequate to support seamless mobility of users. In fact given the high handoff cost of about 1.3 seconds, this became a deterrent to user mobility instead of enabling it, contrary to the expectations of a wireless network. The mobility of a user was exposed to the higher layers of the networking stack in the form of delay jitters and packet losses lasting for the duration of the handoff.

Representing Locality Using Neighbor Graphs

Given notion that user mobility induced locality among APs, the first challenge was to capture this locality in the form of an algorithmic structure. This was an easier problem in the context of cellular networks where each cell has regular coverage and a fixed number of neighbors.

We used the notion of locality to create a graph theoretic structure called neighbor graphs. This structure captured the locality topology of a wireless LAN. Two APs were designated as neighbors if a client could perform a handoff between them. By observing handoffs, the network could construct and maintain (against infrequent topology changes) this structure to reflect the locality in the physical environment. The neighbor graph is directed in nature indicating the asymmetry present in the RF and handoff behavior.

The neighbor graph as a general directed graph was the much needed base to develop algorithms for the two phases of the handoff process. For example, when applied to a cellular network a neighbor graph would exhibit very regular properties and can be approximated using a bounded and fixed degree graph. Thus handoff algorithms which operated on the general assumption of a neighbor graph and required no other inputs from the wireless network would perform well in diverse network deployments both indoor and outdoor.

By applying neighbor graphs to the problem of handoffs which benefits from locality induced by mobility, we observe that neighbor graphs are a good structure to represent such locality. Table 7.1 shows the overall performance improvements we achieved by using neighbor graphs. An overall factor of 10 reduction in the handoff latencies show the practical viability of this representation mechanism. We next summarize the improvements obtained in each phase individually.

Proactive Caching

Based on the hypothesis of locality in mobility (Chapter 2) and the neighbor graph structure that captures this locality in a wireless LAN environment, we developed a proactive caching algorithm that prepositions a roaming client's context information to maintain it one-hop ahead. In the normal case, the Inter-Access Point Protocol (IAPP [39]) was used to perform a *reactive* transfer of the station's context. This method incurred a latency of about 15 ms as observed in our testbed based implementation.

The proactive caching algorithm prepositioned a roaming client's context to the set of candidate APs determined using the neighbor graph. The candidate APs maintain all such contexts in a fixed size memory cache as APs are memory constrained devices. Thus, a cache-

hit indicates that the latency for an IAPP transfer was averted and the caching algorithm was successful.

We evaluated the performance of this algorithm extensively through simulations which considered various randomly generated topologies of different sizes and densities. We observed that the algorithm was scalable in the number of users and the performance degraded gracefully. We also demonstrated the performance through a testbed based implementation and observed a reduction of the re-association latency (without authentication) from 15 ms to about 2 ms on average.

Proactive Key Distribution

The IEEE 802.11i mandated that a fresh full authentication be performed during each re-association. Thus for a secure network, the handoff latency was furthered by about 800 ms. This was essentially a brute-force approach as the security relationship that was built with the previous AP during a handoff was discarded away.

We employed neighbor graphs to build a fast and secure key distribution scheme that provided perfect forward secrecy and the same security properties of a full authentication and costed about 25 ms on average. The backend authentication server (AAA) implementing the scheme constructed the key material in a secure fashion and pre-

distributed it to the candidate set of APs determined using the neighbor graph. We evaluated this approach through a rigorous testbed based implementation and observed that the latency reduced to about 25 ms on average.

Fast Active Scanning

As discussed earlier, the discovery phase of the handoff process deals with searching for a candidate set of APs in the vicinity of the roaming client. Our testbed evaluation showed that most vendors implement a simple brute force approach of scanning each channel available, thus incurring a high latency of about 400 ms on average.

In Chapter 6, we discussed methods to improve the active scan latency by extending neighbor graphs to include edges indicating overlap among APs. Specifically, we constructed an overlap graph structure that captures overlap in coverage among APs. This graph was used to design a fast active scan algorithm that improves the scan latency by (i) reducing the number of channels to scan and (ii) reducing the duration of wait on each channel being scanned. We also discussed a pruning method to opportunistically shrink the set of APs that need to be scanned by computing a non-overlap graph structure. We evaluated both algorithms in contrast with current active scanning techniques us-

ing a testbed implementation and through simulations. We observed that both algorithms perform very well relative to current best known techniques and bring about reductions of about 40-80% depending on various parameters.

Further Applications

Neighbor graphs have shown solid applicability as a locality topology for important optimizations in the wireless LAN environment. They present interesting venues for further applications to mobility and wireless networking in general. We discuss one such possibility in detail:

Given the development of the newer metropolitan scale wireless networks (WiMAX), popularity of the wireless LANs and competition from the cellular companies, a user has multiple options to obtain network connectivity. This inevitably gives a large number of choices and roaming possibilities. Neighbor graphs can have interesting applications here with edges indicating certain types of handoff possibilities. Edges can be categorized as intra-network or inter-network edges. An inter-network edge informs the client of the possibility of using a different network to obtain service. This information can be proactively obtained prior to an impending handoff. A user policy-driven decision mechanism can start the process for a layer-3 handoff to a different

wireless network much before the deadline to perform such a handoff. Such possibilities can realize seamless user mobility both within and among overlapping wireless networks.

In this particular application, neighbor graphs still retain their essence as a locality topology. At the same time, they provide valuable hints on locality among different wireless networks apart from locality among base-stations belonging to the same network. This 'hierarchical' structure of locality as captured by our modifications to the neighbor graphs can have a profound impact on the growing inter-network mobility problem.

This dissertation has proposed and evaluated the concept of locality induced by user mobility in wireless networks. Specifically, we have demonstrated the importance of a graph theoretic structure that captures this locality in a wireless network. Through this research experience we believe that the wireless networks of today and the ones to be designed tomorrow will indeed need to capture locality in user mobility in an algorithmic fashion much like a neighbor graph in order to eliminate the effects of mobility on the higher layers of the networking stack.

BIBLIOGRAPHY

[1] Daniel Noble. The history of land-mobile radio communications. In *Institute of Radio Engineers*, May 1962.

[2] George Calhoun. *Digital Cellular Radio*. Arctech House Inc, 1988.

[3] Theodore S. Rappaport. *Wireless Communications: Principles and Practice*. Prentice Hall Communications, 2002.

[4] IEEE. Standards for local and metropolitan area networks: Medium access control (mac) security enhancements‘. *IEEE Standard 802.11I*, 2004.

[5] IEEE. Standards for local and metropolitan area networks: Standard for port based network access control. *IEEE Standard 802.1X*, 2002.

[6] L. Blunk and J. Vollbrecht. Ppp extensible authentication protocol (eap). *RFC 2284*, March 1998.

[7] T. Dierks and C. Allen. The TLS Protocol. RFC 2246, January 1999.

[8] B. Aboba and D. Simon. Ppp eap tls authentication protocol. *RFC 2716*, October 1999.

[9] Open-source implementation of ieee 802.1x standard. URL: http://www.open1x.org.

[10] Jon Edney and William Arbaugh. *Real 802.11 Security: Wi-Fi Protected Access and 802.11i*. Addison-Wesley Professional, 2003.

[11] IEEE. Ieee recommended practice for multi-vendor access point interoperability via an inter-access point protocol across distribution systems supporting ieee 802.11 operation. *IEEE Standard 802.11F*, 2003.

[12] How mobile phone networks work. URL: http://www.sitefinder.radio.gov.uk/mobilework.htm.

[13] J. Li, N. Shroff, and E.K.P. Chong. Channel carrying: A novel handoff scheme for mobile cellular networks. In *Proceedings of IEEE Infocom*, 1997.

[14] Tara Salih and Kemal M. Fidanboylu. Performance analysis and modeling of two-tier cellular networks with queuing handoff calls.

In *Proceedings of the Eighth IEEE International Symposium on Computers and Communications*, 2003.

[15] Sang-Joon Park, Ji-Young Song, Jongchan Lee, Kwan-Joong Kim, and Byung-Gi Kim. A handover scheme in clustered cellular networks. *Future Gener. Comput. Syst.*, 20(2):221–227, 2004.

[16] International Telecommunication Union. General Characteristics of International Telephone Connections and International Telephone Circuits. ITU-TG.114, 1988.

[17] Mark Brehob and Richard Enbody. An analytical model of locality and caching. *ichigan State Univ. Computer Science Dept. Technical Report*, 1996.

[18] Armand M. Makowski Sarut Vanichpun. Comparing strength of locality of reference popularity, majorization, and some folk theorems. In *Proceedings of IEEE Infocom*, 2004.

[19] Yijun Yu, Kristof Beyls, and Erik H. D'Hollander. Visualizing the impact of the cache on program execution. *Journal for the Integrated Study of AI, Cognitive Science and Applied Epistemology*, 2004.

[20] Virg'ilio Almeida, Azer Bestavros, Mark Crovella, and Adriana de Oliveira. Characterizing reference locality in the www. In *Pro-*

ceedings of IEEE Conference on Parallel and Distributed Information Systems, 1996.

[21] Moustafa Yossef and Ashok Agrawala. The horus wlan location determination system. In *Proceedings of the Third International Conference on Mobile Systems, Applications, and Services (MobiSys 2005)*, 2005.

[22] Paramvir Bahl and Venkata N. Padmanabhan. Radar: An in-building rf-based user location and tracking system. In *Proceedings of IEEE Infocom*, 2000.

[23] OpenBSD based access points using the Soekris Boards. URL: http://www.missl.cs.umd.edu/wireless/testbed/.

[24] Host AP driver for Intersil Prism Cards. URL: http://hostap.epitest.fi.

[25] Linux driver for Prism based wireless cards. URL: http://www.linux-wlan.com/linux-wlan/.

[26] Charles Chien. *Digital Radio Systems on a Chip*. Kluwer Academic Publishers, 2001.

[27] IEEE. Draft Recommended Practice for Multi-Vendor Access Point Interoperability via an Inter-Access Point Protocol Across

Distribution Systems Supporting IEEE 802.11 Operation. *IEEE Draft 802.1f/D3*, January 2002.

[28] IEEE. Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. *IEEE Standard 802.11*, 1999.

[29] Sangheon Pack and Yanghee Choi. Pre-Authenticated Fast Handoff in a Public Wireless LAN based on IEEE 802.1x Model. *IFIP TC6 Personal Wireless Communications 2002*, October 2002.

[30] Sangheon Pack and Yanghee Choi. Fast Inter-AP Handoff using Predictive-Authentication Scheme in a Public Wireless LAN. *IEEE Networks 2002*, August 2002.

[31] C. Rigney, W. Willats, and P. Calhoun. Remote Authentication Dial In User Service (RADIUS). RFC 2869, June 2000.

[32] Diane Tang and Mary Baker. Analysis of a metropolitan-area wireless network. In *Mobile Computing and Networking*, pages 13–23, 1999.

[33] Kevin Lai, Mema Roussopoulos, Diane Tang, Xinhua Zhao, and Mary Baker. Experiences with a mobile testbed. In *Proceedings of The Second International Conference on Worldwide Computing and its Applications (WWCA'98)*, Mar 1998.

[34] A. Balachandran, G. Voelker, P. Bahl, and P. Rangan. Characterizing user behavior and network performance in a public wireless lan, 2002.

[35] Magdalena Balazinska and Paul Castro. Characterizing Mobility and Network Usage in a Corporate Wireless Local-Area Network. In *International Conference on Mobile Systems, Applications, and Services*, May 2003.

[36] Arunesh Mishra, Minho Shin, and William Arbaugh. An empirical analysis of the ieee 802.11 mac layer handoff process. In *Computer Communications Review (ACM SIGCOMM) (To Appear)*, 2003.

[37] R. Koodli and C.E. Perkins. Fast Handover and Context Relocation in Mobile Networks. *ACM SIGCOMM Computer Communication Review*, 31(5), October 2001.

[38] Pat Calhoun and James Kempf. Context transfer, handoff candidate discovery, and dormant mode host alerting. http://www.ietf.org/html.charters/seamoby-charter.html.

[39] IEEE. Recommended Practice for Multi-Vendor Access Point Interoperability via an Inter-Access Point Protocol Across Distribution Systems Supporting IEEE 802.11 Operation. *IEEE Standard 802.1f*, July 2003.

[40] IEEE. Draft 4 Recommended Practice for Multi-Vendor Access Point Interoperability via an Inter-Access Point Protocol Across Distribution Systems Supporting IEEE 802.11 Operation. *IEEE Draft 802.1f/D4*, July 2002.

[41] C. Rigney, S. Willens, A. Rubens, and W. Simpson. Remote Authentication Dial In User Service (RADIUS). RFC 2865, June 2000.

[42] Soekris Engineering. URL: http://www.soekris.com.

[43] M. Nakhjiri, C. Perkins, and R. Koodli. Context Transfer Protocol. *Internet Draft : draft-ietf-seamoby-ctp-01.txt*, March 2003.

[44] IEEE. Recommended Practice for Multi-Vendor Access Point Interoperability via an Inter-Access Point Protocol Across Distribution Systems Supporting IEEE 802.11 Operation. *IEEE Draft 802.1f/Final Version*, January 2003.

[45] Sangheon Pack and Yanghee Choi. Fast Inter-AP Handoff using Predictive-Authentication Scheme in a Public Wireless LAN. *IEEE Networks 2002 (To Appear)*, August 2002.

[46] Sangheon Pack and Yanghee Choi. Pre-Authenticated Fast Handoff in a Public Wireless LAN based on IEEE 802.1x Model. *IFIP*

TC6 Personal Wireless Communications 2002 (To Appear), October 2002.

[47] S. Capkun, Levente Buttyan, and Jean-Pierre Hubaux. Self-Organized Public-Key Management for Mobile Ad Hoc Networks. *IEEE Transactions on Mobile Computings*, 2003.

[48] Radia Perlman. An algorithm for distributed computation of a spanningtree in an extended lan. In *Proceedings of the ninth symposium on Data communications*, pages 44–53, 1985.

[49] Radia Perlman. *Interconnections, Second Edition: Bridges, Routers, Switches and Internetworking Protocols*. Pearson Education, September 1999.

[50] Arunesh Mishra, Nick Petroni, William Arbaugh, and Timothy Fraser. Security issues in ieee 802.11 wireless local area networks: A survey. *Wireless Communications and Mobile Computing*, 2004.

[51] Fahd K. Al-Bin-Ali, Prasad Boddupalli, and Nigel Davies. An Inter-Access Point Handoff Mechanism for Wireless Network Management: The Sabino System. In *ICNN 2003*, 2003.

[52] R. Shirdokar, J. Kabara, and P. Krishnamurthy. A QoS-based Indoor Wireless Data Network Design for VoIP. In *Vehicular*

Technology Conference, 2001. VTC 2001 Fall. IEEE VTS 54th, volume 4, pages 2594–2598, October 2001.

[53] Arunesh Mishra, Minho Shin, and William Arbaugh. Context Caching using Neighbor Graphs for Fast Handoffs in a Wireless Network. In *Proceedings of IEEE Infocom*, 2004.

[54] G. P. Pollini. Trends in Handover Design. *IEEE Communications Magazine*, March 1996.

[55] Mikael Gudmundson. Analysis of Handover Algorithms. In *IEEE Vehicular Technology Conference, VTC91*, pages 537–542, 1991.

[56] N. Zhang and Jack M. Holtzman. Analysis of Handoff Algorithms using Both Absolute and Relative Measurements. *IEEE Transactions on Vehicular Technology*, 45(1):174–179, February 1996.

[57] Lucent Technologies Inc. Roaming with WaveLAN/IEEE 802.11. Technical Report WaveLan Technical Bulletin 021/A, December 1998.

[58] Hiroto Aida, Yosuke Tamura, Yoshito Tobe, and Hideyuki Tokuda. Wireless Packet Scheduling with Signal-to-Noise Ratio Monitoring. In *25th Annual IEEE Conference on Local Computer Networks (LCN'00)*, November 2000.

[59] D. B. Johnson, C. E. Perkins, and J. Arkko. Mobility Support in IPv6. *Internet Draft draft-ietf-mobileip-ipv6-18.txt, Internet Engineering Task Force (IETF)*, June 2002.

[60] David S. Johnson. Approximation Algorithms for Combinatorial Problems. In *Proceedings of the Fifth Annual ACM Symposium on Theory of Computing*, pages 38–49, 1973.

[61] Lucent Technologies Inc. IEEE 802.11 Channel Selection Guidelines. Technical Report WaveLan Technical Bulletin 003/A, November 1998.

[62] Demarc Technologies Group. URL: http://www.demarctech.com.

[63] Robert Baird and Michael Lynn. Airjack Driver. http://802.11ninja.net/airjack.

[64] High Resolution POSIX Timers. http://sourceforge.net/projects/high-res-timers.

[65] Arunesh Mishra, Minho Shin, and William A. Arbaugh. An Empirical Analysis of the IEEE 802.11 MAC Layer Handoff Process. *ACM Computer Communications Review*, 2003.

[66] J. Yeo, S. Banerjee, and A. Agrawala. Measuring Traffic on the Wireless Medium: Experience and Pitfalls. Technical Report CS-TR 4421, December 2002.