Improving the Latency of 802.11 Hand-offs using Neighbor Graphs

Minho Shin
Dept. of Computer Science
University of Maryland
College Park, Maryland 20742
mhshin@cs.umd.edu

Arunesh Mishra
Dept. of Computer Science
University of Maryland
College Park, Maryland 20742
arunesh@cs.umd.edu

William A. Arbaugh
Dept. of Computer Science
University of Maryland
College Park, Maryland 20742
waa@cs.umd.edu

ABSTRACT

The 802.11 IEEE Standard has enabled low cost and effective wireless LAN services (WLAN). With the sales and deployment of WLAN based networks exploding, many people believe that they will become the fourth generation cellular system (4G) or a major portion of it. However, the small cell size of WLAN creates frequent hand-offs for mobile users. If the latency of these hand-offs is high, as previous studies have shown, then the users of synchronous multimedia applications such as voice over IP (VoIP) will experience excessive jitter. The dominating factor in WLAN hand-offs has been shown to be the discovery of the candidate set of next access points. In this paper, we describe the use of a novel and efficient discovery method using *neighbor graphs* and *non-overlap graphs*. Our method reduces the total number of probed channels as well as the total time spent waiting on each channel. Our implementation results show that this approach reduces the overall probe time significantly when compared to other approaches. Furthermore, simulation results show that the effectiveness of our method improves as the number of non-overlapping channels increases, such as in the 5 GHz band used by the IEEE 802.11a standard.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication networks**]: Network Architecture and Design—*wireless communication*; E.1 [**Data Structures**]: Distributed Data Structures

General Terms

Design, Performance, Experimentation

Keywords

802.11, WLAN, Hand-off, Scanning, Probing, Neighbor Graph, Mobility Graph

MobiSys'04, June 6–9, 2004, Boston, Massachusetts, USA.
Copyright 2004 ACM 1-58113-793-1/04/0006 ...$5.00.

1. INTRODUCTION

The 802.11 IEEE Standard [1] enables low cost and effective wireless LAN services. The unlicensed and free spectrum (2.4GHz in 802.11b/g and 5GHz in 802.11a) used by 802.11 networks permits the deployment of high speed (11Mbps in 802.11b and up to 54 Mbps for 802.11g/a) by organizations [2]. Furthermore, the major laptop and handheld computer vendors are quickly integrating WLAN devices into their equipment. This rapid adoption makes many people believe that 802.11 will become the fourth generation cellular system (4G) or a major portion of it. In fact, WLANs in public areas such as airports, hotels, universities [3] [4] and shopping centers [5] have already been successfully deployed. To meet this lofty goal of becoming the next generation cellular system, the hand-offs that occur when a user is mobile must be efficient.

The small cell size in WLAN creates frequent hand-offs potentially causing delays or disruption of communications if the latency of the hand-off is high. A major component of the hand-off process is identifying the new best available access point (AP) and associating to that AP (layer-2 hand-off). When IP connectivity is used, the additional process of layer-3 hand-off must also be completed [6].

Mobility and voice communications are the driving forces in current cellular networks, and the efficiency of both are paramount to the success of a new generation of cellular service. Due to the high bandwidth provided by 802.11 networks, Voice over IP (VoIP) is the logical choice for providing voice service. VoIP, however, requires a maximum end-to-end delay of $50ms$ [7][8]. Unfortunately, previous studies have shown that the majority of WLANs cannot complete the layer- 2 hand-off process in 100 ms [9] [6] [10]. One study [9] found that the observed layer 2 hand-off latencies are from $60ms$ to $400ms$ ($252ms$ on average) depending on the vendors of wireless cards and access points, and that the probe phase (the discovery of next AP) is a dominating factor in layer 2 hand-off latency, accounting for more than 90% of the overall cost [9].

In active scanning[1], the probing latency is affected significantly by two parameters: the probe count and the probe-

[1] In this paper, we consider only active scanning, where the station broadcasts probe-request messages on interested channels. The alternative method, passive scanning may also be used for improving handoff latency by listening to beacons during idle time. But when AP Discovery is required, there is no guarantee that the station can get the list of APs in time.

wait time[9]. The probe count equals the number of channels probed by the station and the probe-wait time is the time spent by the station waiting for probe responses probed access points. In this paper, a station is a 802.11 mobile device, following the convention in IEEE standard. Since the 802.11 IEEE Standard does not specify a method for probing channels, wireless vendors use their own, usually proprietary, algorithms based on heuristics [9]. We categorize these proprietary algorithms as **Full-Scanning** and **Observed-Scanning**. Full-scanning is a brute force algorithm that probes all the legitimate channels (11 channels in US [11]). Observed-scanning, on the other hand, limits probes to a subset of legitimate channels observed by previous probings [12]. The main benefit of observed-scanning over full-scanning is more easily seen by understanding how channel management usually occurs. In the U.S., there are three non-overlapping or independent channels (1, 6, and 11). The exclusive use of these channels prevents interference between adjacent channels. Thus in a network using only the non-overlapping channels, observed-scanning needs to probe at most three channels instead of 11. Observed-scanning, however, does not work well when the number of non-overlapping channels is high as in the 802.11a standard which has 12 non-overlapping channels.

In this paper, we propose two innovative and efficient layer-2 hand-off schemes, **NG algorithm** and **NG-pruning algorithm**. The NG algorithm uses the *neighbor graph* data structure (NG) [13]. The NG-pruning algorithm further improves the discovery process by also using the *non-overlap graph* data structure.

The neighbor graph is a data structure that abstracts the hand-off relationships between access points. An access point, AP_1 has a handoff relationship with AP_2 (from AP_1 to AP_2) if and only if a station can hand-off from AP_1 to AP_2. AP_2 is then said to be a neighbor of AP_1. The neighbor graph captures the following information :

1. The set of channels on which neighbor APs are operating.
2. The set of neighbor APs on each of such channels.

Using the above information, the station can avoid probing unnecessary channels and spending time waiting for responses from non-existing APs.

The non-overlap graph is a data structure that abstracts the non-overlapping relationships between APs. Two APs are non-overlapping if and only if a mobile station cannot communicate with both of them with acceptable link quality. When two APs are non-overlapping, a probe response from one of them indicates the unreachability to the other. Thus, during a hand-off, the station can exclude (prune) such unreachable APs from the list of APs to probe, resulting in a faster hand-off.

Using these data structures, the NG algorithm and NG-pruning algorithm can:

(i) reduce the number of channels to probe

(ii) reduce waiting time spent on each probed channel

(iii) (NG-pruning) reduce number of APs to probe.

We have implemented the probing algorithms (full-scanning, observed-scanning, NG, and NG-pruning). We have also compared their performance by experiments in a deployed IEEE 802.11b indoor network. In our experiments, the NG algorithm reduces the probing latencies of full-scanning and observed-scanning by 80.7% and 30.8% on average, respectively. The NG-pruning algorithm reduces the latencies of full-scanning and observed-scanning by 83.9% and 42.1% on average, respectively. Furthermore, simulations demonstrate the performance of the algorithms in various topologies using different parameters, such as the number of neighbors and the number of independent channels. The simulation results show that the performance gain of the NG and NG-pruning algorithms increases almost linearly as the number of channels increase, and the performance increases dramatically as the average number of neighbors per channel decrease.

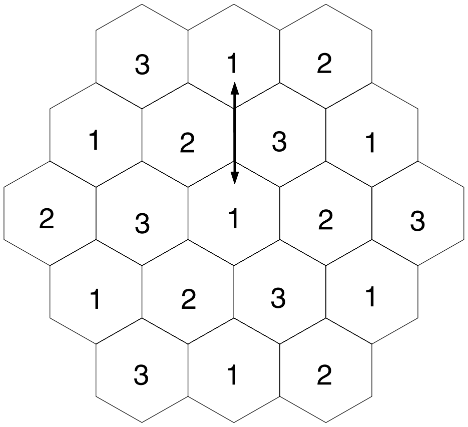

Figure 1: ***Cell concept and optimal channel allocations with three independent channels. The arrow shows the maximum distance between cells with the same channel.***

The paper is organized as follows. In section II, we provide background information on hand-offs and probing algorithms. Section III and IV provide definitions of datastructures and describe our probing algorithms. After discussing the results of our experiments and simulations in section V and VI, related work is presented in section VII. We discuss practical issues and future work, followed by the conclusion in section VIII.

2. BACKGROUND

In this section, we explain the basic cellular concept and the hand-off problems in cellular networks, leading to the discussion of hand-offs in 802.11 networks. The probing process is also described in detail.

2.1 Cellular Concept

The *Cellular Concept* was a major breakthrough in solving the problems of spectrum allocation and user capacity in early mobile radio systems [14]. While a single, high powered base station with an antenna on a tall tower could provide a large coverage area, it made it impossible to reuse the same frequency throughout the area due to interference. One base station also imposed a serious limitation in user capacity. For example, the Bell mobile system in New York City in the 1970s could only support a maximum of twelve simultaneous calls over an area of a thousand square miles

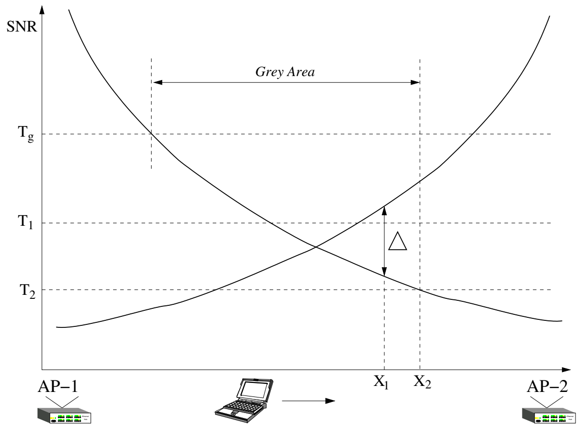

Figure 2: ***SNR changes between two access points and hand-off parameters. Δ denotes hysteresis. Station hand-offs at X_1 when $T_h = T_1$ and at X_2 when $T_h = T_2$. The Grey Area shown is when hand-off threshold $T_h = T_2$.***

[14]. The cellular concept offered very high user capacity within a limited spectrum allocation by replacing a single large powered cell with many low powered cells, each covering only a small portion of the service area (Fig. 1).

The cellular concept introduces two necessary techniques, *optimal channel allocation* and *hand-offs*. Optimal channel allocation allows non-adjacent cells to use the same channel while minimizing interference between such cells (see Fig. 1). Due to the decreased coverage of each cell, however, a mobile station may move into a different cell while a session is in progress. The hand-off process identifies the next base station (cell) and transfers on-going session– ideally with minimal to no disruption of service.

Hand-offs are a major challenge in 802.11 networks since they occur more frequently due to the small coverage area of an access point. In 802.11, to overcome the even smaller coverage area of an access point (on average 30 meters in 802.11b and smaller in 802.11a), multiple access points are necessary to cover the same area as a single CDMA or GSM base-station. Non-overlapping channels, or independent channels, are channels that provides enough frequency separation to co-locate several radios links without interference. In 802.11b, two channels separated by more than 5 channels are known to be independent. The limited number of non-overlapping channels (three in 802.11b, twelve in 802.11a) also necessitate well designed channel assignments to avoid interference from neighboring access points. In Third Generation mobile systems[15], a mobile station can conduct seamless hand-offs by activating several radio links simultaneously (*soft handover* [16]). Unlike 3G mobile stations, a 802.11 mobile station must complete the hand-off process using only a single radio link (*hard handover*) as the standard currently prevents a mobile station from associating to more than one access point within a network at the same time. This limitation makes efficient hand-offs in 802.11 networks difficult to achieve.

2.2 Hand-offs

In 802.11, a station leaving an access point is required to initiate a hand-off process for finding the next access point and establishing a link with that access point (reassociation). The stations must hand-off to maintain service continuity and load balancing of the system[10]. Properly designed hand-offs also should be done early to avoid interference with stations in other cells (cell dragging). To make the hand-off imperceptible to users, a fast hand-off is critical. For example, a hand-off completed in less than $50ms$ provides a VoIP user not only a continuous conversation but also an unnoticeable transition of the call [8], [13]. However, a hand-off longer than $50ms$ can cause disruption of service due to the loss of more than two voice packets. A poorly designed hand-off can also incur *ping-pong hand-offs* due to the momentary fading of the signal strength [17]. Therefore, the when and how to hand-off is a very important design challenge.

When to initiate hand-off has been studied previously [17] [18] [19]. A hand-off algorithm called the *relative signal strength with hysteresis and threshold*, used by Lucent [12] is described below. Fig. 2 depicts the typical *signal-to-noise ratio*[2] (SNR [20]) 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 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 :

$$\begin{cases} S_1(x) < T_h \\ S_2(x) - S_1(x) > \Delta \end{cases} \tag{1}$$

In Fig. 2, 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 [17].

As soon as the hand-off condition holds, the station initiates a hand-off procedure. The following summarizes the steps required for the successful completion of a hand-off.

1. *Probing :* Discover the best available AP in the vicinity.

2. *Layer-2 Authentication :* Authenticate the station to the network. Two kinds of authentications are provided : open authentication and WEP. WEP, however, is known to be insecure [21].

3. *Reassociation :* Establish the communication link with the found AP. Exchange necessary information such as supported transmission rates and beacon interval.

4. *802.11i Authentication :* Authenticate the user by 802.1x [22] and EAP-TLS [23] as described in IEEE 802.11i Standard [24] [25].

5. *Layer-3 Hand-off :* Update binding information and the care of address [26]. This also includes packet forwarding to minimize packet loss.

[2] 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.

2.3 Probing in 802.11

Probing is the dominating factor in hand-off latency, accounting for more than 90% of the overall latency [9]. The probing process (or scanning process) finds a new available AP with the best signal quality with respect to the station. Fig. 3 illustrates the probing procedure as described in the IEEE Standard 802.11 [1]. In the figure, N distinct channels are selected to probe. Once the channels to be probed are determined, the station switches to each selected channel and broadcasts a *probe request* frame. We call this latency the Channel Switch and Transmission overhead (CS&T). In Fig. 3, the arrows toward APs represent such probe request frames broadcast on a channel (numbered in a circle). Upon receiving a probe request, APs respond with *probe response* frames to the station (downward arrows in Fig. 3). After the transmission of the probe request, the station waits for a certain amount of time (*probe-wait time*) before switching to the next channel. After probing all selected channels, the next access point is determined from the information received in the probe responses and their associated SNR.

The algorithm 1 describes the process described above. If the given channels to probe include all legitimate channels, the algorithm is called the full-scanning algorithm. If the channels are observed channels from previous probes (or from passive monitoring), the algorithm is called the observed-scanning algorithm– note that $MaxChannelTime \geq MinChannelTime$.

Algorithm 1 Full-scanning or Observed-scanning algorithm

```
 1: for each channel to probe do
 2:   Broadcast probe request on this channel
 3:   Start probe timer
 4:   while True do
 5:     Read probe responses
 6:     if Medium is idle until $MinChannelTime$ expires then
 7:       break
 8:     end if
 9:     if $MaxChannelTime$ expires then
10:       break
11:     end if
12:   end while
13: end for
```

In line 4 of algorithm 1, the station determines whether to stop on $MinChannelTime$ or stay until $MaxChannelTime$. If the medium is idle for $MinChannelTime$, the station can conclude to stop and probe the next channel. However, if medium is detected to be busy before $MinChannelTime$ expires, the received packet can be either a probe response or other packets. Despite the second case, there is a good reason for the probing station to remain on the current channel because it is possible that an AP responding with a probe response fails to gain access to the medium due to contention with other transmitters.

Fig. 3 illustrates both cases when the probe-wait time is $MinChannelTime$ or $MaxChannelTime$. In the figure, the station starts probing by switching to channel *Ch-1* and transmitting a probe request frame. Since the medium is detected busy before $MinChannelTime$ expires, the station keeps waiting until $MaxChannelTime$ expires. On channel *Ch-2*, the medium is detected to be idle until $MinChannelTime$ expires. Therefore, the station stops waiting and proceeds to the next channel and so on. Note that this algorithm is wasting time on channel *Ch-2* where no APs exist (wasted channel). Moreover, on channel *Ch-N*, the algorithm keeps waiting for more arrivals even after receiving the last probe response (wasted probe-wait). We claim that the removal of wasted channels and wasted probe-wait time is possible with prior knowledge the of local topology, presented by neighbor graph.

3. MOBILITY GRAPH, NEIGHBOR GRAPH[3] AND NON-OVERLAP GRAPH

In this section, we define the mobility graph as the aggregated mobility trace of the mobile stations in a WLAN. Neighbor graph [13] is an adaptive approximation of the mobility graph. The generation of the neighbor graph and its properties are discussed. We also define a notion of a non-overlap graph which is used by the probing algorithm.

3.1 Mobility Graph

Assume $\mathbf{AP} = \{AP_1, AP_2, \ldots, AP_n\}$, is a set of access points in a WLAN under consideration. The association pattern (or mobility pattern) of a mobile station c during a finite period of time (between t_0 and t) can be denoted as $\Gamma(c,t) =< AP_{c1}, AP_{c2}, \ldots, AP_{ck} >$ where k is the number of access points that served the mobile station c and AP_{ci} is the i*th* such access point. From the sequence $\Gamma(c,t)$, we can construct a *personal mobility graph*, $PMG(c,t) =< V, E >$ with the set of vertices V and set of edges E, where $V = \mathbf{AP}$ and

$$E = \{< AP_i, AP_j > \mid AP_i \text{ and } AP_j \text{ are successive in } \Gamma(c,t)\}.$$

A personal mobility graph is a directed graph that represents the mobility pattern of the station in a WLAN. Now, we define a *mobility graph* as the aggregation of personal mobility graphs for all mobile stations, i.e,

$$MG(C,t) = \bigcup_{c \in \mathbf{C}} PMG(c,t)$$

where $\mathbf{C}$ is the set of all mobile stations.

Due to changes in mobility patterns, the mobility graph dynamically evolves over time. For example, at some point, the mobile stations begin to handoff from AP_i to AP_j (edge-addition), or stop handoffs from AP_k to AP_l (edge-deletion). We assume that a mobility graph, $MG(C,t)$, may have edge-additions and edge-deletions during the period (t_0, t).

3.2 Neighbor Graph

3.2.1 Generation

The neighbor graph is an approximation of the mobility graph. A neighbor graph can be configured statically, i.e, generated once and never changes during operation. However, constructing a static neighbor graph to approximate the mobility graph is challenging. The hand-off relationships between access points are not trivial due to the irregularity of radio coverage, caused by various obstacles such as walls, furniture and moving objects. Large radio coverage compared to the inter-AP distance also makes it difficult to predict hand-off relationships without real experiments.

[3] Proactive caching with neighbor graphs has been added to the IEEE Best Practice Document 802.11f [13] [27].

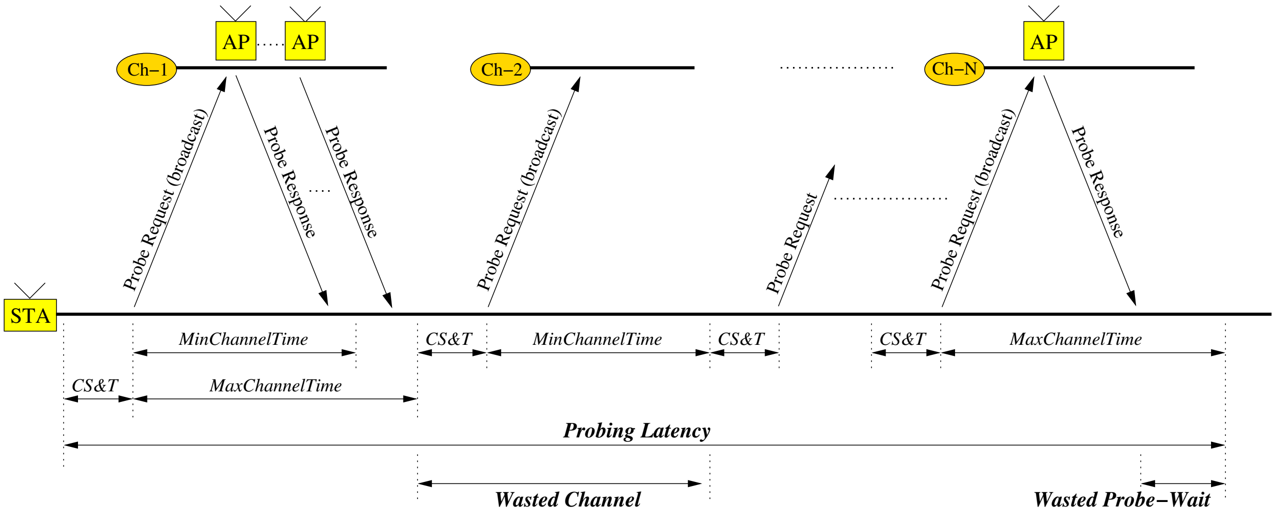

Figure 3: ***A Probing process in the IEEE Standard 802.11. CS&T in the figure means ”Channel Switching and Transmission Overhead”.***

Moreover, a static neighbor graph fails to approximate the mobility graph which changes dynamically over time.

Thus, we explain an empirical method of building a self-adaptive neighbor graph that adaptively changes according to the mobility graph [13]. By the following two operations, such a neighbor graph can be generated and maintained effectively.

- *edge-addition* : when a mobile station hand-offs from AP_i to AP_j and $< AP_i, AP_j > \notin NG$, then add $< AP_i, AP_j >$ to the NG.
- *edge-deletion* : when no staion hand-offs from AP_i to AP_j during a given interval T, remove $< AP_i, AP_j >$ from the NG. T is a chosen timeout value such that $T \geq$ average handoff interval at all edges.

This generates a neighbor graph empirically by observing the actual movement of the mobile stations during operation.

Assume NG is initialy an empty graph. With the aforementioned method of generation, the majority of hand-offs can cause edge-additions during the early age of the neighbor graph. This initial NG-building phase can reduce the benefit of neighbor graphs in the proposed probing algorithms and other neighbor graph dependent systems such as [13] [25]. To eliminate such a high-cost period, the system can conduct a neighbor graph construction phase, in which a dedicated mobile station roams throughout the WLAN area along every possible path so that the system can learn as many edges as possible. In our experiment, a neighbor graph was generated by this method. Alternatively, a fast neighbor graph building method using overlap graphs is given in section VIII.

3.2.2 Implementation

There are various ways of implementing neighbor graphs in a WLAN. In a *centralized method*, the neighbor graph is stored in an NG-server with hand-off events reported by APs. The NG server also provides mobile stations with a neighbor graph when they join the network. This approach, however, has a scalability limitation. In a *distributed method*, each access point stores its local neighbor graph, i.e, the list of its neighbor access points, and mobile stations retrieve local neighbor graphs from access points after (re)association. Building and managing neighbor graphs in a distributed fashion are explained in [13]. In a *user-oriented method*, each user keeps track of its mobility pattern, called a personal neighbor graph. Combining the user-oriented mode with the centralized or the distributed mode can provide more accurate and sophisticated neighbor graph service. Synchronization between neighbor graphs (in a server or APs) with personal neighbor graphs (in each station) should be resolved for proper operation. In our experiment, a global neighbor graph was stored in the mobile station.

3.2.3 Quality

In this section, we introduce two metrics that reflect the quality of neighbor graphs.

The *error* of a neighbor graph, during a period (t_0, t) is the probability that a mobile station makes a hand-off along an edge which does not exist in neighbor graph, i.e,

$$Er(NG, t) \overset{\text{def}}{=} \frac{\text{the number of edge-additions in } (t_0, t)}{\text{the number of handoffs in } (t_0, t)} \quad (2)$$

This measures the probability that a neighbor graph fails to provide the potential next access points to stations. When $Er(NG, t) > 0$, there exist edge-addtions in the mobility graph and the first hand-off along that edge cannot benefit from the use of the neighbor graph. However, the edge is now added providing future mobile stations with an up-to-date mobility graph, minimizing the error of the neighbor graph. The error of any superset of a mobility graph (a complete graph, for instance) is zero, i.e, if $NG \supseteq MG$ at all time in (t_0, t), Error(NG,t) = 0.

The *overhead* of a neighbor graph, during a period (t_0, t) reflects the wasted resource in operating the neighbor graph. Let the residence time of an edge be the total time during which the edge existed in (t_0, t), denoted as $Res(e, t)$ where e is an edge in the NG. Define the overhead of a neighbor graph as:

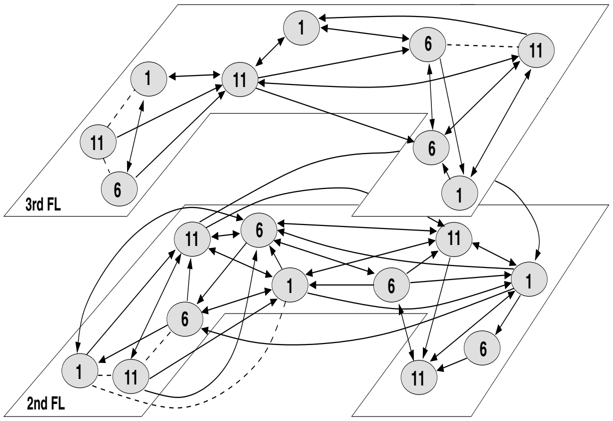

Figure 4: ***Generated Neighbor Graph (solid arrows) and Overlap Graph (solid and dashed lines)***

$$Ov(NG, t) \overset{\text{def}}{=} \frac{\text{the number of edge-deletion} * T}{\sum_{e \in E} Res(e, t)} \qquad (3)$$

where E is the set of all edges that existed in NG during (t_0, t) and T is the edge-deletion timeout.

This measures the ratio of time during which unnecessary edges existed in the NG to the sum of the residence time of all edges. When $Ov(NG, t) > 0$, there exist edge-deletions in the mobility graph, i.e. there exist some unnecessary edges in neighbor graph, for a short period. These extraneous edges not only waste memory but also provide mobile stations with unnecessary neighbor information, degrading the performance of the algorithm. However, a carefully chosen edge-deletion timeout, T, can minimize the overhead, thus making the neighbor graph converge to the mobility graph continuously.

3.2.4 Asymmetry of Hand-offs

A hand-off relationship reflects the proximity of the two APs to each other. One would expect a hand-off relation to be symmetric. However, in reality, the hand-offs are often asymmetric, i.e, a hand-off relationship from AP_i to AP_j does not imply a hand-off relationship from AP_j to AP_i. The asymmetric nature of the hand-off relationship is due to geographical characteristics, the irregularity of radio coverage, high AP density, or the user's uni-directional mobility patterns. The asymmetry of the hand-offs makes a neighbor graph a directed graph. In our experiments, we observed that 57% of hand-off relationships were asymmetric (see Fig. 4).

3.2.5 Edge Degree

The degree of an AP in a neighbor graph is the number of outward edges from that AP. This represents the number of neighbors to which a station can hand-off from the current AP. The degree of an AP bounds the number of probings in the probing algorithms proposed in this paper. The discussion below shows that in most deployments, the degree of the neighbor graph is bounded by 6.

Despite the irregular contour of radio coverage in reality, a circular model is sometimes drawn as the shape of a cell with one access point [4]. However, the hexagon has been adopted as the most appropriate notation of cells in the literature [28]. Fig.1 shows a typical representation of a wireless service area divided by multiple cells. There are several reasons to use the hexagon as a geometric model of a cell. For the simplicity of analysis, non-overlapping regular congruent polygons, *i.e.*, polygon with same length of sides, are preferred over circles. Covering a Euclidean plane with regular congruent polygons is called *Regular Tessellation.* There are only three regular polygons that tessellate the plane : triangles, squares and the hexagons. Among these polygons, hexagon covers the area with fewest number of cells and also closely approximates a circular shape [14]. Under the hexagon tessellation model, the maximum number of neighbor cells is six in a 2D-plane. In reality, the maximum degree of a neighbor graph tends to be smaller in buildings due to the structure of the building and restrictions on mobility due to walls, etc.. In our experiment, the average number of neighbors was measured as 3.15.

When the network must co-locate multiple access points to increase user capacity, the hand-off process must consider load balancing among APs [10]. A neighbor graph with knowledge of neighboring loads can be used for load balancing. In this case, a neighbor count of more than six is possible. However, we can provide the station with a restricted number of neighbors selectively (randomly or according to load information), to prevent redundant information from degrading the mobile station's hand-off performance.

3.3 Overlap Graph and Non-Overlap Graph

The *overlap graph* (OG) is an undirected graph over **AP**, the set of all access points in the network. An edge of an overlap graph, $< AP_i, AP_j >$ represents an overlapping relationship 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. Acceptable link quality means a link quality good enough to avoid the hand-off being triggered, i.e,

$$S_i(x) \geq T_h \text{ and } S_j(x) \geq T_h.$$

Such access points are called *reachable* from the mobile station and otherwise, called *unreachable.* Note that a mobility graph is a subset of the overlap graph, i.e,

$$MG \subseteq OG^5. \qquad (4)$$

When the edges of an overlap graph are considered bidirectional edges, the overlap graph is a neighbor graph with an error of zero, i.e,

$$Er(OG, t) = 0. \qquad (5)$$

Although the OG has some overhead, an OG can converge to a mobility graph with edge-deletion timeouts, removing unnecessary edges from the OG over time.

An overlap graph can be obtained by a mobile station's random measurements of signal strengths from various access points. With probability p, the station initiates a full-scanning probe and reports the overlapping access points to

[4] Some antennas, such as sector antenna, can produce non-circular coverage. This paper assumes omni-directional antenna is used.

[5] We assume only continuous hand-offs in which the stations are continuously running without losing communication links to access points.

the system. We call this reporting an *overlap test.* Note that with a sufficient number of stations, the generation of the overlap graph by overlap tests can be completed faster than the generation of a neighbor graph because it doesn't require user mobility. In our experiment, the overlap tests were initiated periodically by a randomly moving station, resulting in the overlap graph shown in Fig. 4.

The generation of an overlap graph can be expedited by restricting overlap tests to the gray area. The gray area is where the probability of an overlap between the current AP and other APs is high. The mobile station c is in the gray area of AP_i if

$$T_h < S_i(c) < T_g$$

(see Fig. 2). This excludes unnecessary overlap tests where the station is too close to the current AP.

A *Non-Overlap Graph* (NOG) is a complement graph of an overlap graph, meaning that $< AP_i, AP_j >$ is an edge in the non-overlap graph if and only if $< AP_i, AP_j >$ is NOT an edge in the overlap graph. That is,

$$NOG \overset{\text{def}}{=} OG^c. \tag{6}$$

- **Principle of non-overlapping** : The reachability to one of the non-overlapping APs implies the unreachability to the other.

This principle permits the "pruning" algorithm described in the following section.

4. NEW PROBING ALGORITHMS

In this section, we propose two probing algorithms, NG algorithm and NG-pruning algorithm.

4.1 NG Algorithm

There are two main factors that affect probing latency:

(i) the number of channels to probe (*probe-channel count*)

(ii) waiting time on each probed channel (*probe-wait time*) [9].

With prior knowledge of the neighbor graph, the probe-channel count and the probe-wait time can be reduced. Algorithm 2 describes the NG algorithm.

Algorithm 2 : NG 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:     Read probe responses
 6:     if Medium is idle until MinChanTime expires then
 7:       break
 8:     else if all APs on channel $i$ have replied then
 9:       break
10:     else if MaxChanTime expires then
11:       break
12:     end if
13:   end while
14: end for
```

By probing only the non-empty (AP running) channels (line 1), the probing count is reduced by the number of wasted channels (Fig.3), called *minimum-channel probing* . In line 8-9, the algorithm shortens the probe-wait time by removing the wasted probe-wait time (Fig.3). When no more APs are expected to respond, the next channel is probed, called *optimal-wait probing.*

Note that the number of channels where neighbor APs are running (neighbor channel count) is always less than or equal to the number of independent channels. Assuming that no optimal channel assignment will assign the same channel to adjacent cells, the neighbor channel count is strictly less than the number of independent channels, making the NG algorithm always outperform the observed-scanning algorithm. The performance gain increases as the number of independent channels increases such as in 802.11a.

4.2 NG-Pruning Algorithm

We use the non-overlap graph to gain additional performance improvement in probing. With prior knowledge of the non-overlap graph, the station prunes all APs which non-overlap with reachable APs. For example, assume that AP_i and AP_j are non-overlapping. Once the station receives a probe response from AP_i, by the principle of non-overlapping, it is impossible to receive a probe response from AP_j with acceptable link quality. Thus, there is no reason to wait for the response from AP_j, reducing the probe-wait time on AP_j's channel. This may even reduce the number of channels to probe if AP_j was the only AP on its channel. However, if the station cannot receive a probe response from AP_i with acceptable link quality, then we cannot derive any conclusion about the reachability of AP_j. Algorithm 3 describes the NG-Pruning algorithm.

Algorithm 3 : NG-Pruning algorithm

```
 1: let NOG be a local non-overlap graph
 2: while not all neighbor APs are probed or pruned do
 3:   select channel $i$ of AP with maximum NOG-degree
 4:   Broadcast probe request on channel $i$
 5:   while True do
 6:     Read probe responses from $AP_r$
 7:     if Medium is idle until MinChannelTime expires then
 8:       break
 9:     end if
10:     prune all APs non-overlapping with $AP_r$
11:     if All neighbor APs on channel $i$ responded or be pruned then
12:       break
13:     end if
14:     if $MaxChannelTimes$ has expired then
15:       break
16:     end if
17:   end while
18: end while
```

In line 1, a local non-overlap graph (NOG) is a subset of the global non-overlap graph comprised of only neighbor access points of the current access point. In line 3, the degree is examined in the local NOG. Th selection of an AP with maximum degree is by the same heuristics used in the brute-force algorithm for set-cover problem [29]. Line 10 prunes any APs non-overlapping with the already-responded AP, regardless of their channels. Such pruning reduces the number of APs to probe and sometimes resulting in the exclusion

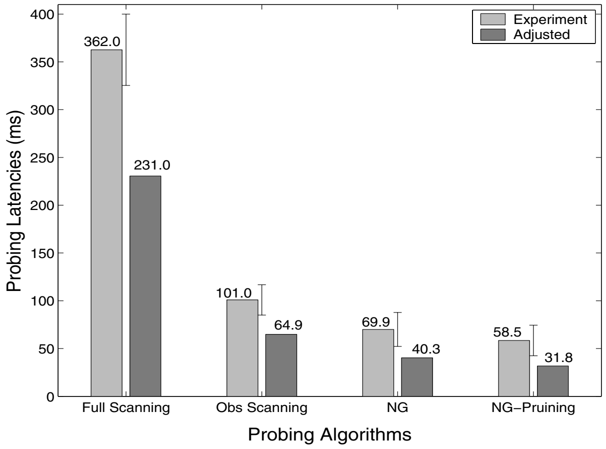

Figure 5: ***Probing Latencies with different Probing Algorithms, and Adjusted Probing Latencies with Channel Switch and Transmission overhead = 10ms. Confidence intervals are also shown.***

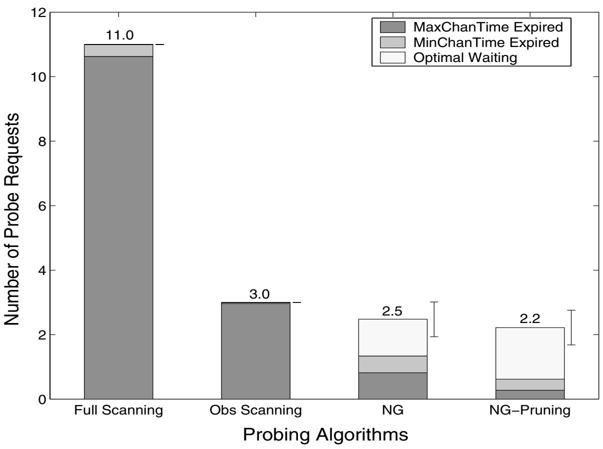

Figure 6: ***Probing Counts and its components with different Probing Algorithms. Each probing is a type of either MaxChannelTime expired, MinChannelTime expired or optimal waiting.***

of a channel to probe. Note that the NG-pruning algorithm also achieves minimum-channel probing and optimal-wait probing, as in NG algorithm.

5. EXPERIMENTS

In this section, we discuss the implementation of the probing algorithms in a deployed 802.11b indoor network. We describe the network configurations, implementations, process of the experiments and the results. In brief, we implemented four different algorithms (full-scanning, observed-scanning, NG, and NG-pruning) and measured approximately 250 hand-offs on the 2nd and 3rd floor in a campus building for each algorithm. The NG algorithm reduced the probing latencies of full-scanning and observed-scanning by 80.7% and 30.8%, respectively. The NG-pruning algorithm reduced the latencies of full-scanning and observed-scanning by 83.9% and 42.1%, respectively. Table 1 summarizes the results. In table 1, probe count means how many probe request frames are broadcast, which equals the number of channels probed. Note that both NG and NG-prunning reduce the probe-wait time and the probe count while observed-scanning only reduces the probe count.

Table 1: Summary of Experiment Results.

Algorithms	Probe Count	Probe-Wait	Latency
Full-Scan	11.0	10.9 *ms*	362 *ms*
Observed-Scan	3.0	11.0 *ms*	101 *ms*
NG	2.5	6.3 *ms*	70 *ms*
NG-pruning	2.2	4.4 *ms*	59 *ms*

5.1 Experiment Configurations

The deployed wireless network spans two U-shaped floors in a campus building (Fig. 4). 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 independent in 802.11b [11]. The geometry of the floors and topologies of the twenty access points are shown in Fig. 4. For the mobile station, a laptop with an Intel Pentium 4 Mobile 1.80 GHz and 256 MB RAM, equipped with a Prism 2.5 based Demarctech card [30] is used. Linux is used as the operating system for implementations and experiments.

5.2 Implementations of Algorithms

In most commercial wireless cards, the hand-off process is implemented inside the firmware for efficient operation. Because the firmware is considered proprietary by all vendors, we emulated the hand-off process using an open source WLAN driver in Linux, e.g. *Airjack driver* v0.6.2-alpha [31]. We used Airjack to implement a roaming daemon in user space that emulates the hand-off process using the different algorithms. The daemon program, named *roamd*, cooperates with the customized Airjack driver so that the mobile station, continuously monitoring the SNR with the current AP, initiates the hand-off process when a certain hand-off condition holds. The Airjack driver is customized for the monitoring functionality.

The Airjack driver works with Prism 2 based wireless cards. As described in section II-C, the probing latency is affected by the cost of channel switching and the transmission (CS&T) latency. In our experiment, the channel switch and transmission latency was on average 22.2*ms* (11.3*ms* for channel switch and 10.9*ms* for transmission). Compared to a 12.4*ms*, channel dwell time measured in [9], this 22.2*ms*[6] is too high for fast hand-offs. This latency is due to the performance limitations of either Airjack driver or the Prism chip itself.

5.3 Measurement Methodology

Since the probing process is controlled by a roaming program, we measure the latencies inside the roaming program for accurate measurement. To overcome the clock resolution of 10*ms* in Linux, we patched the kernel to get microsecond resolution [32].

[6] We have tested several Prism based wireless cards by different vendors with the same results.

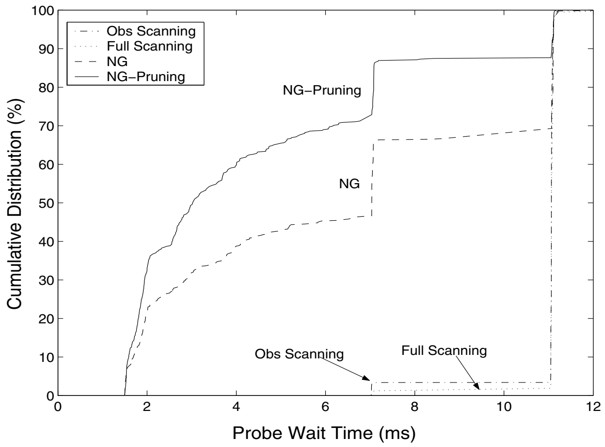

Figure 7: *Cumulative distributions of probe-wait times illustrated for each algorithms.*

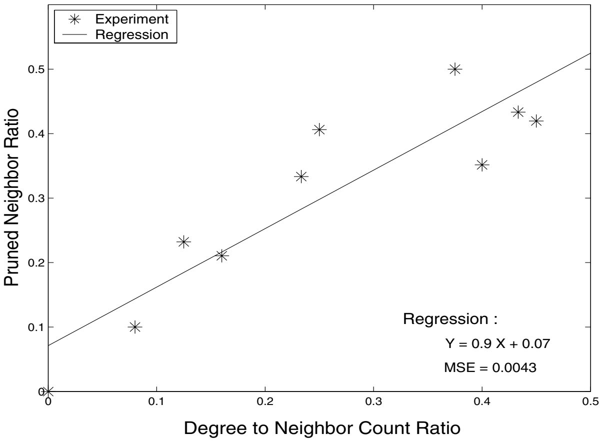

Figure 8: *Pruning performance affected by degree to neighbor count ratio. MSE is Mean Square Error of the shown regression line.*

Sniffing is often used for performance analysis in a WLAN, especially for management frames [9] [13] [33]. One advantage of sniffing is its independence from implementations. However, sniffing fails to measure latencies inside the system. The actual probing latency begins with the internal state transition from a normal state to a probing state, not with the end of successful transmission of the first probe request frame. Our approach uses internal measurements from the roaming program rather than sniffing.

5.4 Experimental Process

Our experiments consists of the following three parts.

1. *Generation of the neighbor graph*
2. *Generation of the overlap graph*
3. *Measurement of the probing latencies.*

For the generation of the neighbor graph, we use the dedicated NG generation phase as described in section II-B-(1). We first construct the neighbor graph inside the station by roaming throughout the 2nd and 3rd floors so as not to miss any possible reassociation edges. For the generation of the overlap graph, we issue 475 random overlap tests while roaming throughout the area. The mobile station measuring the probing latencies is aware of the neighbor and overlap graphs and uses them in the probing algorithms. For the measurement of the probing latencies, we induce 250 hand-offs for each of the four different algorithms over all possible reassociation edges. For identical conditions across the algorithms, we follow exactly the same path for each algorithm.

5.5 Experimental Results

5.5.1 Neighbor Graph and Overlap Graph

Fig. 4 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 in both the neighbor graph and the overlap graph. The direction of an arrow shows the direction of the hand-off relationship. A dashed line represents an edge only in the overlap graph. We find that the number of neighbors is 3.15 on average with a maximum of 6 while the average neighbor channel count is 2.25.

5.5.2 Probing Latencies and Adjusted Latencies

The measured probing latencies are shown in Fig. 5. The x-axis shows the four different algorithms tested and the y-axis shows the probing latencies in milliseconds. The left bars are the results of our experiments while the right bars show adjusted probing latencies that one could achieve when the switching channel and transmission overhead was $10ms$, which could be the case if the algorithm was implemented in firmware. From the experiments, the NG algorithm reduces the probing latencies of full-Scanning and observed-Scanning by 80.7% and 30.8%, respectively. Whereas the NG-pruning algorithm reduced the latencies of full-scanning and observed-scanning by 83.9% and 42.1%, respectively. The adjusted probing latencies show that the probing latency can be less than $50ms$ when using our algorithms.

5.5.3 Probing Counts and Probe-Wait Time

The analysis of the probing counts and the probe-wait times are provided by Fig. 6 and 7. The NG and NG-Pruning algorithms reduce the probing counts of full-scanning by 77.45% and 79.82%, respectively while reducing observed-scanning by 17.33% and 26.00%, respectively. In addition, our algorithms also minimize the number of MaxChannelTime expirations and maximize the number of optimal-wait probings. In Fig. 6, more than 97% of the time is the result of MaxChannelTime expirations in both full-Scanning and observed-Scanning algorithms. However, the ratio of optimal waiting constitutes 46.3% in NG, and 72.0% in NG-Pruning, resulting in improvements in latency. In the experiments, MaxChannelTime expires in $11ms$ and MinChannelTime in $7ms$ as recommended in [9]. Optimal-wait probing is measured to be $2.7ms$ on average with standard deviation of $1.4ms$.

Fig. 7 illustrates the cumulative distributions of the probe-wait time for each algorithms. The graph clearly shows that NG and NG-Pruning have much shorter probe-wait times than the other two algorithms. For example, probings that take less than $7ms$ occur 46.6% of the time in NG and 71.2% of the time in NG-Prunings. The jumps at $7ms$ and $11ms$ explain the high density around MinChannelTime ($7ms$) and MaxChannelTime ($11ms$).

Table 2: Constants used in Simulations

Constants	Values
$MaxChanTime$	11 ms
$MinChanTime$	7 ms
Round Trip Time (RTT)	2 ms
Channel Switch & Transmission	5 ms

Table 3: Varying Parameters in Simulations

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

5.5.4 *Pruning performance vs the number of neighbors*

NG-pruning outperforms the NG algorithm by excluding some APs by using the non-overlap graph. Fig. 8 shows that the higher degree the non-overlap graph has, the more the NG-Prune algorithm outperforms the NG algorithm. The x-axis is the ratio of the degree to the neighbor count and the y-axis is the ratio of the number of pruned APs to the neighbor count. In the pruning algorithm, only the local non-overlap graph is considered. The degree in the graph is the average number of non-overlap degrees in the local non-overlap graph. The figure shows that the ratio of the degree to the number of neighbors directly affects the performance of the pruning algorithm, measured by the ratio of pruned AP with respect to the number of neighbors.

6. 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.

6.1 Simulation Model

The following assumptions are made for simplicity.

1. The radio coverages of all APs are identical circles centered by the serving 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 :

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

   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

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

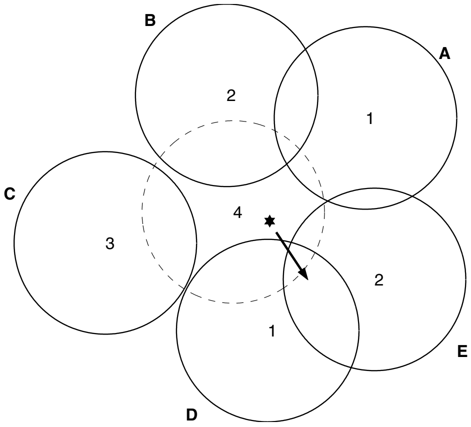

Figure 9: Example Topology Generated by Simulations. The dashed circle is the current cell, solid circles are the neighbor cells. The number in the circle is the assigned channel and the alphabet is the label of the cell. The station, represented by a star is moving toward the direction of arrow.

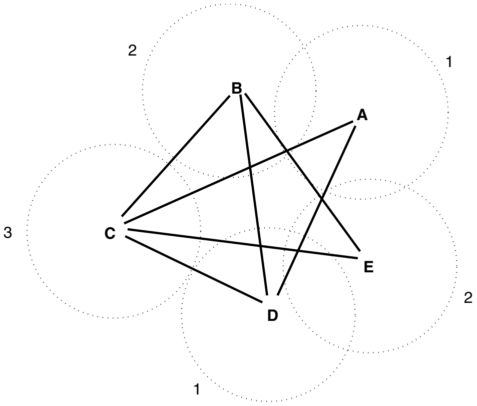

Figure 10: Example of a Non-overlap Graph Generated in Simulations

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 2

6.2 Simulation Process

Table 3 shows the varying parameters used in the simulations. For each combination of parameters, ten different local topologies are randomly generated according to the model. Channels are assigned to the APs according to the following rules :

- The channel used by an AP is not assigned to any of its neighbors.

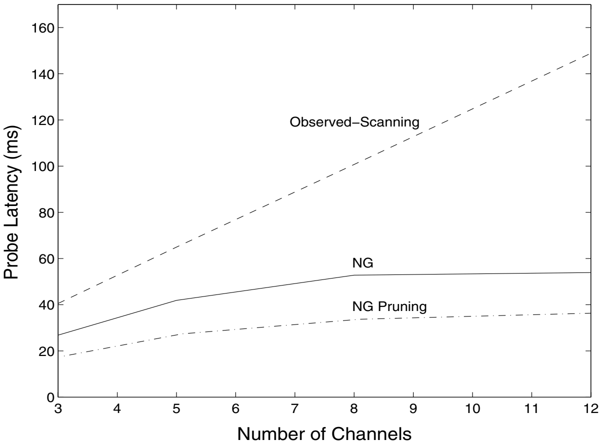

Figure 11: *probing latencies of three algorithms vs. the number of channels*

Table 4:

Channel Counts	NG	NG-Pruning
3	33.8 %	56.1 %
8	47.6 %	66.5 %
12	63.8 %	75.6 %

- If *channel count > neighbor count*, then assign distinct channels to the neighbors while leaving one channel for the current AP
- 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 a topology and channel assignment are determined, the mobile station makes ten different handoffs toward randomly chosen directions and measures the probing latencies using the four different algorithms. Figure 9 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. Note that the neighbor cells need not cover all the boundaries of the current AP.

Figure 10 shows the generated local non-overlap graph from the above topology with average degree of 2.8.

6.3 Simulation Results

6.3.1 Increasing Number of Channels Improves Performance

Figure 11 clearly shows that the performance of our algorithms improve when the number of independent channels increases. While table 4 shows the reduction percentage of the probing latencies compared to the observed-scanning algorithm for three different channel counts.

Table 4 shows that the reduction of latency grows almost linearly with a coefficient of 3.48 for NG and 2.26 for NG-Pruning, obtained by a linear regression using the least square method.

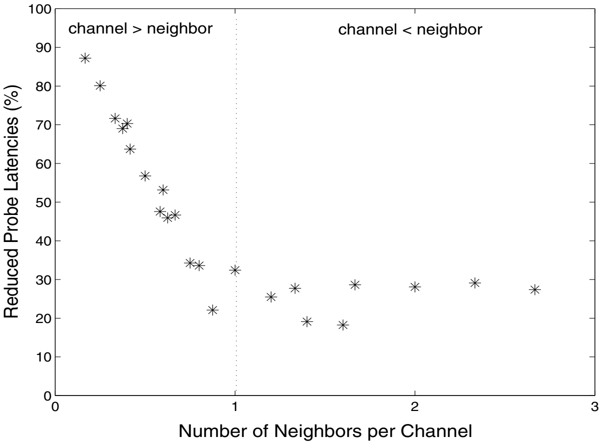

Figure 12: *Performance of the NG algorithm over observed-scanning, as a function of Neighbor-per-channel Density*

6.3.2 Smaller Neighbor-per-Channel Density Helps

Neighbor-per-channel density is the average number of neighbors per channel. The smaller this value is, the better the NG algorithm outperforms the observed-scanning algorithm, shown in Fig.12. But when the neighbor count grows above the channel count (on the right side of the vertical dotted lines), the local channel reuse increases resulting in less performance gain of the NG algorithm over observed-scanning.

6.3.3 Better Pruning with Increasing Number of Neighbors

Figure 13 shows the performance differences between NG and NG-pruning versus neighbor counts. As shown in the graph, the pruning algorithm performs better with more neighbors. This implies that increasing the neighbor counts promotes the ratio of the degree of the local non-overlap graph to the neighbor counts. This is the case when the number of overlapping edges per each neighbor is limited to a small number, for example, two in our simulations. Let OD denote the upper bound of this overlapping degree and NB denote the neighbor count. Then, the ratio of the non-overlap degree to the neighbor count is expressed as,

$$\frac{NB - OD}{NB} = 1 - \frac{OD}{NB}$$

which grows as the neighbor count increases. Figure 14 shows the relationship of the pruning performance and the non-overlap degree to the neighbor count ratio. Similar but extended results of the experiments are shown in Figure 8

7. RELATED WORK

Handoffs in WLAN have been consistently reported to be inefficient in the literature [10] [9] [6]. [10] measures the impact of the handoff process on layer-3 bandwidth. They observed that probing on every second reduces the bandwidth of on-going communication by 15%. Also, a UDP stream was delayed for 1 second while the station handoffs. More

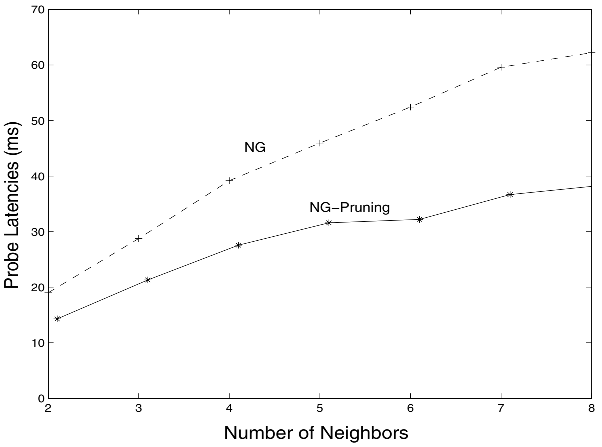

Figure 13: ***The performance of pruning by number of neighbors***

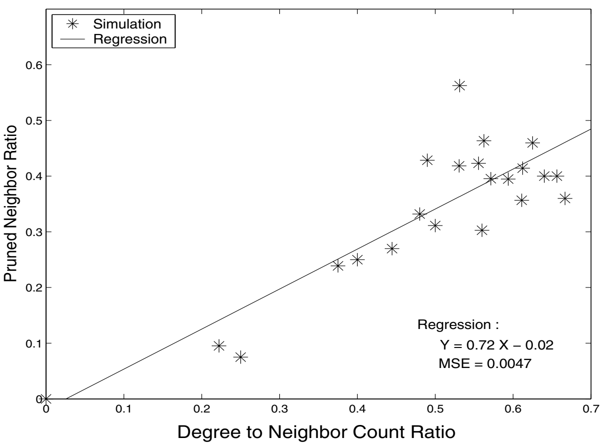

Figure 14: ***Probing Latencies with different Probing Algorithms, and Adjusted Probing Latencies with Chanel-switch-Tx overhead = 10*** *ms*

precise measurements of Layer-2 handoffs were reported in [9]. By sniffing the medium while the station handoffs, they measured handoff latencies between $53ms$ to $420ms$. Among the components of handoff latency, the probing latency is identified to be the dominating factor. These experiments also showed that handoff latencies vary significantly depending on the vendor equipment used. [6] reports the consistent results by measuring Layer-2 handoffs from $120ms$ to $158ms$. In addition, they measured Layer-3 (Mobile IPv6 [26]) handoff latencies between 2.9 and 4.7 seconds.

Previous studies have tried to reduce handoff latency in various aspects. Fast handoff schemes in layer-3, especially with Mobile IPv6 have been suggested in [34] and [6]. [34] and [6] exploit relatively fast layer-2 handoff to facilitate layer-3 handoff. In [34], rather than waiting for a routing advertisement(RA) or simply increasing the frequency of the RA, the scheme allows the layer-2 of the station to initiate router solicitation (RS), explicitly asking for the start of layer-3 transition. [34] also employs tunneling between access routers to avoid packet loss.

To reduce layer-2 handoff, [9] provides several fast-handoff strategies such as reducing the probe-wait latency to an optimal value. Our scheme adopts the suggested values : $7ms$ for MinChannelTime and $11ms$ for MaxChannelTime.

NeighborCasting in [35] provides a distributed and dynamic data structure that maintains the list of candidate foreign agents (FA) to proactively forward data packets during handoffs. In the NeighborCasting mechanism, FAs learn about their neighbor FAs by allowing the mobile stations to report the identity of the old FA to the new FA. The new FA, then notifies the old FA of their neighborhood relationship. Using layer-2 triggered data forwarding, layer-3 handoff latency is reduced to be comparable to layer-2 handoff latency.

[13] and [25] transfer or distribute security material to access points using neighbor graphs. In [13], the security context contained in current AP is proactively distributed to its neighbor APs through the IAPP protocol [27], so that the security context is always one hop ahead of the station's movement. In contrast to [13], [25] maintains the neighbor graph in a centralized server. Instead of generating a PMK for 802.11i authentication [24] at each handoff with the high cost of $800ms$, the authentication server proactively distributes PMKs to neighbor APs, reducing the authentication latency down to less than $20ms$. To maintain the freshness of the PMKs, each distributed PMK is generated by a hashing tree.

8. DISCUSSION AND CONCLUSION

In this section, we provide additional methods for the generation and maintenance of the neighbor and overlap graphs.

8.1 Generation of Neighbor Graph using Overlap Graph

To reduce the NG generation period, we propose to use the overlap graph as the initial neighbor graph. The generation of the overlap graph can be completed much faster than the neighbor graph because the overlap test does not require mobility of the stations. For the fast generation of the overlap graph in the initial phase, the overlap test probability should be high at first and decrease in time. Noting that the OG is a superset of the mobility graph in equation 4, i.e. OG does not have any missing edges but does have some redundant edges. Such redundant edges are a pair of APs that overlap each other but no handoff is possible, called overlap-only edges. In our experiments, we find overlap-only edges are 12% of all overlap edges, and 35% if inter-floor edges are included. These redundant edges will time-out and be discarded over time. Once a mobile station makes a hand-off along an edge in the OG, the system adds that edge to a confirmed neighbor graph. In this way, the overhead of the initial NG will decrease to zero.

8.2 Change of Topology

The quality of neighbor graph can be impaired by significant topology changes such as the addition of a new AP or the deletion (or failure) of existing APs. When an AP is deleted, all the edges to that AP will be evicted due to their inactivity. This leaves some redundant and isolated nodes in the NG but with no impact on the performance. The addition of a new AP will also be detected by the overlap tests. Once a new AP is found responding to the overlap tests, all other APs overlapping with the new AP will add an edge to the new AP.

8.3 Future Work

The NG reflects the aggregated mobility patterns of individual users. However, the mobility pattern of each user can be significantly different because of personal habits, location of residence, or even the social relationships between users. In cellular networks, studies show that an individual user's mobility is routine and predictable with higher accuracy than system-wide approaches [36].

A *personal neighbor graph* (PNG), defined in section III-B-2, is a neighbor graph that reflects the mobility of a particular user. A PNG has many encouraging properties : (i) minimum overhead and (ii) highly predictable. In the future, we plan to research a prediction based probing scheme using PNG. Also we plan to investigate a hierarchical system with NG and PNG for an efficient, accurate and flexible framework for mobility management in wireless networks.

8.4 Conclusion

The main contribution of this paper is to introduce two novel and efficient probing algorithms, the NG algorithm and the NG-pruning algorithm, both of which perform better than the two most commonly used probing schemes by major vendors. We implemented four different algorithms (full-scanning, observed-scanning, NG and NG-pruning), applied in a campus-wide 802.11b WLAN, and measured their probing performances. For more general results, simulations with different configurations were conducted and analyzed. Also we discussed the generation, and maintenance of neighbor and non-overlap graphs to show the feasibility of our suggested probing algorithms.

The results of our experiment show that the NG algorithm reduces the probing latencies of full-scanning and the observed-scanning algorithms by 80.7% and 30.8%, respectively. Also NG-pruning reduces the probing latencies of full-scanning and observed-scanning by 83.9% and 42.1% respectively. By adjustments to the experimental overheads to account for our user space implementation rather than a firmware implementation, we claim that our algorithms can achieve probing latencies of less than 50ms, which provides a strong foundation for fast hand-offs in WLAN supporting VoIP and other multimedia applications.

From experiments and simulation results, we conclude that the algorithms perform best when there are an abundant number of independent channels available such as in 802.11a.

9. REFERENCES

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

[2] V. Nee, "New High-Rate Wireless LAN Standards," *IEEE Communications Magazine*, vol. 37, pp. 82–88, Dec. 1999.

[3] D. Corner, J. Lin, and V. Russo, "An Architecture for a Campus-Scale Wireless Mobile Internet," Tech. Rep. CSD-TR 95-058, Purdue University, Computer Science Department.

[4] A. Hills and D. Johnson, "A Wireless Data Network Infrastructure at Carnegie Mellon University," *IEEE Personal Communications*, vol. 3, pp. 56–63, Feb. 1996.

[5] P. Bahl, A. Balachandran, and S. Venkatachary, "Secure Wireless Internet Access in Public Places," in *Proceedings of IEEE Internationa Conference on Communications 2001*, June 2001.

[6] T. Cornall, B. Pentland, and P. Khee, "Improved Handover Performance in Wireless Mobile IPv6," in *Communication Systems, 2002. ICCS 2002. The 8th International Conference on*, vol. 2, pp. 857–861, Nov. 2002.

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

[8] 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*, vol. 4, pp. 2594–2598, Oct. 2001.

[9] A. Mishra, M. Shin, and W. A. Arbaugh, "An Empirical Analysis of the IEEE 802.11 MAC Layer Handoff Process," *ACM Computer Communications Review*, Apr. 2003.

[10] F. K. Al-Bin-Ali, P. Boddupalli, and N. Davies, "An Inter-Access Point Handoff Mechanism for Wireless Network Management: The Sabino System," in *ICNN 2003*, 2003.

[11] Lucent Technologies Inc., "IEEE 802.11 Channel Selection Guidelines," Tech. Rep. WaveLan Technical Bulletin 003/A, Nov. 1998.

[12] Lucent Technologies Inc., "Roaming with WaveLAN/IEEE 802.11," Tech. Rep. WaveLan Technical Bulletin 021/A, Dec. 1998.

[13] A. Mishra, M. Shin, and W. A. Arbaugh, "Context Caching using Neighbor Graphs for Fast Handoffs in a Wireless Network," in *IEEE Infocom 2004*, Mar. 2004.

[14] T. S. Rappaport, *Wireless Communications*. Prentice Hall PTR, 2002.

[15] Clint Smith et. al, ed., *3G Wireless Networks*. McGraw-Hill Telecom, 2002.

[16] 3GPP, "Technical Specification Group Services and System Aspects; Vocabulary for 3gpp Specifications (Release 6)," Tech. Rep. 3GPP TR 21.905 v6.4.0, Sept. 2003.

[17] G. P. Pollini, "Trends in Handover Design," *IEEE Communications Magazine*, Mar. 1996.

[18] M. Gudmundson, "Analysis of Handover Algorithms," in *IEEE Vehicular Technology Conference, VTC91*, pp. 537–542, 1991.

[19] N. Zhang and J. M. Holtzman, "Analysis of Handoff Algorithms using Both Absolute and Relative Measurements," *IEEE Transactions on Vehicular Technology*, vol. 45, pp. 174–179, Feb. 1996.

[20] H. Aida, Y. Tamura, Y. Tobe, and H. Tokuda, "Wireless Packet Scheduling with Signal-to-Noise Ratio Monitoring," in *25th Annual IEEE Conference on Local Computer Networks (LCN'00)*, Nov. 2000.

[21] W. A. Arbaugh, N. Shankar, J. Wang, and K. Zhang, "Your 802.11 Network Has No Clothes," *IEEE Wireless Communications Magazine*, Dec. 2002.

[22] IEEE, "Standards for Local and Metropolitan Area Networks: Standard for Port Based Network Access Control," *IEEE Draft P802.1X/D11*, Mar. 2001.

[23] B. Aboba and D. Simon, "PPP EAP TLS Authentication Protocol," *RFC 2716*, Oct. 1999.

[24] IEEE, "Draft Amendment to STANDARD FOR Telecommunications and Information Exchange Between Systems-LAN/MAN Specific Requirements. Part 11: Wireless Medium Access Control and Physical Layer(PHY) Specifications: Medium Access Control (MAC) Security Enhancements," *IEEE Standard 802.11i*, May 2003.

[25] A. Mishra, M. Shin, and W. A. Arbaugh, "Pro-active Key Distribution using Neighbor Graphs," *IEEE Wireless Communications Magazine*, Feb. 2004.

[26] 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)*, Jun. 2002.

[27] IEEE, "Draft 5 Recommended Practice for Multi-Vendor Access Point Interoperability via an Inter-Access Point Protocol Across Distribution Systems Supporting IEEE 802.11 Operation," *IEEE Draft 802.11f/D5*, Jan. 2003.

[28] J. Yee and H. Pezeshki-Esfahani, "Understanding Wireless LAN Performance Trade-Offs," *Communication Systems Design*, pp. 32–35, Nov. 2000.

[29] D. S. Johnson, "Approximation Algorithms for Combinatorial Problems," in *Proceedings of the Fifth Annual ACM Symposium on Theory of Computing*, pp. 38–49, 1973.

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

[31] R. Baird and M. Lynn, "Airjack Driver." http://802.11ninja.net/airjack.

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

[33] J. Yeo, S. Banergee, and A. Agrawala, "Measuring Traffic on the Wireless Medium: Experience and Pitfalls," Tech. Rep. CS-TR 4421, Dec. 2002.

[34] R. Koodli, "Fast Handovers for Mobile IPv6," *Internet Draft draft-ietf-mobileip-fast-mipv6-08.txt, Internet Engineering Task Force (IETF)*, Oct. 2003.

[35] E. Shim, H. yu Wei, Y. Chang, and R. Gitlin, "Low Latency Handoff for Wireless IP QoS with NeighborCasting," in *Communications, 2002. ICC 2002. IEEE International Conference on*, vol. 5, pp. 3245–3249, Apr. 2002.

[36] L. Perato and K. Al Agha, "Handover Prediction: User Approach versus Cell Approach," in *Mobile and Wireless Communications Network, 2002. 4th International Workshop on*, pp. 492–496, Sept. 2002.