LeapFrog: Fast, Timely WiFi Handoff

Yanfeng Zhang, Yongqiang Liu, Yong Xia, Quan Huang
NEC Laboratories China
{zhangyanfeng, liuyongqiang, xiayong, huangquan}@research.nec.com.cn

***Abstract*—Widely used wireless LAN infrastructure often deploys multiple access points to cover a large area. A user terminal needs to handoff between these access points as the user moves. Unfortunately, the IEEE 802.11 standard introduces large latency during the handoff process (> 300ms), which greatly affects the performance of mobile applications. Worse yet, it only attempts a handoff when wireless signal quality degrades to a point where current connectivity is threatened. From the application point of view, a desirable WiFi handoff scheme should have sufficiently low handoff latency to support even the most demanding real-time applications, make timely handoff decisions to always connect the terminal to the access point of the best signal quality and/or the lowest load, not incur significant extra power consumption at the terminal, and facilitate easy deployment and be backwards compatible with the 802.11 standard. Very few existing proposals have all these properties. This paper presents the design, implementation, and experimental evaluation of a scheme called LeapFrog that, with efficient, proactive WiFi channel probing, achieves all the four properties.**

Keywords — IEEE 802.11, Handoff, Channel Probing, Mobility

I. INTRODUCTION

The IEEE 802.11-based wireless LAN (WLAN) products have recently been experiencing an unprecedented growth. Since these products work in the unlicensed frequency bands, their power is limited (lower than 100mW) by the regulation agencies. Consequently, multiple WLAN access points (APs) have to be deployed to cover a large area. To keep continuous connectivity, a user terminal has to handoff its connection from one AP to the next when the user moves in the area.

The 802.11 standard (also called WiFi) handoff mechanism works as follows. A WiFi terminal continuously monitors the signal quality of its current serving AP. When the signal quality degrades below a pre-defined threshold (e.g., -90dBm), the terminal disconnects from the current AP, and then probes all the eleven WiFi channels (i.e., frequency bands). Based on the probing results, the terminal chooses a new AP with sufficiently high signal quality and connects to it. This handoff mechanism, although simple to implement, has two weaknesses that greatly affects mobile applications performance. First, it incurs large handoff latency. Experiments in [6][10] show that the latency is usually over 300ms, which can result in service interruption in delay-sensitive applications, and can also cause packet loss during the handoff process, thereby hurting data applications. Second, when the signal quality of the terminal's serving AP is still higher than the pre-defined handoff threshold, even if there exists another AP with far better signal quality (and, potentially, higher throughput), the terminal is unable to handoff to the latter, as illustrated in Fig. 1. To enable uninterrupted mobile real-time applications like VoIP over WiFi and large-scale mobile data access, these two obstacles must be removed.

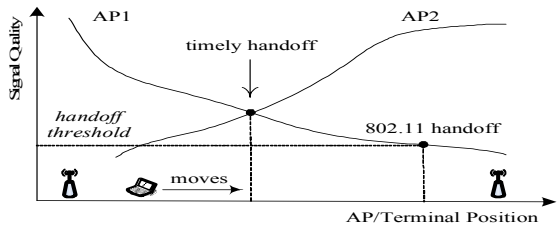

Figure 1. Two WiFi APs serving a user terminal

A number of proposals, with details reviewed in Section V, have been designed to overcome the problems of the 802.11 standard. Before we evaluate these proposals, let's consider what properties a desirable handoff scheme should have. From the application point of view, we believe:

1) The handoff latency, which decides the time during which an ongoing connection breaks, should be sufficiently low. If the latency is higher than 150ms, human users are able to perceive the interruption of real-time communications. On the other hand, higher handoff latency is likely to cause more packet losses during the handoff process, which in turn affects data applications performance;

2) The user terminals should make timely decisions on when to handoff to which AP. Under a typical scenario of multiple APs serving multiple terminals, each terminal should be able to connect to the AP with the best WiFi signal quality and/or the lowest load (or other measures of interest), as shown in Fig. 1. The terminals' overall throughput can therefore be optimized collectively;

3) The handoff process should not introduce significant extra power consumption at the user terminals, simply because the terminals are often battery-powered;

4) The handoff scheme should facilitate easy deployment, and should be backwards compatible with the 802.11 standard. If the scheme makes changes at the AP side, the 802.11 standard terminals should be able to work with the changed AP. If the scheme makes changes at the terminal side, the new terminal should also work with the 802.11 standard AP. This requirement is crucial for the handoff scheme to gain market acceptance, since there are already millions of standard-compliant devices out there.

Few proposals achieve all the four properties above. The pros and cons of these proposals are analyzed in Section V. In this paper, we design and implement a new scheme, LeapFrog, that achieves a good balance between all the properties. In what follows, we continue to discuss at a high level in Section II how to achieve the four goals previously set up, and then present the

1930-529X/07/$25.00 © 2007 IEEE

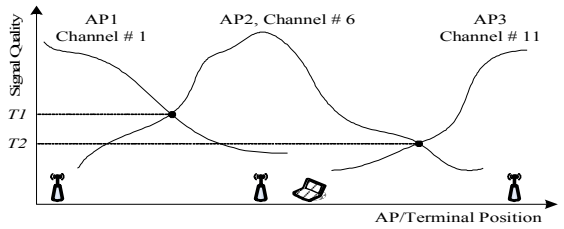

Figure 2. Three APs working in three different channels

design details of our scheme in Section III. We present detailed performance evaluation results in Section IV.

II. Design Guidelines

Before describing the details of our LeapFrog scheme, we present the design guidelines. According to the discussion in Section I, a desirable WiFi handoff scheme has to achieve the four goals listed in Section I.

To see how to achieve the first goal let's take a look at the 802.11 standard handoff process, which probes all the eleven channels right before a handoff action needs to be taken. Probing one channel involves sending and receiving of probing packets over that channel and takes about 15ms. Considering the channel switching delay (10ms), in total the channel probing procedure takes about $11 \times 15 + 10 \times 10 = 265$ms. In contrast, the actual handoff action exchanges a pair of authentication packets and a pair of association packets between AP and terminal, which takes less than 20ms, far shorter than the channel probing time. So, the key to cutting the handoff latency is to significantly reduce the time spent in channel probing. There are several ways of doing this. One is *selective* probing. Instead of blindly probing all the eleven channels, it probes only those used by the surrounding APs. For example, [11] builds in each AP a list of neighboring APs' working channels to do selective probing. Another way is *proactive* probing, which, unlike the 802.11's *reactive* probing, continuously monitors all the channels, even if no handoff is imminent. As a result, when handoff is about to happen, information about which AP/channel to connect is immediately available. Proactive probing completely removes the channel probing procedure from the handoff process, and thus minimizes the handoff latency.

To achieve the second goal, i.e., timely handoff to the AP of the best signal quality, it is necessary for a terminal to monitor the signals of all its surrounding APs. To understand why this is the case, let's see Fig. 2, which illustrates three APs' signal-quality curves. The terminal is currently connecting to the AP in the middle. To always stay on the higher segments of the curves, if the terminal moves leftward (or rightward), it should handoff at the signal-quality threshold *T1* (or *T2*). In general, however, $T1 \neq T2$. So, it is impossible to achieve the second goal by triggering handoff at a *single, pre-defined* signal-quality threshold (as what the 802.11 standard does), because there does not exist such a value in the first place! Instead, if the terminal has a real-time sampling (via proactive probing) of all its surrounding APs' signal quality, it will be able to perform timely handoff at the crossing points of the signal-quality curves (i.e., *T1* and *T2*).

Since proactive probing is crucial to achieving the first two goals, now we proceed to discuss it in further detail, and at the same time analyze how the remaining two goals are affected. Comparing to selective probing, proactive probing comes at a higher cost, since it needs to continuously monitor all the surrounding APs' signal quality, even if the terminal has ongoing communications with its serving AP. The challenge, therefore, is how to implement proactive probing at an acceptable cost. We can think of four approaches to do proactive probing.

a) Each terminal is equipped with one extra WiFi interface, which continuously probes all the channels. This approach, suggested by MultiScan [1], has the following costs: i) One extra interface per user terminal is a high price, since in practice the number of user terminals is huge; ii) The power consumed by channel switching and by the interface to send and receive the channel probing packets;

b) In SyncScan [8], all APs' clocks are synchronized, so do these APs' WiFi beacon messages. A terminal with the same synchronized clock can utilize this feature to collect all the beacons from those APs working in the same channel by switching into that channel and listening for very short period of time. The costs of this approach are: i) The power consumed by the interface to switch between all the channels and to receive the beacons; ii) During the channel switching, both the serving AP and the terminal have to buffer the ongoing sessions' data packets, if any;

c) Unlike SyncScan's synchronized AP/terminal clocks, we can have the terminal switch between all the channels and actively send probing requests to the APs and collect probing replies. Comparing to SyncScan, this approach has one more cost, i.e., sending all the probing requests. Frequently exchanging probing packets between APs and terminals will also lead to lower net throughput achieved by the terminal;

d) We can equip each AP with one extra WiFi interface. This interface is used to broadcast beacon messages across all the channels. Now the terminals can stay working in their current channels and at the same time catch the beacons fed by the APs. Based on the beacons captured, each terminal can compute a real-time sampling of the surrounding APs' signal quality, measured at the terminal. This approach avoids the weaknesses of all the approaches above. The price is the one extra interface *per AP*. Since the number of APs is generally much less than the number of user terminals, this price is much lower than that of MultiScan [1].

In reality, there are already lots of WiFi APs and terminals deployed. In case we can deploy new APs, the approach (d) can be applied. Otherwise, we can use the approach (c). For the former case, an 802.11 standard-compliant terminal can work with the new APs, since these APs' beacons contain necessary information about their own primary, data-carrying interfaces' working channels (see Sections III.A and IV.E). Both the approaches (c) and (d) are implemented in this paper. Users can choose either one according to their own situation.

III. The Scheme Details

Now we describe the details of LeapFrog's two proactive probing approaches introduced in Section II.

A. The AP-based Approach

The key idea behind the AP-based approach is that each AP installs an extra WiFi interface to broadcast information-carrying beacons. The terminals can hear the beacons without switching off its working channel, which in turn eliminates the data buffering cost. To implement this idea, we need to make both AP- and terminal-side changes to the 802.11 standard.

1930-529X/07/$25.00 © 2007 IEEE

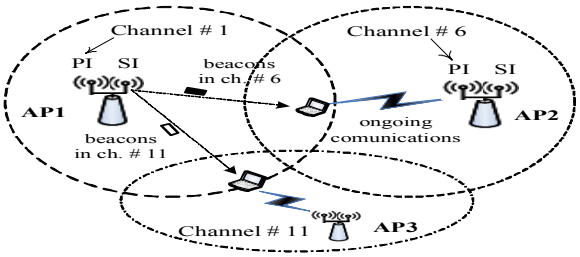

Figure 3. Illustration of the AP-based approach

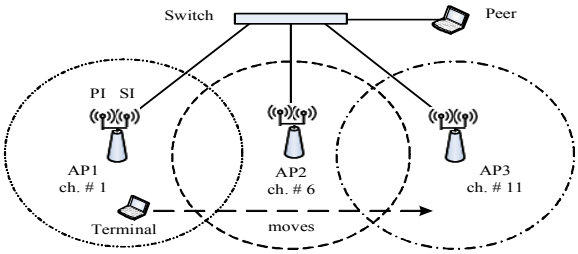

Figure 4. The testbed network topology

1) AP-side Modifications

We use Fig. 3 to illustrate how the changed AP works. As the figure shows, each AP has two WiFi interfaces: one is called primary interface (PI) and the other secondary interface (SI). The PIs of AP1, AP2, and AP3, working in the channels #1, #6, and #11, respectively, provide the normal functions of the 802.11 standard. Namely, the three PIs form a Basic Service Set (BSS) and transmit data packets for their served terminals as well as sends out beacons in their respective channels. For example, the PI of AP2 is serving the upper terminal working in the channel #6. The PI also sends out one beacon every 100ms (beacon interval) in the same channel, according to the 802.11 standard specification.

Now we consider the SIs. The SIs do not carry any data traffic. Instead, it periodically switches between all the eleven channels and broadcasts beacon messages in each channel. So the terminals working in *any* channel can hear some of the beacons. The beacon contains the following information: the AP's BSSID and SSID and the corresponding *PI's* capacity and working channel, etc. For the PI and SI on the same AP, packets sent out from both interfaces travel roughly the same path to a terminal, which can use the SI beacons to monitor the PI's signal quality.

To maintain the same beacon frequency (1 beacon per AP every 100ms) as specified by the 802.11 standard, the SI has to send out 1 beacon every 10ms, since there are ten channels to cover (the whole eleven channels minus the corresponding PI's working channel). This could be a high cost, which can be reduced by the following improvement. Instead of sending out beacons in all the eleven channels, each AP's SI broadcasts only in the working channels of the neighboring APs' PIs. For example, in Fig. 3, the AP1's SI just needs to send out beacons in the channels #6 and #11, one beacon every 50ms. To do this, the discussed AP must know a list of its neighboring APs' PI working channels. Such a list can be manually configured. It can also be automatically constructed with the help of the handoff terminals, since the terminals, during the handoff process, knows the neighboring APs' related information. After the handoff, a terminal can tell the old AP's PI working channel to the new AP. As the users move within the APs' coverage, each AP will gradually build a complete list of the needed information of its neighboring APs. Such method is similar to the "Neighbor Graph" proposed in [11].

2) Terminal-side Modifications

The terminal-side changes are very simple. After capturing a beacon from either its serving AP's PI or the neighboring APs' SIs, the terminal can get a signal-quality sample for one surrounding AP. Since the wireless signal is very oscillating, the terminal applies an exponentially-weighted-time-average (EWTA) algorithm ($sq(n+1) = 3 * sq(n) / 4 + sample(n+1) / 4$) to the raw signal-quality time series to obtain low-pass filtered, smoother curves (see Fig. 5). To avoid ping-pong effect in the handoff decisions, the terminal decides to handoff to a new AP with higher signal quality not at the crossing points of two signal quality curves (as in Fig. 2), but only if the new AP's signal quality surpasses the serving AP's by a positive margin Δ. We set Δ to 5dBm in our implementation.

3) Overhead Analysis

Having described the AP- and terminal-side changes, now we analyze their computational cost. We consider a typical condition, where on average each AP has 6 neighbors and each terminal, including its serving AP, has 4 surrounding APs.

On the terminal side, capturing the beacons from all the surrounding APs does not involve any channel switching. The terminal does not need to send any channel probing packets, but only receives the beacons at a frequency of 4 beacons every 100ms. Calculating four new low-pass filtered signal-quality values with the EWTA algorithm in Section III.A.2 only needs to do 8 bit-wise shifts and 8 additions every 100ms. This is not computationally intensive.

On the AP-side, the beacons from the 6 neighboring AP's SIs, sent in the working channel of the serving AP's PI, at a frequency of 6 beacons every 100ms, occupy some of that PI's transmission time. Since the beacon packet size is about 100 bytes, and it is transmitted at a data rate of 1Mbps (much lower than the 802.11b's peak data rate 11Mbps). The overhead is therefore $6 \times 100 \times 8$ / 1Mbps / 100 ms < 5%, which is a low cost. This cost can be further reduced by compressing some of the fields in the beacons, since the BSSID, SSID and capacity information does not change frequently.

B. The Terminal-based Approach

The AP-based approach discussed above makes changes over the 802.11 standard AP. Sometimes we don't have such freedom (e.g., the APs have been deployed). Therefore, we also design a terminal-based approach. Comparing to the AP-based, the only difference is that the AP-side modifications described in Section III.A.1 (i.e., how to collect at a terminal the signal quality information of all the surrounding APs) now has to be implemented at the terminal side. The functionalities described in Section III.A.2 (i.e., filtering signal-quality curves and hand-off decision-making) needs no change. Due to space limit, here we don't discuss the details. The reader is referred to [13].

IV. Performance Evaluation

Now we present the prototype implementation of the two LeapFrog handoff approaches. We evaluate them using testbed experimental results as well as analysis. Due to space limit, we don't report all the results here, which can be found in [13].

1930-529X/07/$25.00 © 2007 IEEE
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE GLOBECOM 2007 proceedings.

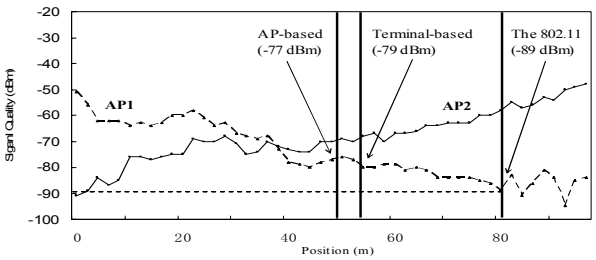

Figure 5. Timeliness of handoff decisions

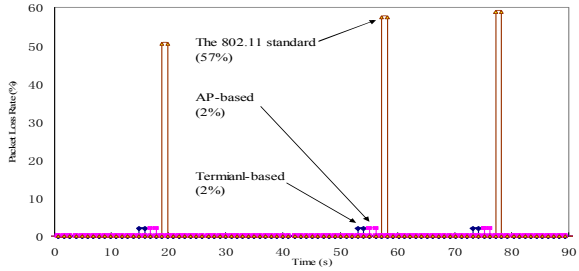

Figure 6. End-to-end packet loss rate

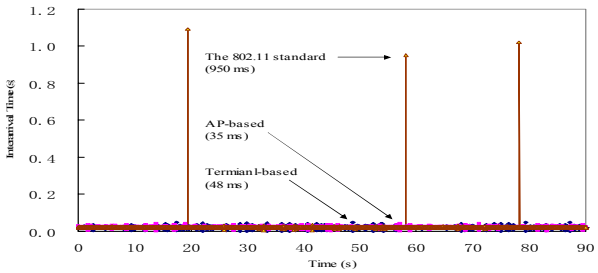

Figure 7. End-to-end packet interarrival times

A. *Prototype Implementation*

Fig. 4 shows the network topology. The testbed locates in a 35m × 20m room in our labs. We build the APs with desktop PCs. Each AP installs two 802.11b/g WiFi cards. One card is the primary interface and provides wireless access to the terminals; the other is the secondary interface to broadcast beacons. For the terminal-based approach and the 802.11 standard, the secondary interface is not used. All the APs run Fedora Linux and the *madwifi* driver [4].

B. *Handover Timeliness*

First we evaluate the handoff timeliness. Fig. 5 shows, when a terminal moves between two APs, the relationship between the handoff time and APs' signal quality. For the 802.11 standard, the terminal does not have the knowledge of its surrounding AP's signal quality before handoff, and can only depend on the absolute value of its serving AP's signal quality to trigger a handoff. So although the neighboring AP can provide better connection quality, the 802.11 terminal is unaware of this and thus does not initiate the handoff until its serving AP's signal quality dropped to around -89dBm.

Using our two approaches, the terminal now monitors the signal quality of all the APs. So it can decide handoff in a more timely fashion. As shown in Fig. 5, the handoff for the AP-based approach happens at the signal quality -77dBm, and the terminal-based at -79dBm, at least *10dBm* higher than the 802.11. Higher signal quality often results in higher throughput for the terminal, which lifts its application performance.

C. *Impact of Handover Latency*

When doing handoff, our scheme does not perform any reactive probing. It only involves exchanging a pair of 802.11 authentication and association packets between the terminal and the AP. Since transmission of each packet takes no longer than 5ms, the total handoff latency is less than 20ms.

To evaluate the impact of the handoff latency, we use indirect measures from the application layer. For this purpose, we send constant bit rate UDP traffic from the mobile terminal to its peer. The packets are generated at a 20ms interval, and the size of each packet resembles a typical 20ms RTP audio packet. We measure packet loss rate and packet interarrival times at the peer side. Three experiments are conducted, using the 802.11 standard handoff, and our AP- and terminal-based approaches. The other experiment conditions are kept the same.

Fig. 6 shows the packet loss rate in each one-second interval, and Fig. 7 shows the packet interarrival times. Three handoffs happen at 20s, 58s, and 78s, respectively. For the AP- and terminal-based approaches, each handoff causes only one lost packet, which contributes to the 2% packet loss rate. In contrast, the 802.11 handoff introduces about 25~30 lost packets and the packet loss rate is 50%~60%. When there is no handoff, the packet interarrival time is between 0~40ms. At the handoff points, it has a sharp increases (949~1093ms) for the 802.11 standard, but keeps very small for both the AP-based approach (0~40ms) and the terminal-base one (0~60ms). The AP-based performs better than the terminal-based, because in the former, the APs do not need to buffer the terminal's data packets, while in the later they do. Buffering introduces delay, thereby increases packet interarrival times.

D. *Terminal Signaling Power Consumption*

Compared to the 802.11 standard, the proactive probing mechanism incurs extra power consumption mainly because the terminal has to send or receive probes/beacons continuously to collect the signal quality of its nearby APs. Experiments have shown that if receiving a packet costs U watt, sending the same packet will cost 2U watt [2]. Note the probe request/reply, the beacons and the PSM packets have similar size. Assume on average each terminal has a serving AP and 3 other surrounding APs, each of them working in a different channel. The terminal needs to sample 10 times per second (i.e., beacon interval is 100ms). Now we compare the terminal signaling power cost of SyncScan [8] and our two approaches.

1) SyncScan: To probe one channel, the terminal needs to send a PSM packet to the serving AP to initiate data buffering and then a PSM packet to flush the data buffer. So in total, the terminal sends $2 \times 3 \times 10 = 60$ PSM packets per second. It also receives $4 \times 10 = 40$ beacons from all the 4 APs during the same time. So, the signaling power consumption at the terminal is $60 \times 2U + 40U = 160U$ watt;

2) The terminal-based approach: Besides all the PSM packets transmitted as in SyncScan, the terminal also sends 1 probing request to each neighboring AP every 100ms – in

total $3 \times 10 = 30$ probing requests and 30 probing replies. It also receives 10 beacons from the serving AP. So the power consumption is $(60 + 30) \times 2U + 30U + 10U = 220U$ watt;

3) The AP-based approach: The terminal does not need to send the PSM packets due to no need to buffer data of its ongoing sessions. It only receives $4 \times 10 = 40$ beacons per second. Therefore, the power consumption is 40U watt.

It can be seen the terminal power consumed by the signaling packets in the AP-based approach is much lower than in SyncScan, which is lower than in the terminal-based approach.

E. Deployment and 802.11 Compatibility

The AP-based approach places most handoff-related functionalities into the AP, and the terminal-side algorithm is very simple to implement. For the networks where the APs have been deployed and can not be upgraded, we can use the terminal-based approach, where the handoff functionalities are all implemented at the terminal side.

Our approaches are completely backwards compatible with the 802.11 standard. A new terminal, upgraded with the terminal-based approach, can work with the 802.11 standard AP, as it is designed so. On the other hand, a new AP, running the AP-based upgrades, can also work with the 802.11 standard terminals. The reasons are: i) When the 802.11 standard terminal is having ongoing communications and does not perform handoff, it filters all the beacons that do not come from its serving AP; ii) When the 802.11 standard terminal is about to handoff and performs channel probing, the captured beacons from the new AP's secondary interface (say, working in the channel #SI) contains information about its primary interface (particularly the channel number, say #PI). If the standard terminal decides to connect to the new AP, it can do so since it knows the primary interface's channel #PI from the beacons.

We have also conducted tests on the handoff performance of the AP-based approach in an outdoor, high moving-speed, real-world scenario. Our tests show that it can support smooth video streaming (1.3Mbps) at a speed up to 160km/h [13].

V. Related Work

A number of proposals have been designed to solve these problems. Some of these focus on reducing the channel probing time and thus achieves fast handoff. The idea is, instead of fully probing all the eleven channels, probing only a subset of them. However, they still perform channel probing in a reactive manner, i.e., channel probing is initiated only when handoff is about to happen. For example, [7] discloses a system using a table of pre-configured neighboring APs to prioritize channel probing, and [9] uses similar selective probing and "AP cache" to achieve the same goal. Using automatically constructed topographical knowledge of AP placement (called neighboring AP graphs), both the number of probes and the waiting time during probing are reduced in [11]. Another method is presented in [12], which lowers the number of probed channels by tracking past user movements.

Reactive probing is unable to timely handoff a terminal to the AP with the best signal quality. To ensure timely handoff, proactive probing, which periodically scans all the surrounding APs and generates a real-time picture of these APs' signal quality information, is necessary. SyncScan synchronizes the clocks of all the APs and the terminals [8], which enables the terminals to capture all the beacons from the APs working in the same channel in a very short period of time. A similar mechanism, proposed in [5], utilizes the beacons heard from overlapped channels to generate some of the signal-quality curves. MultiScan equips each terminal two WiFi interfaces [1]. The terminal uses its second interface to opportunistically scan and pre-associate with a new AP and eventually seamlessly handoffs the ongoing connections, while its first interface keeps communicating with the serving AP. Of course, all these proactive schemes incur additional overhead. MultiScan requires the terminal be equipped with two interfaces. SyncScan requires precise clock synchronization among the APs and the terminals.

VI. Summary

This paper proposes four properties that a desirable WiFi handoff scheme should have, analyzes the key algorithm (proactive probing) to achieve these properties, and designs a scheme to realize the algorithm. Our scheme includes two implementations. One implementation places major functionalities on the access point, while the other is entirely terminal-based. Extensive analysis, indoor experiments and outdoor real-world tests are conducted to evaluate the performance of the scheme. Due to space limit, we leave two issues not discussed. One is the access points' load information, which should be considered when a terminal decides which access point to handoff to. The other is the authentication process during handoff. Our implementation includes both these improvements.

Acknowledgement

The authors wish to thank Mr. Maosheng Ren of Peking University and their colleagues, Dr. Min-Yu Hsueh and Dr. Toshikazu Fukushima, for their help on this project.

References

[1] V. Brik, A. Mishra, and S. Banerjee. Eliminating Handoff Latencies in 802.11 WLANs using Multiple Radios: Applications, Experience, and Evaluation. ACM IMC Conference, 2005.

[2] Card Power Comsuption Websites. (1) http://www.wirelessnetwork.ch/flash/support/fujitsu/connectair_pccard.pdf, (2) http://gicl.cs.drexel.edu/people/tjkopena/wiki/pmwiki.php?n=SWAT.Wi-FiStats, (3) http://www.provantage.com/proxim-8482~7PRXW03K.htm

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

[4] Madwifi driver. http://madwifi.org.

[5] V. Mhatre and K. Papagiannaki. Using Smart Triggers for Improved User Performance in 802.11 Wireless Networks. ACM MobiSys, 2006.

[6] A. Mishra, M. Shin, and W. Arbaugh. An Empirical Analysis of the IEEE 802.11 MAC Layer Landoff Process. Univ. Maryland, Tech. Rep. UMIACS-TR-2002-75, 2002.

[7] PCT. WO 2004/054283 A2. System and Method for Performing a Fast Handoff in a Wireless Local Area Network.

[8] I. Ramani and S. Savage. SyncScan: Practical Fast Handoff for 802.11 Infrastructure Networks. IEEE Infocom, 2005.

[9] L. Ravindranath, J. Prashanth, D. Praveen, L. Prasath, and A. Siromoney. Improving the Latency of 802.11 Handoffs using Sentinel Based Architecture. HiPC Poster, 2005.

[10] S. Shin, A. Forte, A. Rawat, and H. Schulzrinne. Reducing MAC Layer Handoff Latency in IEEE 802.11 Wireless LANs. Workshop on the Mobility Management and Wireless Access Protocols, 2004.

[11] M. Shin, A. Mishra, and W. Arbaugh. Improving the Latency of 802.11 Handoffs Using Neighbor Graphs. ACM MobiSys, 2004.

[12] U.S. Patent No. US 2006/0072507 A1. Minimizing Handoffs and Handoff Times in Wireless Local Area Networks.

[13] Y. Zhang, Y, Liu, Y. Xia, and Q. Huang. LeapFrog: Timely, Fast WiFi Handoff. NEC Labs China Tech Report RN-LC-0011, Dec. 2006.