# Improving the IEEE 802.11 MAC Layer Handoff Latency to Support Multimedia Traffic

Yogesh Ashok Powar and Varsha Apte
Dept. of Computer Science and Engineering
Indian Institute of Technology Bombay
Mumbai 400 076, India
Email: {yogesh, varsha}@cse.iitb.ac.in

***Abstract*— Multimedia applications can be offered to mobile users on the IEEE 802.11 WLAN, if their bandwidth, delay and jitter requirements are met, even in the presence of handoffs. Applications such as VoIP require a delay of less than 150 msecs and packet loss less than 3%. Studies have shown that the latencies achieved by existing handoff implementations can exceed 200 ms and packet loss can exceed 10%. Thus, there is a need for fast handoff solutions, that can meet multimedia traffic requirements.**

**We propose a mechanism for layer-2 fast handoff with the help of background scanning, restricted channel set and preauthentication which does not require any code modification at the access points. The mechanism has been implemented in the MadWiFi wireless open source Linux drivers and tested in an indoor wireless environment. Results have shown that our mechanism achieves latency of less than 10 msec with negligible packet loss even in the presence of handoffs. Thus our mechanism makes mobile multimedia applications possible in IEEE 802.11 WLANs.**

## I. Introduction

The IEEE 802.11 wireless LAN (WLAN) technology has spread rapidly, as it is cheap, and allows anytime, anywhere access to network data. As bandwidth of WLANs has grown, multimedia over WLAN has begun to look feasible. The IEEE 802.11n standard now offers a bandwidth of 100 Mbps which is sufficient for high-quality multimedia. Mobility is a natural consequence of wireless technology especially when used for applications such as voice communication. However, mobility with multimedia communication requires fast seamless transfer (i.e. *handoff*) of the wireless device's connection from one access point to the other.

A Wi-Fi network has access points (AP) to which stations get associated. Data transfer between stations take place through the AP. An AP and its associated mobile stations form a basic service set (BSS). A handoff occurs when a *station* (STA) moves beyond the radio range of its *access point* (AP), and enters another BSS. Management frame exchange takes place between the STA and the AP during the handoff procedure. Consequently, the handoff process incurs some latency, during which the STA is unable to send or receive traffic. Because of the latency involved in handoffs, real-time multimedia applications experience excessive delays and jitter which may be unacceptable. E.g., VoIP expects a one way delay of lesser than 150 milliseconds and packet loss no more than 3%[3]. However, previous studies and our experiments have shown that existing WLAN implementations cannot complete the layer-2 handoff process in less than 150 ms [4], [5].

A major problem with the Wi-Fi technology is the coverage of APs. The radio range of APs is around 200-300 meters in an outdoor environment and just about 50 meters in an indoor environment. These small size cells result in frequent handoffs, potentially causing delay or disruption of communication if the latency of handoff is high. Thus there is a need for fast handoff solutions, so that mobility can be enabled for real-time multimedia communication.

### A. Related Work

Apart from proprietary solutions (e.g. [6]), several layer-2 handoff mechanisms for WLAN have been proposed in recent papers [1], [2], [4], [7], [8].We describe some of the important solutions briefly and compare them.

In SyncScan[7], clients passively scan the channel by switching its current working channels. A beacon broadcast arrangement is done for time synchronization. For a short time after this beacon, the AP does not send any data to the clients which avoids loss of data frames which are destined to the client. By periodic switching to each channel all nearby access points can be discovered and thus eliminating the need of discovery of APs at the time of handoff.

In [1], the authors propose an algorithm in which an AP informs the client about its neighbour list and the client scans only those channels which are in that list.

In Multiscan[8], a dual radio wireless network card is used. One radio is used for packet transmission while the other radio is used for background scanning and pre-associating with alternate APs.

In [2], a client measures the signal strength of received beacon of all the APs operating on the current, and the overlapping channels. Depending on the long term and short term trends in these signals received from the different APs, handoff decision is taken.

The schemes proposed above have some disadvantages. Most need code modification at the APs. They are also not available as open source implementations. Some solutions require multiple wireless cards. This motivated us to build an open source handoff solution which improves on some of the features of previous papers, and presents some new features that improve handoff efficiency. Specifically, we improve the neighbour list mechanism so that no AP-side code modification is required, and add background scanning with *preauthentication* to further reduce handoff latency. The solution requires no modification at AP side and is implemented in the open source MadWiFi[9] linux drivers[1].

The remainder of this paper is organized as follows. In Section II, we present an overview of the design of our fast handoff solution. In Section III, details about our implementation are presented. We discuss the results of our validation experiments in Section IV. We conclude this paper in Section V.

## II. Design of a Fast Handoff solution

Existing implementations check the RSSI of the packets received from the currently associated AP. A handoff procedure is triggered when this RSSI falls below a certain threshold. This procedure consists of the following phases: 1) Probe phase, in which all 802.11 channels are scanned, meaning that a probe is sent, and the RSSI and other information is noted from the response. The AP with the best RSSI is selected as the new AP to associate to. 2) Authentication phase. 3) Reassociation phase, after which the data transfer begins.

Various research papers have already shown that the probing phase (AP scanning phase) is the bottleneck for fast handoff [4], [5]. A station can reduce the probe delay if the list of potential APs and their operating channels is known beforehand. In that case, at the time of handoff, a station can reduce the number of channels to be scanned by scanning only these known channels. This technique was used in [1] and we improve on it in our work.

Our proposed solution rests on three main mechanisms for reducing total handoff latency: background scan, server based restricted channel set and preauthentication; with no modifications at the AP.

### A. Background Scanning

Background scanning, which is available in MadWiFi [9], is the key to faster handoff. It makes use of *power saving* (PS) mechanism defined in IEEE 802.11 standard. In power save mode, the station goes to sleep mode by switching off its radio. In PS mode, the AP buffers all the packets destined to the station. All stations in PS mode are synchronized to wake up at the same time. At this time the AP announces buffered frames for the receiver. A station that receives such an announcement frame stays awake until the frame is delivered.

In background scanning, the client periodically goes in to power save mode for a fixed duration; the duration is also called the *dwell time*. But instead of going in to sleep mode, the client changes its current working channel and starts active scan to find out potential APs in its neighborhood. The information (e.g. RSSI, rate, etc) about the scanned APs is cached is *scan cache*. After finishing an active scan on one channel, or on completion of maximum dwell time, the

[1]Our fast handoff solution is available as a patch on the MadWiFi driver at http://www.cse.iitb.ac.in/perfnet/fasthandoff/

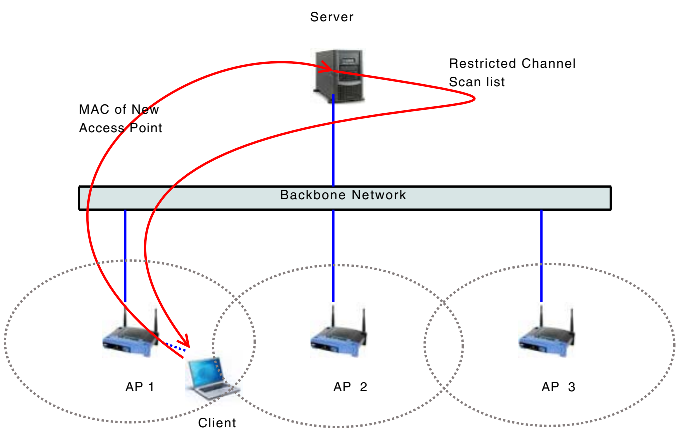

Fig. 1. *Restricted channel list query handling*

device goes back to its original channel and waits to receive beacon frame. If the beacon frame suggests that AP has frames buffered, then the station comes out of power save mode and gather those frames.

The background scan reduces handoff latency drastically, as there is no need to scan APs for their Received Signal Strength Index (RSSI) during the handoff. Instead, the station selects the "best" AP among the potential APs directly from its scan cache.

### B. Restricted Channel Set

The IEEE 802.11 standards have specified 11 channels in 2.4 GHz and 40 channels in 5 GHz. In background scanning, periodic switching to these channels for maintaining the table of potential APs can result in frequent interrupts to data transfer. In our experiments, for higher mobility users, we noticed that cached data is valid only for very short time: typically 3-5 secs. Hence there is need to rescan all the channels frequently.

This overhead can be reduced if the client knows exactly which channels to scan so that it may discover some potential APs. To tackle this problem without having to change anything at the APs, we have introduced a server, called the *NeighBour List Server (NBL Server)* in the backbone network which is configured manually with the neighbour list of each AP. This concept is derived from the neighbor graph mechanism [1]. A neighbor graph is an undirected graph with vertices representing access points. An edge exists between two vertices $X_1$ and $X_2$ if, in any random walk, a mobile station can have a handoff between $X_1$ and $X_2$.

The client-server query handling for getting the restricted channel list is shown in Figure 1. There are multiple APs in the figure. Each of them is connected to backbone network. The client requests the NBL server, which is on the backbone network, for a new restricted channel set every time it gets associated to a new AP. E.g., when the client is associated to $AP_1$ it sends the *MAC* address of $AP_1$ to the server. The NBL server replies to this request with the channel numbers of neighbor APs of $AP_1$, i.e., $AP_2$. In case of the $AP_2$ MAC address, the server will respond with channel numbers of both $AP_1$ and $AP_2$. The client then initiates background scanning only on the restricted channels. Note that the IP address of the NBL server must be configured manually at the client.

### C. Preauthentication

Handoff latency can further be reduced by eliminating the authentication delay if the client "preauthenticates" itself to all the potential APs. Like background scanning, preauthentication also makes use of power saving mechanism defined in the IEEE 802.11 standard.

In our preauthentication scheme, on occurrence of a new potential AP in the scan cache, the client goes in to power save mode for the dwell time. But instead of going in to sleep mode, the client authenticates itself to the new AP found in scan cache. The client then updates the scan cache based upon the authentication response from the AP. Once authentication procedure is finished, or on completion of maximum dwell time, the client goes back to its original channel and breaks the power save mode.

The procedure is repeated for all the potential APs from the scan cache. As described earlier, standard handoff has four phases: scan, authentication, association and run.With our modification, at the time of handoff, the authentication phase is skipped if the client has preauthenticated to the AP earlier and it starts directly from re-association phase.

## III. Implementation of Fast Handoff solution

We now explain the implementation details of the schemes presented in the previous section.

### A. MadWiFi Device Driver

Our proposed mechanisms were implemented in the MadWiFi driver [9], which is a Linux kernel device driver for wireless LAN chipsets from Atheros Communications [10].

The MadWiFi code consists of four main modules: net80211 stack, the Atheros specific 'ath' part, Hardware Abstraction Layer (HAL) and rate algorithms for selecting the best transmission rates. Our modifications have been in the net80211 module.

We now discuss how a potential AP is extracted from the scan cache. We use two main parameters: RSSI threshold and hysteresis, which we explain in the following section.

### B. RSSI Threshold and Hysteresis

MadWiFi uses privacy policy, failure count, max rate, frequency band and RSSI as parameters for choosing the next AP, whenever the RSSI or the rate of current AP falls below a specific threshold. The AP which is better than all the APs in the list is chosen as next potential AP.

The signal strength of the APs change rapidly with respect to space and time. It might happen that just after the handoff, the signal strength of the old AP is better than the current AP and thus the station initiates a handoff with the destination as the old AP. This effect is called the "toggling effect". To avoid this toggling, we have added a mechanism of *hysteresis*; i.e.,

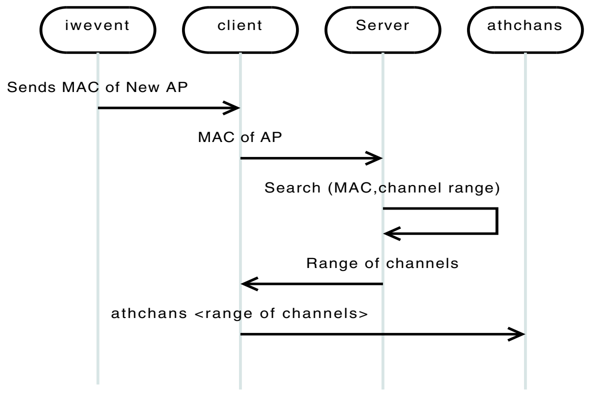

Fig. 2. *Implementation of NBL Server Model*

the signal strength of the new AP must be better than the older one by at least a hysteresis constant. Thus, unnecessary handoffs can be reduced with the use of hysteresis. The parameters RSSI Threshold and hysteresis constant are configurable, and are currently set to 20 db and 5 respectively These numbers were determined experimentally to give best results.

We now explain the implementations of background scan, restricted channel list and preauthentication in the following subsections.

### C. Background Scan

MadWiFi provides the functionality of background scanning since version 0.9.3.2. We have modified this to periodically probe only the neighbor APs by calling the *ieee80211_bg_scan* function. The function calls the *scan_restart* function to initiate the power save mode. The device then goes off-channel (switches to another channel) for a fixed interval that is large enough to scan most APs. Then the *scan_next* function is used to scan the channels with active scan. In active scan, the device sends a probe request frames with blank SSID field and waits for probe responses. The received probe response RSSI values, from all the APs in that channel, are extracted and stored in *scan cache*. After scanning one channel, the device comes to its original channel and waits to receive a beacon frame. If the beacon frame suggests that the AP has frames queued, then the station comes out of power save mode, and gathers buffered frames from the AP. After collecting these frames, the whole process is repeated again for other channels marked for scanning.

### D. Server-based Restricted Channel List

In order to avoid modification of code at AP side, we use the NBL server to retrieve information about the neighbor AP channel list. For implementing this scheme, we have used two utilities implemented in MadWiFi: '*athchans*' and '*iwevent*'.

*1) 'iwevent':* '*iwevent*' implements a user space 'IOCTL' to query the drivers about different types of events such as association with new access point and completion of scan request. In the former case, when a station joins a new access point, or when it loses association, the address of the AP as stated by '*iwconfig*' is reported. Similarly, whenever a

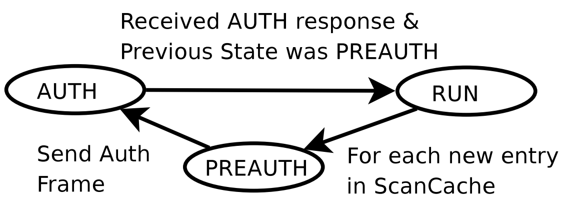

Fig. 3. *Modified IEEE802.11 Finite State Machine inside Driver*

scanning request is completed and results of the scan are available, an event is generated. This utility makes use of IOCTL to request the new access point's MAC address by calling IOCTL with command ID as *SIOCGIWAP*.

*2) 'athchans':* '*athchans*' is a utility developed by MadWiFi developers group to define the channel range for MadWiFi devices. '*athchans*' constrains the set of channels used when scanning for access points. The tool implements user space IOCTL call for command *IEEE80211_IOCTL_SETCHANLIST* which is handled at the driver.

The steps required to get the restricted channel set are shown in Figure 2. In our implementation, the '*iwevent*' is used to capture new AP association event generated by wireless drivers. Every time the client gets associated to a new AP, it first gets the MAC address of the new AP and then queries the NBL server with this address. It gets as response, a list of channels. The client then feeds that list to the '*athchans*' to restrict the scan procedure to look for the new AP from the specified list. The client and server code is implemented in C and in user space.

### *E. Preauthentication*

For implementing preauthentication support in *MadWiFi*, we have added a separated preauthentication module which does the work of preauthenticating the station to all the nearby APs in the background. This algorithm is run every time background scan is called.

1) Periodic background scan starts by calling *ieee80211_bg_scan()*.
2) Station goes into power save mode for scanning different channels by calling *ieee80211_sta_pwrsave()*.
3) Following steps are carried out to preauthenticate to an AP.
    a) *Scan cache*, which is maintained by background scan, is searched to find the next potential AP for preauthentication.
    b) *AUTH* frame is sent on the AP's channel in order to authenticate to the selected potential AP.
    c) The scan cache is modified if the station successfully authenticates the AP (the *'has_preauth'* flag is set).
4) If there does not exist an AP to which client can preauthenticate then the scan cache is periodically searched to find an AP to preauthenticate.

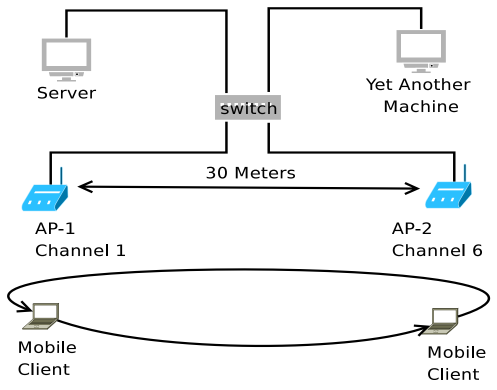

Fig. 4. *Experimental Testbed*

For enabling roaming with preauthentication support we have modified the state machine of *MadWiFi*. We have introduced a new state, called '*PREAUTH*'. Figure 3 shows the added state transitions for preauthentication support in the original state machine. Following are the changes to the original state machine of *MadWiFi*.

1) If current *RSSI* falls below some certain threshold and with given hysteresis then handoff is triggered.
2) During the handoff, the '*has_preauth*' flag is checked from the AP's entry retrieved from the scan cache. If the flag is set then authentication state is skipped.
3) Else standard handoff procedure is carried out.

In summary, the fast handoff solution works as follows:

- As soon as a client associates with a new AP, it requests the neighbour list of this AP from the NBL server, which forms its restricted channel set.
- Background scan is done periodically, in which channels from the restricted channel set are scanned.
- Preauthentication is done periodically to the new APs found as a result of the previous background scan.
- Handoff is triggered only if RSSI falls below a threshold by the difference of the hysteresis constant.
- Once handoff is triggered, the best AP is selected from the scan cache.
- If the AP has been preauthenticated to, the authentication phase is skipped, otherwise, authentication is carried out, followed by the rest of the phases.

## IV. Testbed and Results

We now discuss the evaluation of our handoff schemes. We performed experiments in an indoor environment. We first describe the testbed deployment and then discuss results of different experiments that were performed on the deployed testbed.

### *A. Testbed*

Our wireless mobility testbed consists of two APs, a server and a mobile client. APs are desktop machines with Netgear WG311T PCI cards kept in two different rooms with distance of 30 meters between them. They also have a wired Ethernet connection. We used Madwifi drivers to configure the WNIC to the Master (AP) mode. APs are configured as bridges between the wireless and wired subnet. Because of this, any broadcast on wired will also be a broadcast on wireless side; this can be done with the help of *brctl* Linux Ethernet bridging utility. Transmit power for both of the APs was reduced in order to shrink their cell size so as to trigger the handoff within a short distance from any of the APs. The mobile client is a laptop with Madwifi device drivers on Linux.

The mobile client has a fixed IP as both the APs are in same subnet. The NBL server (fixed IP) is also present in the network.

In every experiment, the client was moved from one access point to another for observing the number of handoffs, handoff latencies and packet loss. We carried out all the experiments at normal walking speed (4-5 Kmph). Initially, the client is associated to AP-1 (see Figure 4) then it starts moving towards AP-2. Once it has reached AP-2, the mobile client comes back to its original position. The client associates to its nearest AP as signal strength from that AP is better than the other. Every time it associates to another AP, it also requests the NBL server for the restricted channel list of the new AP.

## *B. Results*

We now discuss the results of different experiments that were performed on the deployed testbed to measure multimedia traffic performance on the modified drivers. We executed VoIP flows on the testbed.

*1) VoIP Traffic:* VoIP codec G729 generates 50 packets per second with packet size of 66 Bytes. We created similar traffic with the help of the *ping* utility. We generated packets every 20 ms with size of 80 bytes. During the experiment, the client was transmitting and receiving ICMP packets from a machine on the network.

The experiment was performed with the legacy code, i.e. MadWiFi 0.9.3.2, and with our implementation. A graph of the round trip time (RTT) versus sequence number of the packets from the ping traffic is shown in Figure 5. The RTT is in milli-seconds and is plotted in log scale. Significant variations in RTT is the result of high multipath fading in indoor environment, constant change of wireless environment due to mobility and interference. Figure 5(a) shows the performance of legacy code. RTT increased as the station moved away from the associated AP. The legacy code failed to initiate a handoff. In the graph, we can see that, the client had no connection with any AP when packets numbered 3500 to 4000 were transmitted, hence they were lost. The average RTT was 6.15ms, the maximum was 311 ms and the minimum was 0.56 ms. 99% of the packets had RTT below 58ms. 14% of the total packets were lost during this experiment.

Figure 5(b) plots round trip time versus sequence number of the ping traffic with our modified code for fast handoff. During the run, we observed two handoffs. The first handoff was triggered around request number 3200. The second was triggered around request number 7400. When the client triggers a handoff, RTT of the ping request improves. This is because the client associates with an AP with better signal strength than its current AP.

A total of 7938 ping requests were transmitted, of which only one packet was lost. Average round trip time observed was 1.057 ms. The RTT varied from a minimum of 0.366 ms to a maximum of 46.618 ms which is much below the VoIP delay requirement. 99% of the total packets had RTT below 6ms. Hence, VoIP calls can be supported satisfactorily with our fast handoff solution.

*2) Handoff Latency:* We calculated the handoff latency by observing debug output at */var/log/messages*. We used *80211debug* tool to capture all important events from the driver. We calculated handoff latency as the difference in the time at which the decision for joining a potential AP is taken to the time at which the client is successfully associated to the AP. The handoff latencies were measured for (1) legacy code (2) background scan with restricted channel set modification added, but no preauthentication and (3) background scan+ restricted channel set+preauthentication. The observations are listed in Table I. The latency for legacy code is high because each channel is passively scanned with dwell time of at least 150 ms resulting in 1650 ms just for scanning all the 11 channels.

| **Scheme** | **Open Authentication** | **Shared Key** |
|---|---|---|
| **No bgscan** | 1.8 - 2 secs | 1.8 - 2 secs |
| **bgscan** | 8 - 12 msec | 10 - 14 ms |
| **bgscan+preauth** | 7 - 8 ms | 7 - 8 ms |

TABLE I
*Handoff latencies*

Our solution works for open authentication as well as shared key authentication scheme. With background scan and restricted channel set, we observed layer-2 handoff latencies from the range of 8 to 14 ms.

Preauthentication avoids the transfer of two management frames in case of open authentication and four frames in case of shared key authentication. Hence handoff latency was seen to be further improved by 1-6 ms with preauthentication enabled.

To increase the test reliability, we noted the results of twenty handoffs which were recorded one after another while moving from one access point to another, with the traffic consisting of ICMP packets generated at the rate of one per 20ms. The

| **Run** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
|---|---|---|---|---|---|---|---|---|---|---|
| without Preauthentication | | | | | | | | | | |
| **Latency (ms)** | 10 | 10 | 12 | 12 | 10 | 8 | 8 | 12 | 10 | 10 |
| **Packet Lost** | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| with Preauthentication | | | | | | | | | | |
| **Latency (ms)** | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 8 | 7 |
| **Packet Lost** | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |

TABLE II
*Handoff latency and packet loss of 20 successive handoffs*

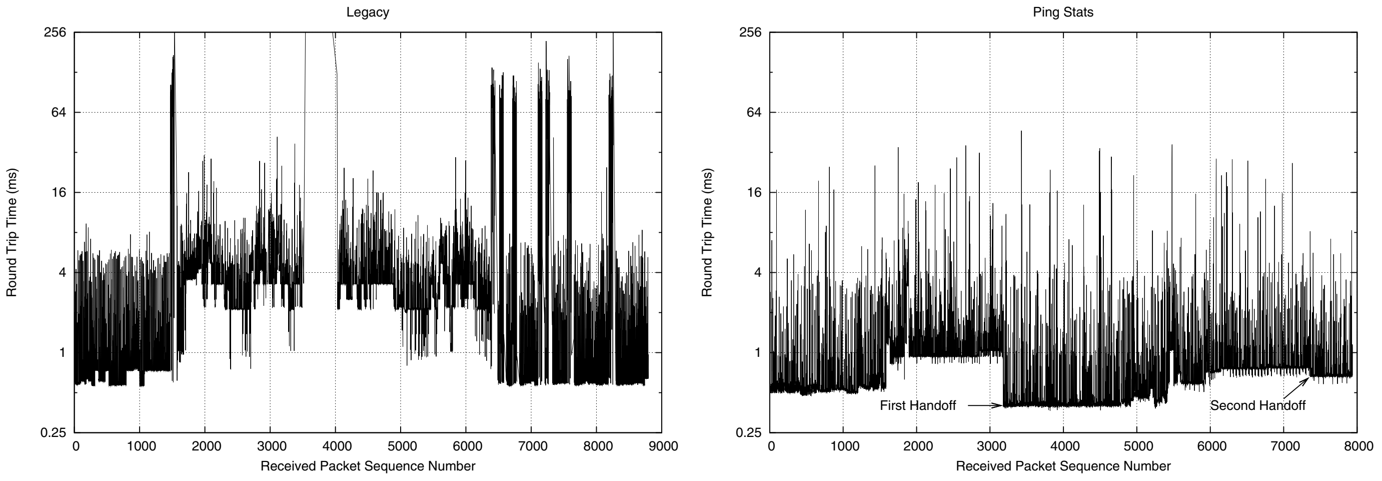

Fig. 5. *Ping Statistics with (a) legacy and (b) our solution*

statistics of this experiment can be seen in Table II.

For the first 10 handoffs preauthentication was disabled. For those handoffs, we observed handoff latencies from 8 ms to 12 ms with average latency as 10.2 ms. 9 handoffs dropped just one packet. We observed one handoff which took just 8 ms and dropped no packet.

For the remaining 10 handoffs, preauthentication was enabled. Almost all handoffs recorded just 7 ms as handoff latency; one handoff took 8 ms. There were 6 occasions where the station did not drop any packet during the handoff. The remaining 4 times, the station dropped one packet each time.

## V. Summary and Future Work

We have designed and implemented a fast handoff mechanism for IEEE 802.11 WLANs, to reduce the latency incurred due to handoff at layer-2 in such a way that requirements of multimedia traffic are met. Our proposed fast handoff solution rests on three main mechanisms for reducing the handoff latency: background scanning, restricted channel list and preauthentication. The solution was implemented in an open source Linux device driver, MadWiFi, and the solution requires only client side modification.

Tests of our fast handoff implementation in an indoor wireless environment have shown very promising results. We observed handoff latency of just 7 ms. We executed VoIP calls and measured one way delay of just 25 ms with negligible packet loss. This makes real-time applications possible even for mobile clients.

There are few improvements that are possible to our solution. E.g., the neighbor list of all the APs is maintained at the NBL server. This list is currently manually configured. This maintenance can be automated if clients report the server with events such as finding of a new AP which was not present in neighbor list or not finding an AP which was present in neighbor list.

Similarly, there is need for efficient use of power saving mechanism for triggering background scan and preauthentication to extend the life of battery.

In the paper, we have not studied the effect of speed of mobility on the handoff latency. Results might vary for different speeds of mobility. Then, the frequency of invoking background scan and preauthentication can be determined based on the speed of mobility.

## Acknowledgements

We would like to thank the Ministry of Human Resource and Development, Govt. of India, for funding this project. This project was a continuation of work done by Anand Parhar (M.Tech. IIT Bombay, 2007) as part of his Masters thesis.

## References

[1] Minho Shin, Arunesh Mishra, and William A. Arbaugh. Improving the latency of 802.11 hand-offs using neighbor graphs. In *MobiSys '04: Proceedings of the 2nd international conference on Mobile systems, applications, and services*, pages 70–83, New York, NY, USA, 2004. ACM Press.

[2] Vivek Mhatre and Konstantina Papagiannaki. Using smart triggers for improved user performance in 802.11 wireless networks. In *MobiSys '06: Proceedings of the 4th international conference on Mobile systems, applications and services*, pages 246–259, New York, NY, USA, 2006. ACM Press.

[3] Hongqiang Zhai, Xiang Chen, and Yuguang Fang. How well can the IEEE 802.11 wireless LAN support quality of service? *IEEE Transactions on Wireless Communications*, 4(6):3084–3094, December 2005.

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

[5] H. Velayos and G. Karlsson. Techniques to reduce the IEEE 802.11b handoff time. In *IEEE International Conference on Communications*, volume 7, pages 3844–3848 Vol.7, 2004.

[6] Meru Networks, 2008. `http://www.merunetworks.com`, last accessed January 2008.

[7] S. Ramani, I. Savage. Syncscan: Practical fast handoff for 802.11 infrastructure networks. *IEEE INFOCOM*, 1:675–684, 2005.

[8] A. Mishra V. Brik and S. Banerjee. Eliminating handoff latencies in 802.11 WLANS using multiple radios: Applications, experience, and evaluation. *ACM IMC*, Oct 2005.

[9] Atheros Communications. The Open Source Project Site for MadWifi. `http://madwifi.org/`.

[10] Atheros Communications. `http://www.atheros.com`, last accessed July 2008.