A Simple Client Software Upgrade Enables Fast Handoff in IEEE 802.11 Wireless Networks

Jaeouk Ok
The University of Tokyo
okjaeouk@mlab.t.u-tokyo.ac.jp

Pedro Morales
The University of Tokyo
pedro@mlab.t.u-tokyo.ac.jp

Hiroyuki Morikawa
The University of Tokyo
mori@mlab.t.u-tokyo.ac.jp

1. INTRODUCTION

Handoff procedure in IEEE 802.11 wireless networks must be accomplished with as little interruption as possible to maintain the required quality of service (QoS). We have developed a fast handoff scheme, called AuthScan, to reduce the time-consuming channel scanning latency. AuthScan comprises two steps: First, a client caches its handoff history with beacon information. Second, when in need of handoff, a client transmits *Authentication Request* frames to the selected Access Points (APs) from the cache instead of broadcasting *Probe Request* frames like in active scan to discover the next AP. Our proposed method does not require any support from the infrastructure network and improve the efficiency of channel scanning. Furthermore, AuthScan requires neither hardware upgrades to the client, nor any modification to currently deployed APs. This paper presents the theoretical handoff latency of AuthScan, the effectiveness of our system through experiments, and the details of our demonstration.

2. AUTHSCAN

AuthScan is a combined approach of a handoff history cache [1] and selective unicast scan [2]. While the former enables a client to scan only the channels where nearby APs exist in habitual places, the latter enables to reduce unnecessarily long waiting time at each channel. AuthScan improves 1) AP selection propriety among a large number of cached APs by comparing acquired signal strengths from each of them right after handoff trigger, 2) handoff latency by replacing broadcast *Probe Requests* with unicast *Authentication Requests* to scan target APs.

2.1 Target AP Cache

AuthScan builds a target AP list by caching previous handoff information. Caching does not require support from infrastructure network, modifications to the protocols, hardware upgrade to any devices, nor impose degradation in the performance of data communication. All these benefits come at the cost of unimproved support for a client's unusual behavior like visiting a place for the first time.

The cache is initially populated at a stage when the handoff is achieved through conventional active or passive scan. The reason for this is that the *Authentication Response* does not contain some information found in *Probe Responses* or beacons. Therefore, for successful completion of handoff, the cache structure includes the BSSID of prior AP, the BSSID of posterior AP, channel number, PHY type, capability information, SSID, the supported rates, PHY parameter sets, and WPA parameters, etc. The recurrence of handoffs during a fixed period is used to rearrange the order of target APs in the list. The cache can be updated by either periodic or on-demand active scan to accommodate the changes of APs in the habitual places.

2.2 Handoff Procedure

When detecting the need for link-layer handoff based on its policy (e.g. signal strength, transmission rate, etc), a client looks up its target AP cache and chooses one as the next target AP. Then, it sets up its interface to the desired channel and PHY type and transmits *Authentication Request* to the target AP.

We define two algorithms: *comparative mode*, and *fast mode*. In *comparative mode*, a client checks all the target APs in the cache, before selecting the AP to handoff. This is done by sending *Authentication Request* to all the target APs, and comparing the signal strength from the received *Authentication Responses*. This is in conformation with the standard which allows authentication with multiple APs.

In the case of *fast mode*, if the signal strength of the received *Authentication Response* is higher than a certain threshold, the client immediately moves to the reassociation phase. If it receives *Authentication Response* whose signal strength is lower than the threshold or there is no response during *MinChannelTime*, it repeats this operation with the next target AP in the cache. In case no AP in the cache can satisfy the client's handoff policy, the client moves to active scan and saves the newly found AP in the cache.

2.3 Handoff Latency

In terms of theoretical delay of the procedure, and assuming at least one of the cached APs is available and fulfills the handoff policy, *comparative mode* takes $M * RTT + (N-M) * \mathit{MinChannelTime}$, where N is the number of target APs in the cache and M is the number of *Authentication Responses* received. In the case of *fast mode*, in the best case scenario the first AP from the cache provides signal strength higher than the threshold, so the delay is $1 * RTT$. When the last

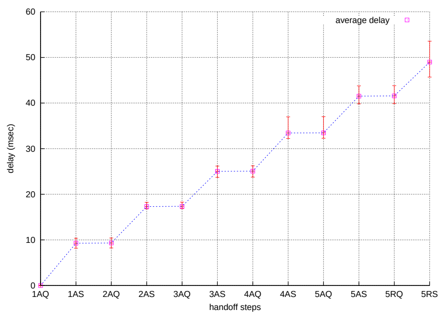

Figure 1: Average handoff delay checking five APs

AP from the cache does so, the delay is the same as that of *comparative mode*.

For example, assume that there are five target APs under open system authentication in the cache and four of them return *Authentication Response*. The highest total handoff latency will be composed of four *RTTs* for the APs with responses, one waiting period of *MinChannelTime* due to the AP with no response and one *RTT* for reassociation phase. Therefore it will be $4 * RTT + 1 * MinChannelTime + 1 * RTT = 4.024\ msec$, where *RTT* is 0.6 msec, *MinChannelTime* is 1024 μsec.

3. IMPLEMENTATION AND EXPERIMENTAL RESULTS

We implement AuthScan in Debian Linux 4.0 Etch with a 2.6.18-5 kernel. We opt for dividing the system into two main parts: changes to the kernel driver, and a userland tool to control the handoff procedure. The driver changes are implemented in the madwifi driver [3], and consists of changing the protocol state machine, adding system calls to manage the driver from the application (*ioctls*) and generation of informational events from the driver to the application. The userland tool is implemented by modifying wpa_supplicant [4], and involves the creation of its own state machine and the addition of the cache logic.

We evaluate the performance of our prototype by measuring handoff delay across BSSs of different PHY types and channels, where six APs are available: AP0(11b, ch. 11), AP1(11a, ch. 42), AP2(11b, ch.14), AP3(11g, ch. 6), AP4(11g, ch. 1), and AP5(11a, ch. 34). In order to get the delay from the user's perspective, we add a checkpoint right before calling the driver *ioctl* for requests, and right after receiving the driver's informational event for responses.

Figure 1 shows the average delay from ten runs of the handoff experiments from AP0 to AP5 since the sending of the *Authentication Request* to the first AP scanned (1AQ in the graph). The x-axis shows the steps in the authentication scanning process. They correspond to the sending of the *Authentication Request* (AQ), reception of the *Authentication Response* (AS), sending of the *Reassociation Request* (RQ) and reception of the *Reassociation Response* (RS). The number before each of them is a sequential number related to authentication scanning steps. The y-axis is the time delay in milliseconds measured at the checkpoints in the application level. The whole handoff delay obtained by checking five APs is 48.97 msec in average with open authentication. We believe that we can make further improvements by moving most of the authentication scanning logic from the userland to the driver.

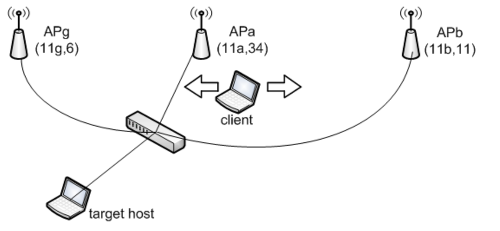

Figure 2: Demonstration setup

4. DEMONSTRATION DETAILS

The goal of our demonstration is to show the performance of our prototype system: a simple client software upgrade performs fast handoff in the IEEE 802.11 wireless networks. The demonstration setup is described in Figure 2. Three PCs are set up as APs with different PHY types and channel numbers. A client with an IEEE 802.11 a/b/g NIC is associated to one of the APs, and performs handoff moving back and forth among the APs. It transmits ICMP Echo Request frames to the target host in the same subnet. We set ICMP frame size as 480 bytes, and interval as 10 msec. At the moment of handoff it scans the other APs in comparative mode before reassociating. The delay caused by handoffs can be observed from the changes of RTTs and the number of lost packets shown on the client's display. The needed space for our demo is a table on which five PCs can be placed with some space to one another. The required setup time is about an hour. The needed facility is power outlets for the PCs.

5. CONCLUSION

We have developed a fast handoff scheme, called AuthScan, that not only achieves fast handoff, but also satisfies the fresh handoff metrics and interoperability with already deployed IEEE 802.11 wireless networks requirements. AuthScan maintains a target AP list cached from a client's previous handoffs, and performs unicast scanning by transmitting not *Probe Request* frames, but *Authentication Request* frames only to the selected APs. AuthScan also provides a novel usage of open system authentication phase which has become redundant with the advent of WPA. In this paper we have shown the theoretical handoff latency, the effectiveness of our system through experiments, and the details of our demonstration.

6. REFERENCES

[1] S. Shin, *et al.*: "Reducing MAC layer handoff latency in IEEE 802.11 wireless LANs," MobiWac, USA, 2004.

[2] H. Kim, *et al.*: "Selective Channel Scanning for Fast Handoff in Wireless LAN using Neighbor Graph," ITC-CSCC, JAPAN, 2004.

[3] "MadWifi - a Linux kernel device driver for Wireless LAN chipsets from Atheros." [Online]. Available: http://madwifi.org/.

[4] "Linux WPA/WPA2/IEEE 802.1X Supplicant." [Online]. Available: http://hostap.epitest.fi/.