Evaluating Fast Handoff Solutions in 802.11

Ethan Goldblum & Marinos Bernitsas

The Problem

- 802.11 does not specify a specific handoff scheme
- The Mobile Host will switch Access Points only when there is a significant deterioration in the received signal
- Typical handoff time ranges from 100ms to several seconds
- Long handoffs disrupt application performance for mobile users switching between Access Points, especially for latency-sensitive applications like VoIP

The Objective

Reduce the time spent when switching between two 802.11 Access Points in the same network

The Approach

- We focus on reducing the **Probe Delay**
- Probe Delay: the time it takes for the Mobile Host to actively scan each channel to find which Access Point it should connect to
  - $t_{handoff}=\boldsymbol{t_{probe}}+t_{assoc}+t_{auth}$
  - Probe Delay accounts for 90% of the handoff delay
- We implement two handoff protocols on the MadWifi Driver for Atheros Chipsets

Beacons are missed and the Handoff begins. The station first tries to reassociate with the old Access Point

Association timeout after 5 seconds, proceed to active probing

After each of the 11 channels is probed (11*200ms), the next closest Access Point is chosen.

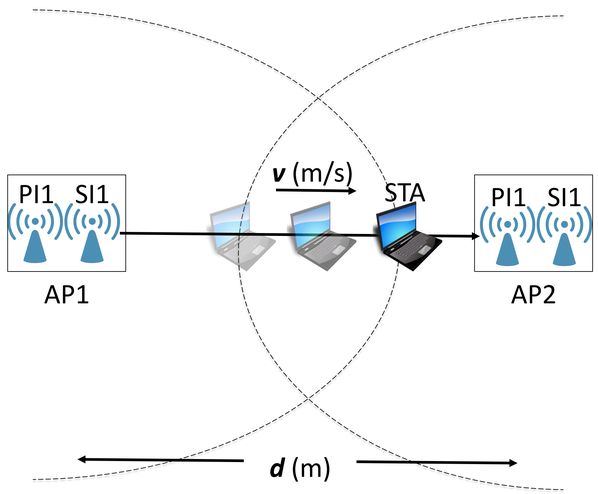
An authentication request is sent to the new Access Point

The Mobile Host associates with the new AP and the handoff is complete

MadWifi state transitions during handoff

Leapfrog

- Each Access Point has two antennas, the **Primary Interface** and the **Secondary Interface**
- The Secondary Interface iteratively sends a beacon to each channel, signifying the presence of the Primary Interface
- The Mobile Host uses the Received Signal Strength Indicator of the beacons sent by the Secondary Interfaces to predetermine the best Access Point to associate with during the Handoff
- Once it is time for the Handoff to take place, the Mobile Host can avoid the probing process and move directly to the Association Process

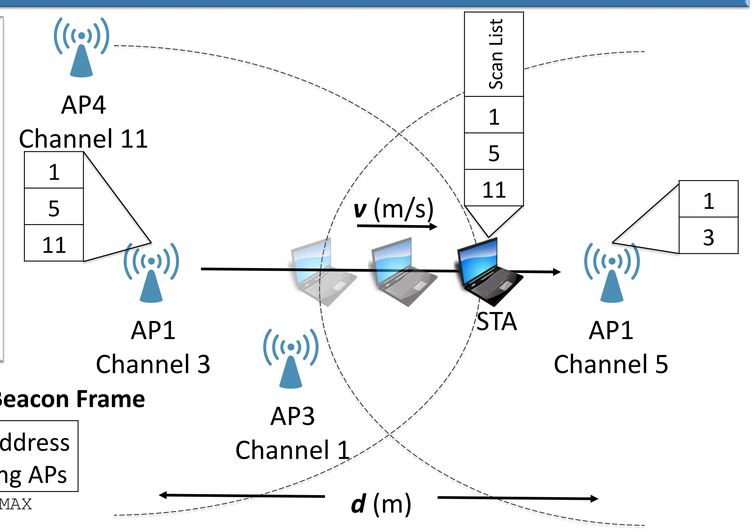


Vendor-Specific Element in Leapfrog's Beacon Frame

Element ID	Length	OUI	Primary Interface's Channel & MAC Address
bytes: 1	1	3	8

Shared Beacon

- Each Access Point is preconfigured with a table of its nearest Access Points
- Every beacon contains the table of nearest Access Points' Channel and MAC Address
- When it is time to handoff to a new network, only the list of nearest Access Points is probed, saving the Mobile Host of having to iterate through all channels



Vendor-Specific Element in Shared Beacon's Beacon Frame

Element ID	Length	OUI	Channel & MAC Address List for Neighboring APs
bytes: 1	1	3	8*SHAREDBEACON_MAX

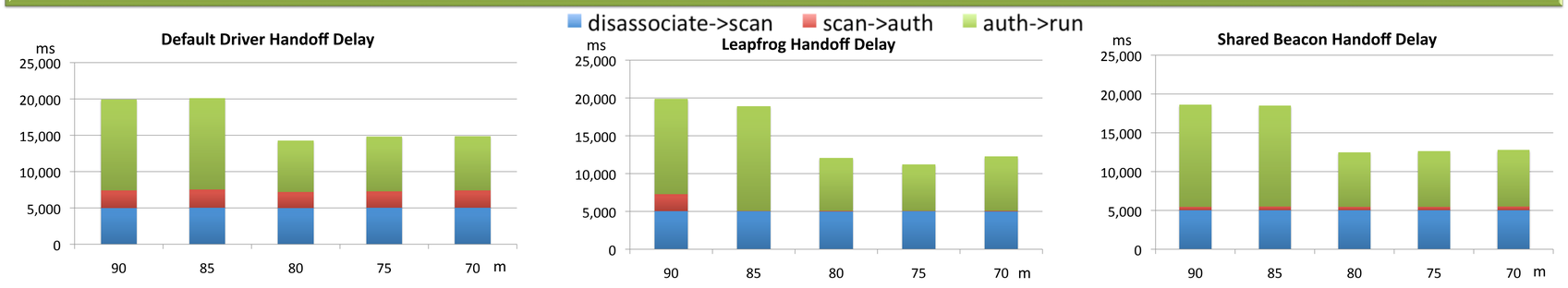
Experimental Setup

- We use the CMU Wireless Emulator to run our experiment
- We first run the stock MadWifi driver to establish our benchmark
- We implement each scheme by modifying the MadWifi Driver's beacon, send/receive and scan procedures
- We use a generic model of signal propagation:
  - LogDistance loss model with **n**=3, Ricean Fading with **k=3** (rooms with brick walls) and **0** additional Delay

Key Variables

- We vary the size of the region for which the two Access Points overlap, by varying the Access Point separation ***d*** (in meters) for a fixed **v** of 1 m/s
  - ***d***>90m: There is no overlap, so leapfrog degrades to default MadWifi driver's performance
  - ***d***<90m: There is an overlap region
  - ***d***<60m: The Mobile Host does not disassociate with the first Access Point

Results



Key Metrics	
t(disassociate->scan)	Time from disassociation to decision to perform scan
t(scan->auth)	Time from scan to transmission of authentication frame to the new Access Point
t(auth->run)	Time from transmission of authentication frame to successful association with the new Access Point

Assessment

- Leapfrog and Shared Beacon managed to decrease the probing delay significantly
  - Unlike Shared Beacon, Leapfrog may not need to probe when there is enough overlap between two Access Point ranges
- Total Handoff Delay decreased, however, since in MadWifi probe delay accounts for the smallest portion of total handoff delay, the overall improvement was minor

Carnegie Mellon Electrical & Computer ENGINEERING