An experimental study of IEEE 802.11b handover performance and its effect on voice traffic

Jon-Olov Vatn

July 2003

TRITA-IMIT-TSLAB R 03:01
ISSN 1651-7709
ISRN KTH/IMIT/TSLAB/R-03/01--SE

Telecommunication Systems Laboratory
Department of Microelectronics and Information Technology (IMIT)
KTH, Royal Institute of Technology
Stockholm, Sweden

An experimental study of IEEE 802.11b handover performance and its effect on voice traffic

Jon-Olov Vatn
KTH, Royal Institute of Technology
P.O. Box Electrum 299
SE-16440 Kista, SWEDEN
vatn@imit.kth.se

Abstract

In this study we present experimental results on the performance of the IEEE 802.11b handover mechanism and its effect in terms of loss and delay on an ongoing voice stream. Our results show that the mobile station (STA) does **not** fully stop communicating during the handover interval, but is able to send and receive some packets via its *old* access point (AP). When a STA (with the "best" WLAN card tested) received voice traffic generated at 20 ms interval during the handover, 4 packets (in median) were forwarded by the old AP, however, the burst of packets containing losses (and occasionally intermediate successful packets) were only 3 (in median). Upstream data were generally *delayed* rather than *lost*, but interestingly the second voice packet buffered during handover was lost for the WLAN card (otherwise) considered best. The delay for the upstream packets formed a spike where the first packet was delayed the most and the delay of the following packets decreased roughly linearly. For the "best" card the first packet was delayed 52 ms on average and 3-4 packets were affected in total. When the STA was subject to competing traffic at the new AP, the delays involved increased. There was also a greater variation in handover behavior in that the STA often started to search for new APs without *executing* the handover (leading to multiple loss burst or delay spikes), and that during a handover more voice data was exchanged between the search and execution phases.

1 Introduction

IEEE 802.11b[5] wireless local area networks (WLANs) have had a tremendous market success as a wireless network extension, replacing wired Ethernet as the access technology of choice in corporate and campus networks as well as at public hot-spots and in home networks. As commonly used IEEE 802.11 can be viewed as a "wireless Ethernet" where the wireless stations (STAs) communicate via an access point (AP) attached to a wired backbone, and the STAs compete for the media access by a distributed medium access control (MAC) protocol referred to as the distributed coordination function (DCF). WLANs have so far been used mostly for ordinary Internet services such as web browsing, file transfer and electronic mail, however, when the usage of IP telephony, often referred to as voice over IP (VoIP), will take off, these WLAN networks will also be used as cordless telephony systems. Furthermore, being able to use this WLAN infrastructure as a mobile network where a STA can perform a handover between different APs, with little or no performance degradation on an ongoing conversation, will become an important feature.

This report considers the IEEE 802.11b handover process and its performance. We consider the case when a mobile STA performs a handover between two APs, and where the APs are attached to the same LAN via an Ethernet switch. The mobile STA is communicating with a host on the wired LAN and the traffic sent between the STA and the host emulates voice data.

1.1 Qualifications

In this study we have limited the scope to concern in-LAN handover, where the APs are configured to belong to the same extended service set (ESS). We have assumed that APs are densely populated and that the STA is configured to take advantage of that (this enables the STA to initiate a handover before losing connectivity with its old AP). We have also assumed that the STA performs *active* scanning when searching for candidate APs as handover targets, and that open system authentication (i.e., no authentication) is used for access control.

In our tests we generate UDP packets with 172 bytes of payload every 20 ms in order to emulate a 64 kbit/s pulse code modulated (PCM) voice stream packetized into 160 bytes units plus a 12 byte real-time protocol (RTP)[21] header. These UDP packets are then carried inside IPv6 packets, although except for the difference in IP header size our measurements should be valid for IPv4 as well.

As access points we have used stationary Linux PCs equipped with WLAN PCI cards and the HostAP WLAN driver[4]. This enabled us to initiate handover by controlling the transmission power of the APs instead of moving the STA.

The outline of the rest of the report is as follows. Section 2 covers related work regarding other experimental WLAN handover studies. Section 3 provides some theoretical background and section 4 describes the test-bed used for the experiments. Section 5 presents the results of the measurements carried out and section 6 contains conclusions and also ideas of future work.

2 Related work

This study is part of a larger project concerning the possibility to use WLANs for IP based mobile telephony[23]. The two main purposes of this study were to establish a suitable test-bed for different handover latency measurements and to use this test-bed to measure *in-LAN* handover latency in IEEE 802.11b WLANs.

Other studies concerning practical measurements of the IEEE 802.11b handover latency include [15] and [24]. My initial aim was therefore to verify their results while establishing a test-bed to be used for more complex handover scenarios. However, our studies were performed differently and the result shows both similarities and differences.

Mishra et al.[15] measured handovers for a mobile STA, which is moved around in a real WLAN network. The handover is measured by observing the 802.11 management traffic via a set of monitoring machines. They observed that the handover latency is significant and that the *probe* phase (referred to as the *search* phase here and in [24]) was the primary contributer to the handover latency (more than 90%). The handover latency was found to depend heavily on the type of WLAN card used, in particular in the STA, but to some extent also in the AP. The best combination among the equipment tested (Lucent STA and Cisco AP) showed an average latency in order of 60 ms. The study also gives a good description of the factors affecting the *search* phase. The main difference between their study and this study is that they do **not** transfer any voice data during the handover and therefore miss some of the relevant handover properties.

Velayos and Karlsson[24] also measure handover latency by observing the messages exchanged when a STA reassociates with a new AP. In their study the STA actually exchanges data traffic with some host on the *wired* side of the AP, however, no details of this data traffic is given except that it had "the characteristics of voice over IP". In their setup they use stationary Linux PCs with PCI WLAN cards as APs rather than commercial APs. To a large extent their study comes to the same result as [15]. The biggest difference between these two is that [24] claims that the movement *detection* is a large contributer to the handover latency. Their study also contains a good analysis of the detection and search phases and gives suggestions of how to reduce them. The significant difference between their study and my study is probably due to the way handover is triggered. In their study the power to the *current* AP is switched off, while handover in my study is triggered by *decreasing* the transmission power of the current AP. That is, in their handover scenario the STA **suddenly** looses connectivity with its current AP, while the STA in my case will begin the handover before losing the connectivity via the current AP.

The study described here should be seen as a complement to, rather than a replacement for these other studies.

3 Theoretical background

This section contains background information on the handover scenario considered in this study, i.e., when a STA performs an *in-LAN* handover between two bridging APs attached to an Ethernet bridge (Layer-2 Ethernet switch), as shown in figure 1. Section 3.1 covers the IEEE 802.11 handover process and is primarily based on information from the IEEE 802.11 standard[5]. Section 3.2 concerns the LAN backbone infrastructure, with a focus on the address filtering feature of bridges.

In the discussion of IEEE 802.11 the STA is assumed to have only a single interface as the focus of the discussion is about the 802.11 MAC and its effect on the link layer and not about higher layer protocols. Thus more properly "STA" should really be written as "a given IEEE 802.11 interface to a mobile station", but in keeping with the terminology of the standard we use the term "STA" and will later in the document return to the discussion of mobile stations with multiple interfaces.

3.1 IEEE 802.11 handover procedure

When a mobile STA is operating in *infrastructure* mode it will try to *associate* with an AP in its vicinity. Each AP constitutes a *basic service set* (BSS) and all the traffic to and from the STA will go via the AP, even traffic between two STAs associated with the same AP. To cover a larger area, multiple APs can be connected via a *distribution system* (DS) to form an *extended service set* (ESS). A STA moving out of the coverage area (cell) of one AP could *reassociate* with another AP (within the same ESS) in the new location, thus performing a layer-2 (link layer) handover. APs with overlapping coverage areas are commonly configured to operate on different frequency channels to avoid interference between the cells.

The standard does not specify the design of the DS[5], but a commonly used solution is to connect a set of *bridging APs* via one or several Ethernet bridges as shown in figure 1.

When a STA is moving away from its current AP, the signal quality of the messages received from the AP will decrease. At some (configurable) signal quality threshold, the STA will start to look for a better AP to reassociate with, thus triggering a handover procedure. The standard specifies that a STA can only be associated with one AP at a time[5], so there is a risk that the communication is interrupted while performing the handover. The duration of the period when the STA in unable to exchange data traffic via its old and new AP is often referred to as the handover latency or handover delay. However, the full definition of the handover latency needs to be done with more care. To facilitate this discussion figure 2 shows the messages usually exchanged as part of a handover.

If the mobile STA experiences degraded signal quality in the communication with its AP (e.g., by measuring the quality of the received beacons), it will at some point in time trigger a handover procedure to start. If the handover threshold value is configured so that a handover is triggered before connectivity with the current AP is lost, then the time to detect movement will **not** affect the total handover latency. To find candidate APs to reassociate with the STA will start to *scan* the different radio channels; scanning can either be done

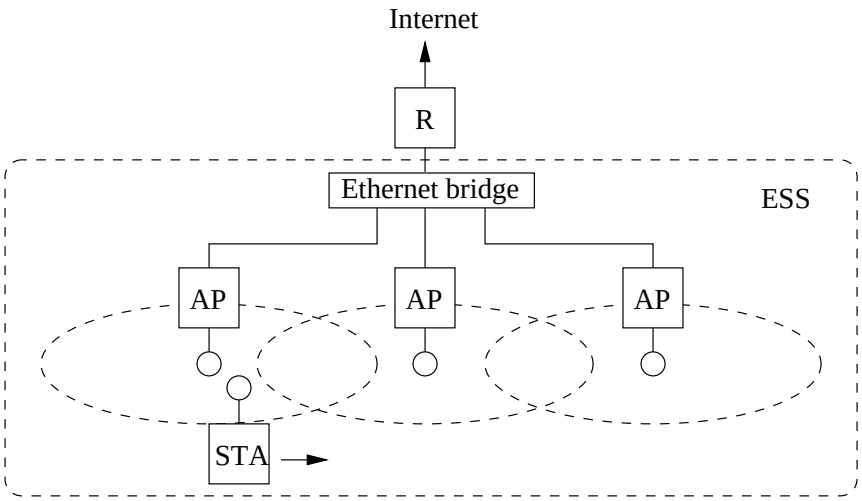

Figure 1: A STA about to perform a handover between APs in the same ESS. The APs are attached to an Ethernet bridge (layer-2 Ethernet switch).

passively (by listening for beacon messages from APs) or actively by sending a *probe request* message on each channel and listening on that channel for *probe responses* from APs, see figure 2 messages 1-4 (the STA will send at least one probe request and receive zero or more responses per channel depending on the number of APs on that channel serving the ESS specified in the probe request). In this study we assume active scanning. This phase of the handover is referred to as as the *search* phase[24].

When the STA has finished scanning for candidate new APs, it will initiate the reassociation procedure with the best candidate AP. This reassociation can be divided into two sub-procedures, the authentication message exchange (messages 5 and 6 in figure 2) and the reassociation message exchange (message 7 and 8 in figure 2). [24] refers to all of this as the *execution* phase of the handover. However, to give a more precise description of the execution phase the following should be noted. There are two different authentication methods: open system authentication which uses a 2-way handshake (as shown in figure 2) and shared key authentication (not considered in this study) which involves 4 messages between the STA and the new AP. The STA is allowed to authenticate with multiple APs, and the STA can do so while keeping its association with its AP (this feature is known as pre-authentication). When subsequently reassociating with the new AP, the STA could also send a *Disassociation* message to its old AP, notifying it that it is leaving. Neither the pre-authentication feature nor the *Disassociation* message were observed in the measurements performed in this study.

If we consider figure 2 and assume that the STA is not able to send or receive any data during the *search* or *execution* phases, then the handover latency would be the sum of the search and execution latencies. We make this assumption because the WLAN hardware we have used has only a single receiver and single a single transmitter; both operating in the same radio channel. Measuring

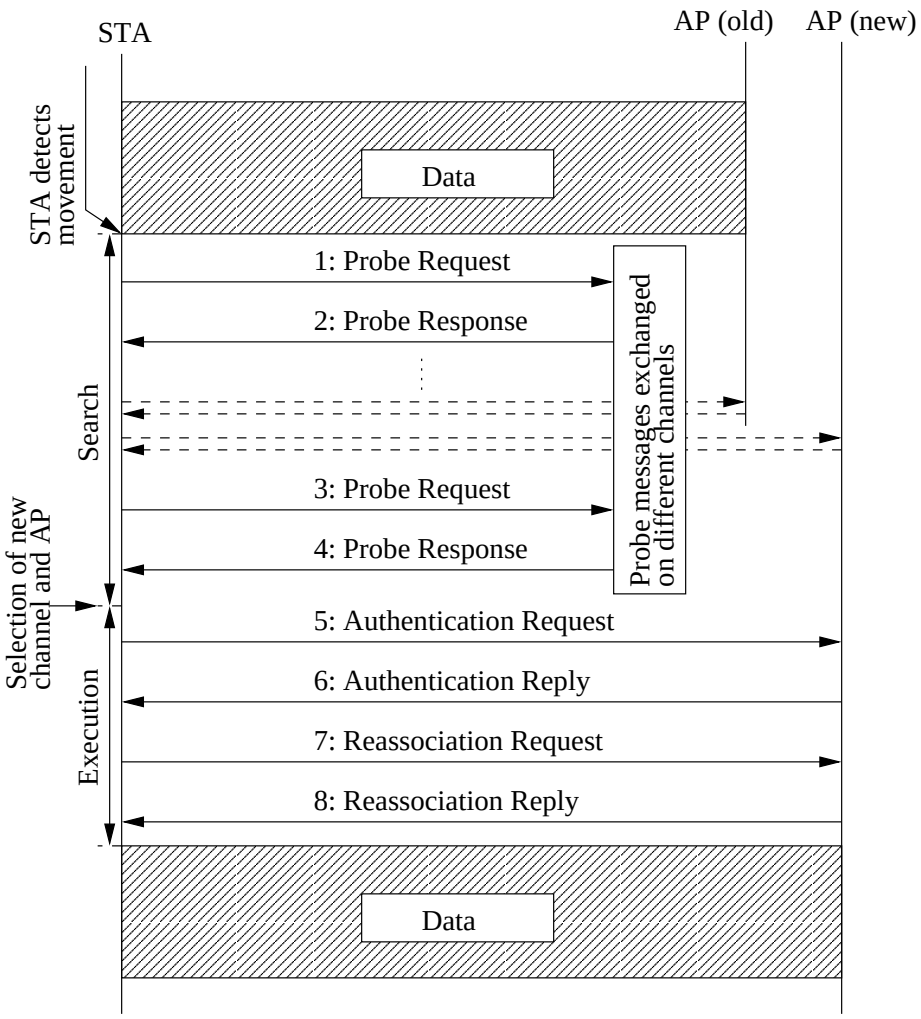

Figure 2: Handover procedure as commonly described.

the handover latency could be done by adding some *hooks* into the WLAN driver at the STA, and having it measure the time from when it decides to trigger a handover until the *Reassociation Reply* is received. Alternatively one could *estimate* the handover latency by monitoring the management and/or data packets sent between the STA and the APs over the wireless links. The latter method is used within this study.

However, when analyzing the results of the measurements in this study it was found that figure 2 did **not** match the actual message exchange seen on the wireless links, and that the STA was able to exchange some data traffic during the period from initiating the search phase until the end of the execution phase. This is shown in figure 3, which describes the handover procedure that we will refer to in this study.

If the STA is continuously transmitting data, these *upstream* packets will

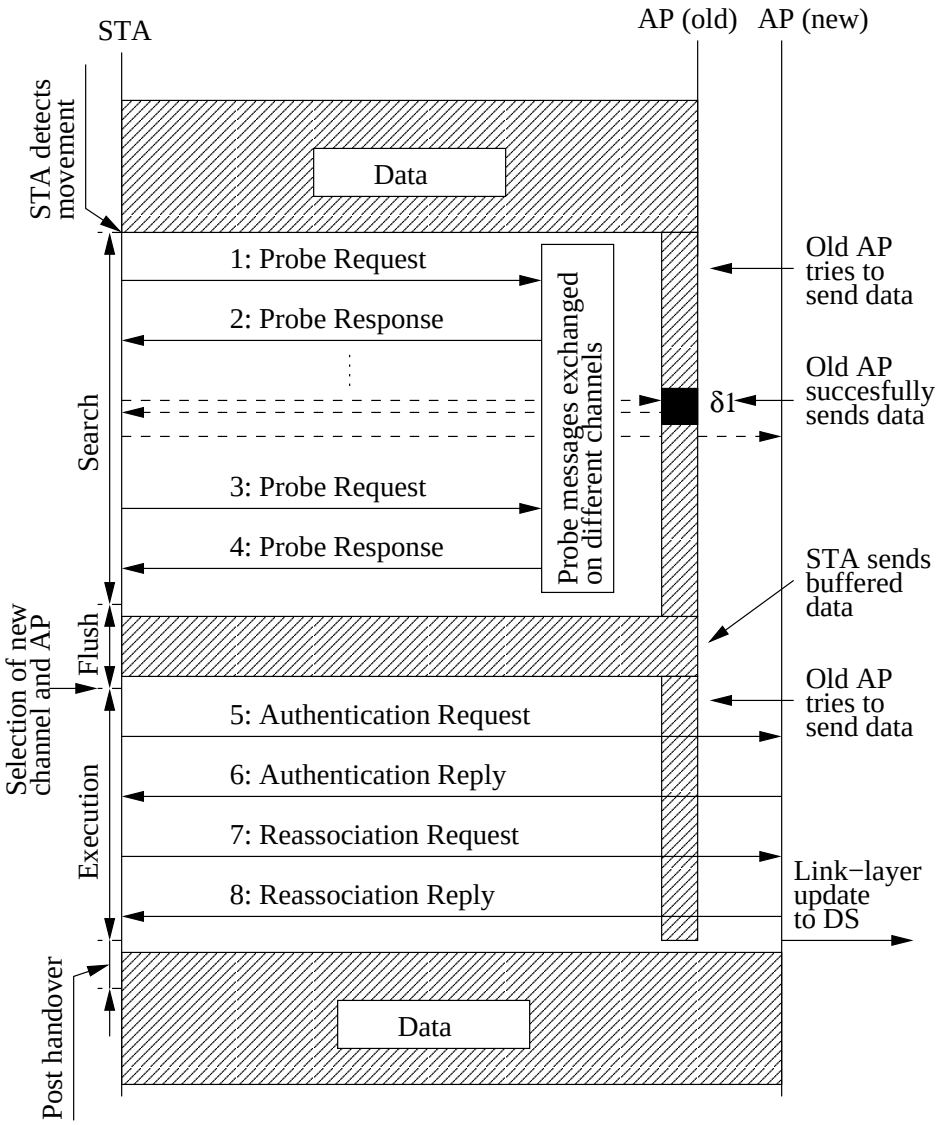

Figure 3: Handover procedure according to this study.

be buffered in the STA during the search and execution phases. Upstream data scheduled to be sent during the search phase may be *flushed* to the old AP between the search and execution phases or sent to the new AP after the execution phase. I have denoted these phases as the *flush* phase and the *post-handover* phase in figure 3. (Figure 3 also contains an additional message, the link-layer update message, that we will return to in section 3.2.)

If we consider the opposite case where the old AP receives (downstream) data for the STA, the AP may buffer these packets if told to do so by the STA (the STA could do this by "telling" the AP that it enters power save mode). After the search phase the STA could notify the AP that is back in *active* mode,

which would enable the AP to *flush* buffered data to the STA before the STA enters the execution phase.

Unless the STA does the *power save mode trick*, the AP will try to transmit data frames to the STA **as they arrive**. What is interesting is that these packets need not necessarily be lost; during the search phase there is a chance that some of these downstream packets may actually get through to the STA. This can happen if the AP sends these frames while the STA is scanning on the channel used by the old AP (marked $\delta 1$ in figure 3).

The exact handover behavior will not only vary depending on the hardware (and resident software or firmware) used, but on whether there are upstream, downstream, or bidirectional data flows during the handover[1]. For example, if there is only downstream data during the handover, there will generally[2] be no *flush* phase and *post-handover* phases. Defining good quality metrics for the handover becomes a bit complex, since it is not only the time intervals of the different phases that are of interest, but also the direction of the data flow and the pattern of packets that may get through during these phases which is of interest. Below we give the metrics we find useful for handover measurements, and their corresponding definition as used in this study.

Definition 1 (Handover interval) *The handover interval ($T_{handover}$) is time interval when packets scheduled for transmission over the WLAN link can be affected (delayed or lost) due to the STA performing a handover.* □

Definition 2 (Handover data) *Those data packets transmitted over the WLAN link or buffered at the STA or (old) AP during the handover interval.* □

Definition 3 (Upstream delay) *The delay imposed on upstream handover data packets.* □

Definition 4 (Lucky packet) *A (downstream) data packet transmitted by the old AP over the WLAN link during the handover interval that manages to reach the STA.* □

Definition 5 (Downstream loss) *The loss pattern for the downstream handover data packets. Since some of these data packets may actually get through, i.e., lucky packets, we have divided the downstream loss into three sub-definitions:*

- *All scheduled packets ($packet_{all-sched}$) which corresponds to the number of all downstream handover data packets.* □
- *Observed handover packets ($packet_{observed}$) which corresponds to the handover data burst length actually perceived by the applications, which can be less than or equal to $packet_{all-sched}$. The difference is that $packet_{observed}$ only counts packets from the first lost to the last lost handover packet (inclusive), while $packet_{all-sched}$ also includes lucky packets at the start or end of the handover data.* □
- *Maximum consecutive handover data packets lost ($packet_{max-cons}$), which corresponds to the longest consecutive loss burst within each handover data.* □

[1] If the handover takes place between talk-spurts, the users may notice the handover at all!
[2] There *can* be a flush phase for downstream traffic if the *power save mode trick* is used.

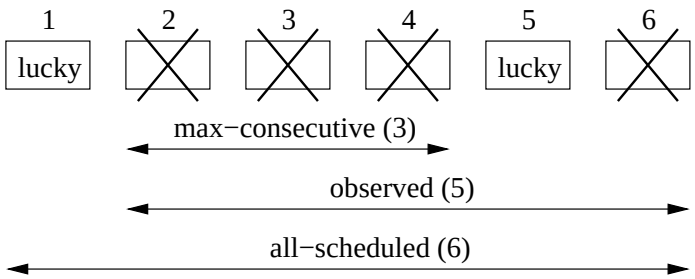

Figure 4: An example set of handover data, and the corresponding values for ($packet_{all-sched}$), $packet_{observed}$ and $packet_{max-cons}$.

Note that $packet_{all-sched}$, $packet_{observed}$ and $packet_{max-cons}$ will all be equal if there are no lucky packets. Furthermore, these numbers will depend on the transmission interval for the (audio) data stream and its correlation with the handover. Figure 4 shows $packet_{all-sched}$, $packet_{observed}$ and $packet_{max-cons}$ for an example set of handover data.

Regarding the *handover interval* ($T_{handover}$) we will here approximate it with the sum of the search, flush and execution phases, although in some cases (upstream) packets, generated after the execution phase, are delayed due the transmission of buffered handover data. Furthermore, viewing the search, flush, and execution phases as purely additive components of the handover interval need not be true, for example, if terminals have multiple WLAN interfaces or if the terminal had a single WLAN interface with an additional receiver (see section 6).

3.2 Design of AP backbone

In this study we consider handover between bridging APs attached to the same LAN and serving the same ESS (as was shown in figure 1). In this section we will discuss the use of bridges (layer-2 switches) in the APs and the rest of the distribution system. A major issue is the address learning feature that is used by the bridges to send unicast data only on the port (currently) heading towards the destination.

It should be noted that although we have limited the scope of this report to the situation when a STA roams within an ESS, the case when the STA performs handover between APs connected to different LANs is also of interest. Such a *cross-LAN* handover will require the STA to change IP address. Since it is often undesirable to use the same ESS identifier (ESSID) on different LANs, a cross-LAN handover may also need additional functionality in order to associate with APs announcing different ESSIDs, but only actually associating with APs/ESSIDs belonging to operators that are willing to serve the user.

As mentioned before, APs are commonly connected to Ethernet bridges and serve the same ESS. For access networks with only a few APs it would also be possible to accomplish this by using Ethernet hubs. Traditional hubs broadcast an incoming packet on all other ports, which can become a problem if the access network grows in terms of APs, number of users per AP, and traffic per user (these changes could lead to the links within the AP backbone becoming

a bottleneck). Newer hubs are a little smarter, since they able to learn on which port to forward packets to reach a certain MAC address. However, these smart hubs are commonly unable to handle multiple data streams going to/from different pairs of ports (since the links attached to such a hub still constitute a single collision domain). If we instead use bridges to connect the APs, the bridges will split the different LAN segments into different collision domains. Just like some of the smarter hubs, Ethernet bridges have the ability to learn where different MAC addresses reside, so that unicast packets only need to be forwarded onto the port heading towards the destination (host). In summary one could say that the bridges learn based on the source MAC address and forward frames based on the destination MAC address.

The bridge will store information about which MAC addresses reside on which ports in a *learning cache* (also known as a *station cache*[19]). This address learning feature will only work if the nodes are actively transmitting traffic or if entries are manually placed in the table (for instance via SNMP). Traffic to unknown destinations (i.e., when the link layer destination address is unknown to the bridge) will be broadcasted on all ports of the bridge, except for the incoming port, thus packets to inactive destinations will be flooded all over the LAN[3].

Learning cache entries will eventually be aged out and removed if the host has been inactive more than a configurable timeout (the default recommended value is 5 minutes[6]). This feature is important, since it enables traffic to reach a station that has moved within the LAN, even if it is not actively transmitting any data. For hosts in wired LAN segments this works fine, since these hosts are not moved very often, and when they are moved the user is likely to be aware of it and can accept some delay before communication resumes. However, a wireless LAN user in the middle of a phone conversation would not tolerate an interruption in the order of minutes or even seconds. To reduce the handover interval for passive STAs (a STA who is not actively sending data at the time of the handover) [5] states that the AP should inform the DS upon a successful association of a STA. The way to inform the DS is **not** standardized yet, but [8] suggests that this should be done by having the new AP send a *link-layer update* message to the link-layer broadcast address on behalf of the STA, i.e., with the STA's MAC address as source address. The message would be flooded throughout the LAN (along the spanning tree) and all bridges would see the source MAC address of the STA and update their learning cache accordingly. Downstream data will immediately be redirected to the new AP. Although not part of the IEEE 802.11 standard[5], we have assumed that the new AP sends a link-layer update message to the DS upon a successful handover, as shown in figure 3.

[3]Bridging APs are a bit special in this way. If the destination address does not match any of the STAs currently associated with the AP it will not forward the packets on the WLAN link.

4 The test-bed

In the test-bed a single mobile station (STA) was able to perform handovers between two access points (AP1 and AP2) attached to the same LAN via an *smart* Ethernet hub[4], see figure 5. To observe the handover process a monitoring machine (Sniff) with two WLAN network cards was used. Sniff was also used to exchange *voice-like* data traffic via the wired LAN with the mobile STA during the tests.

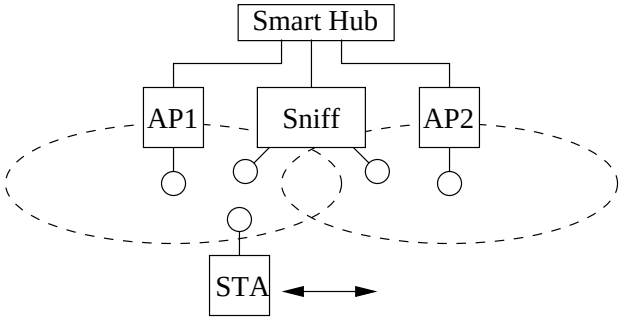

Figure 5: Basic test-bed layout

The two APs and the monitoring machine (Sniff) were Compaq EVOD 300v stationary PCs (Intel Celeron 1100 MHz, 128 MB RAM) running Redhat Linux 8.0. The APs operated on channels 5 and 13 respectively and were equipped with one D-Link 520 PCI WLAN (firmware version 1.3.4) card each, and they used the HostAP (see section 4.2) driver in *Manager mode* in order to act as access points. The APs were configured to send *beacons* at 100 ms interval. Sniff was equipped with two such WLAN cards (and the HostAP driver operated in *Monitor mode*) to enable monitoring of the traffic on both channels 5 and 13.

The mobile STA was a Dell Latitude CPx laptop (Intel Pentium III, 500 MHz, 128 MB RAM) running Redhat 7.3 (linux 2.4.18 kernel) and using either a Lucent Orinoco Silver or D-Link 650 WLAN PCMCIA card (the Lucent card had firmware version 7.28[5] and used the *orinoco_cs* driver (v0.9b) included in the 2.4.18 linux kernel; the D-Link card had firmware version 1.3.5 and used an older release of the HostAP driver (May 19 2002) running in *Managed mode*).

4.1 Linux bridging code

To use Linux PCs as bridges the Linux bridging software developed by Lennert Buytenhek was used[12]. The bridging software consists of two parts, a kernel module (included in the regular Linux kernel distribution) which does the actual bridging tasks and a management application tool (brctl) that can be used to configure the bridge and monitor its status. In our setup the APs had only two interfaces configured for bridging, a WLAN interface (wlan0) and an Ethernet interface (eth0).

[4]This *switching Ethernet hub* (Netgear Fast Ethernet Switch (FS108)) was able to learn on which ports different MAC addresses reside as described in section 3.2.

[5]The Lucent card was reported to have firmware version 7.28 by the orinoco_cs driver and 7.52 by the Orinoco Client Manager (v.2.00) Windows software.

An example of how to configure a bridge interface (br0) with two associated (physical) interfaces (eth0 and wlan0) for AP1 is shown below.

```
ap1#>brctl addbr br0
ap1#>brctl addbif br0 wlan0
ap1#>brctl addbif br0 eth0
ap1#>ifconfig eth0 192.168.4.1 up
ap1#>ifconfig wlan0 0.0.0.0 up
ap1#>ifconfig br0 0.0.0.0 up
```

In this example eth0 was assigned an IP address for the purpose of remote login and for the inter-access point protocol (IAPP)[8] support included in the HostAP implementation. For more information about possible bridge configurations, see the `brctl(8)` manual page or the README file included in the Linux *bridge-utils* package[12].

4.2 HostAP

The Intersil Prism (2/2.5/3) WLAN chip-sets residing in WLAN cards from various vendors have what is called a *HostAP* mode which enables them to act as access points. Such cards together with the HostAP driver[4] written and maintained by Jouni Malinen makes an excellent environment for WLAN experiments.

4.2.1 Implementation of link layer update frame in HostAP

As specified in the IAPP draft[8] the AP should broadcast a *link-layer update* to the distribution system on behalf of the STA in order to update any stale learning caches. At the time of this study, the HostAP CVS-version[6] contained some rudimentary IAPP support including sending a link layer update frame upon a successful handover. Figure 6(a) gives an overview of the way the link-layer update frame was implemented.

Upon a successful reassociation the HostAP daemon (hostapd) sent a Reassociation Reply message (indicating success) to the STA out on the wlan0 interface and a link layer update frame directly out on the eth0 interface (using a packet socket[7] to get direct access to the link layer). This is shown in figure 6(a). The link layer update frame would reach all bridges in the distribution system (including all other bridging APs), updating their respective learning cache, and downstream data would immediately be directed to the new AP. However, the problem with this approach was that the new AP itself dropped the packets, since it had not updated its **own** learning cache. That is, the bridging code in the new AP had a stale entry in its learning cache specifying that the MAC address of the STA resided on the eth0 side of the AP rather than the wlan0 side. This entry could have been added at the time that the STA reassociated with the old AP (resulting in a link layer update) or if it had recently sent a broadcast ARP message. It should be noted that the IAPP support in the version of HostAP used was in an early development stage and had not been as thoroughly tested and used as other parts of the excellent HostAP software.

[6] HostAP CVS-version as of March 24 2003.
[7] Linux packet socket manual page, (`packet(7)`).

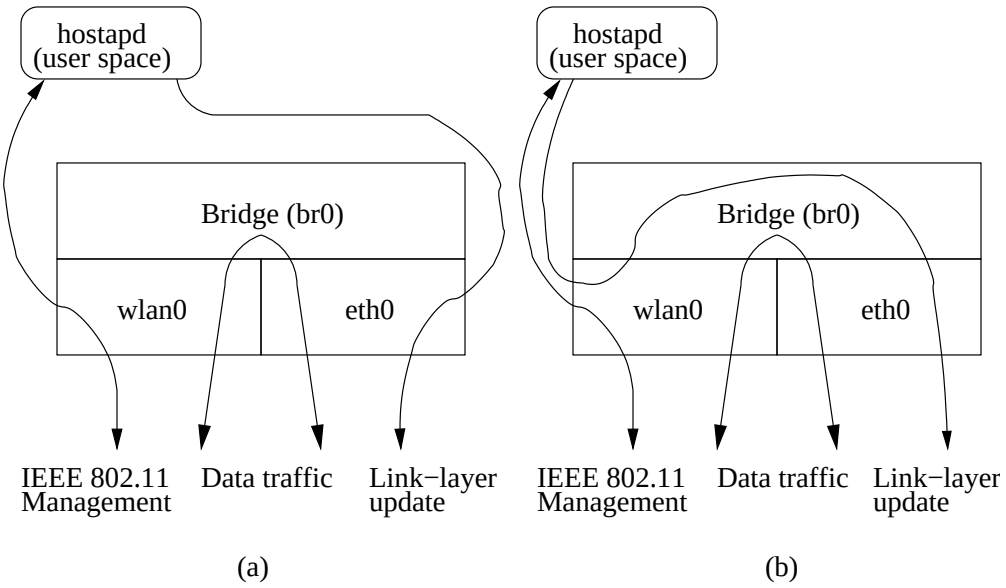

Figure 6: Implementation of link-layer update frame: (a) before contributed patch, (b) after contributed patch

To make the link layer update frame take affect also in the AP's learning cache one could try to integrate the HostAP and the bridging code so that *HostAP* informed the *Linux Bridge* upon a successful reassociation, thus the Linux Bridge could update its learning cache correspondingly. In this study another solution was chosen, namely to put the link layer packet on the input queue of the WLAN interface (wlan0), see figure 6(b). When the user HostAP daemon (hostapd) accepts the reassociation request it will inform the kernel (or more precisely the HostAP WLAN driver) via a new *ioctl* command that it should create a link layer packet on behalf of the STA. The packet is put into an appropriate buffer (sk_buff) and then the WLAN driver calls *netif_rx()* to inform the kernel that a new packet has been received on the WLAN interface. The bridging code in the AP will forward the packet on all other bridge interfaces (here only eth0) and will also update its learning cache (STA resides on wlan0).

A copy of the used HostAP source code version (hostap-cvs-20030324.tgz) as well as the contributed patch (patch-linklayerupdate) can be downloaded from ftp://ftp.ssvl.kth.se/research/mvoip/.

4.3 Handover trigger, emulating movement

A major reason for using PCs and HostAP as access points (rather than commercial products) was that it enabled easy configuration of the AP's transmission power level, by use of the *iwconfig* command from Jean Tourrilhes' wireless tools[8]. In the test-bed this feature was used to emulate user mobility by decreasing the transmission power of the AP that the mobile STA was currently

[8]Wireless LAN resources for Linux , http://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/

associated with. In the test runs the STA was "moved" back and forth between the two APs by repeatedly decreasing/increasing the transmission power of the APs. Alternate ways to trigger handovers are e.g., by physical movement of the mobile STA[15], turning on/off the APs[24] or specific driver support in the STA to force handover[3]. The method of triggering handovers based on power levels (as used in this study) has the desirable properties of both enabling automated testing and still resembling real-world handovers reasonably well.

On the STA, the sensitivity (or AP density) was set to *high*. In this mode of operation the STA can trigger a handover procedure when the signal quality of the current AP drops below some threshold, even if it is still able to communicate via the AP. The result of this was that our test-bed could emulate *soft handover* were the STA would trigger the handover before packets start to be lost, thus movement detection would not affect the handover performance.

4.4 Generating voice-like data traffic

To observe how the handover would affect an ongoing VoIP phone call during the handover the STA and Sniff were exchanging "voice-like" traffic via the APs, see figure 7.

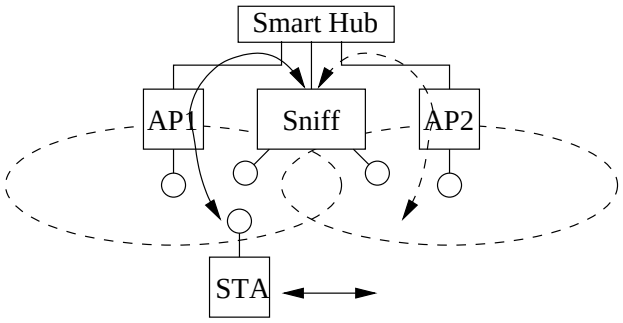

Figure 7: Voice-like traffic flow between STA and Sniff.

To generate "voice-like" traffic the Multi-Generator tool (MGEN)[9] was chosen. MGEN can generate unicast and multicast UDP traffic, both for IPv4 and IPv6. In this setup we used MGEN to generate unicast UDP datagrams over IPv6 with 172 bytes payload every 20 ms in order to emulate a PCM voice stream between IPv6 capable nodes (i.e., 64 kbit/s data stream packed into 160 byte units plus 12 bytes of RTP header). Furthermore, to emulate silence detection most tests only involved one-way communication, either **to** or **from** the mobile STA.

The MGEN sender puts a sequence number and a time stamp in each packet, which is useful when analyzing packet loss and end-to-end delay. Knowing which packets that were successfully received and which were lost is useful, since it complements the tcpdump[10] traces collected by the sniffer (e.g., the sniffer can

[9]MGEN web site, http://mgen.pf.itd.nrl.navy.mil, MGEN version 4.0x6 was used in this study.

[10]See the `tcpdump(1)` manual page.

observe a packet sent from the AP to the STA, but cannot say whether the STA actually received it).

4.5 Clock synchronization and accuracy of measurements

In this study we are interested in performing the following measurements:

1. The *handover interval*, i.e., the time interval during which data packets can be delayed or lost due to an in-progress handover procedure, see also definition 1 on page 8.

2. The timing of the different handover related packets, such as authentication and reassociation messages in order to measure the duration of the different handover phases, see figure 3.

3. The delay imposed on those handover data packets that manage to get through. This is of interest not only for upstream traffic, but also for (lucky) downstream traffic.

For cases 1 and 2 we can use Sniff to passively monitor both WLAN links and the Ethernet backbone, see figure 7. Sniff is running the *tcpdump* program on both its WLAN interfaces and on the Ethernet interface attached to the hub. Since only Sniff is used for this monitoring, there is no need to synchronize Sniff's clock with other machines' clocks in the setup as far as the measurement of cases 1 and 2 are concerned. However, it is important that the frequency of Sniff's system clock is reasonably stable, so that the error of the measured time intervals is within acceptable bounds. It should be noted that the time of an observed data packet is sometimes used to estimate the start or end of some measured time interval. This can give a quite course granularity of the time measurement, since the arrival time of a data packet often (but not always) depends on the transmission interval of the audio application (here 20 ms). This should be kept in mind, so that we do not put unnecessary requirements on the accuracy and precision of the timestamps received by Sniff.

To measure the effect of the handover on the end-to-end delay (case 3) we used the receiver logs generated by the MGEN application. These logs shows the sequence number as well as the sender and receiver timestamps of the successfully received packets. For this measurement the clocks of Sniff and STA must be synchronized.

4.5.1 Reducing clock drift at the monitoring machine (Sniff)

The system clock of the monitoring machine (Sniff) need not show the *wall clock time*, since the measurements for case 1 and 2 are done relative to its own clock. However, when measuring the different time intervals it is important that the clock drift relative to the real time is kept within acceptable bounds. In our test-bed the system clock is controlled by a hardware timer interrupt and for each interrupt the system clock is advanced a certain time quanta. If this quanta is not set properly the system clock will drift compared to real time.

To reduce the clock drift of Sniff's system clock a simple method described in the Linux Clock Mini-Howto[1] was used. Sniff was initially calibrated by letting it synchronize with some NTP[14] servers external to the test-bed. Once

the clock drift of Sniff had been estimated, we could adjust the factor by which the clock was advanced on interrupts using the *adjtimex*[11] program. The external NTP servers are thereafter no longer used (except perhaps for occasional verification that the residual clock drift is within acceptable limits). To adjust the drift of the system clock by use of adjtimex, one would have to get the *adjtimex* parameters *tick* and *frequency*. If one runs *ntpd*[12] as a client on Sniff for a couple of hours (at least) the ntpd estimates for these parameters can be retrieved by running `adjtimex --print`.

The residual clock drift was insignificant for the measurements of this study (upper bound 3.3 us/s, although probably much lower than this if a method with smaller event granularity had been used).

4.5.2 Time synchronization between Sniff and other machines

There were two different reasons to have Sniff synchronized with the other machines in the test-bed. One reason was to simplify the management of the measurements. The APs were configured to alternatively reduce their transmission power temporarily (thereby triggering a handover) every other minute using the Unix *cron*[13] tool. Furthermore, to reduce the size of the monitoring trace files, data packets were only sent during the time interval when a handover would occur. These *management* reasons put only moderate requirements on the time synchronization, and was addressed by letting Sniff act as an NTP server[14] and the other machines as NTP clients. The STA had a dedicated Ethernet connection (not shown in figure 7) to Sniff for this purpose.

A second reason for time synchronization between the different machines was that Sniff and the STA needed synchronized clocks when measuring the effect of the handover on the end-to-end delay, i.e., case 3 above. The initial idea was to rely on NTP and let the STA synchronize with the NTP server on Sniff via a dedicated Ethernet connection as described above. The effect on the *handover data packets* would then be measured by inspecting the end-to-end delay in the MGEN receiver logs of these packets and compare this with the average delay of the *normal* data packets. However, it turned out that the clock drift between Sniff and the STA was non-negligible, probably because the NTP client at STA had not been up long enough. The drift varied for the different test runs, but were in generally below 5 us/s. Comparing the delay of handover packets of each handover with the average delay of packets from the whole test run only gave acceptable results for handovers in the middle of the test run (each test run lasted for about 30 minutes; the average delay for the normal data packets could differ by up to a few milliseconds ($15 \cdot 60 \cdot 5$ [us] = 4.5 [ms]) at the start/end of the test-run compared to overall average.). To address this problem the delay of the handover packets were compared to the average delay of a set of normal data packets just before and after each respective handover.

As a suggestion for future measurements of this kind one could either let the NTP client (at the STA) synchronize with the NTP server (at Sniff) for a

[11] See the `adjtimex(8)` manual page.

[12] The ntpd daemon is part of the NTPv4 release for Unix, VMS and Windows, http://www.eecis.udel.edu/~mills/ntp.htm

[13] Cron is a daemon that executes scheduled commands, see also the `cron(1)` manual page.

[14] The *refclock.htm* page of the ntp-4.1.1 documentation gives more information about how to setup a local NTP server with no external synchronization sources.

longer period of time, or one could skip the NTP client and directly adjust the system clock advancement by adjtimex as described in the previous section.

4.5.3 Random errors

To measure case 1 and 2 (the handover interval and the timing of the handover packets) we use tcpdump on the machine Sniff to capture the packets on the WLAN channels used by AP1 and AP2 as well as on the wired backbone (to capture the link-layer update frames). The tcpdump timestamps reflects the time the kernel first saw the packet and not the time the packet was actually captured by the network interface[15]. If the time it takes for the interface to deliver the packet to the kernel varies, this will introduce a random error in our measurements. Pozzobon has evaluated the usage of laboratory computers running tcpdump as traffic monitors[20], and found "that in general all configurations time stamped the network traffic transmitted to within +/- 27usec on average" for the various computers tested. Although they were not using WLAN interfaces we believe that using tcpdump for measuring the time intervals in case 1 and 2 gives results with adequate precision. Our results are generally presented as an average (and standard deviation) for a set of measured handovers with 1/10 of a millisecond resolution.

The measurement of the additional end-to-end delay (case 3) on upstream packets and the "lucky" downstream packets will not be as exact. The MGEN timestamps used are generated at the application layer (in user space), which leads to larger random errors than if the timestamping is handled in kernel space. Furthermore, when calculating the additional end-to-end delay for the handover packets we compare the delay of these packets with the delay of a set of packets just before and after the handover (as described above), which will also introduce some error. Therefore, when concerning results of case 3 measurements we only report millisecond resolution.

[15] See the `tcpdump(1)` manual page.

5 Measurements

This section presents the results of the measurements performed in this study. We are primarily interested in the case were a STA is either sending or receiving an audio stream – this would model the case when a STA is performing a handover during a voice call when only one party is talking and silence detection is performed by the other party. Section 5.1 covers this "base case" where a unidirectional audio stream is sent either to or from the STA, and these tests are done using two different PC-cards (Lucent or D-Link) at the STA. Section 5.2 covers two other situations: when both parties are sending data (bidirectional) when the handover is performed, and when there is another STA at the new AP competing for media access. These additional tests were only performed using the Lucent card at the STA.

5.1 Base case – sending data to or from a STA during a handover

In this section the measurements for the "base case" are presented, here a STA is sending or receiving a unidirectional audio stream while performing a handover. Sections 5.1.1 and 5.1.2 cover the case when the STA is sending *upstream* data for a Lucent STA and D-Link STA respectively. Sections 5.1.3 and 5.1.4 cover the corresponding cases when the STA is receiving *downstream* data. The results for these base cases are summarized and analyzed in section 5.1.5.

5.1.1 Sending upstream data from Lucent STA

The typical message flow when performing a handover using the Lucent Silver card in the middle of an upstream talk spurt is presented in figure 8 (table 1 presents the corresponding measured values). At some point in time the STA decides to initiate the handover procedure and starts to actively scan for candidate (new) APs on each of the possible[16] channels. Upstream data packets scheduled for transmission during this *search* phase are buffered for later transmission. After having finished scanning for candidate APs the STA transmits a single packet (the first packet buffered) to the old AP before initiating the execution phase.

The *flush* phase is measured as the time from when the *intermediate* data packet ($data_{k+1}$) is observed until the first message in the execution phase is observed. This should give a reasonable estimation of the time added between the search and execution phases for sending the intermediate packet ($data_{k+1}$) to the old AP.

In the execution phase the STA (incorrectly) transmits an initial reassociation message, which is rejected by the new AP. This incorrect behavior of some WLAN cards (or more precisely their driver or firmware) was also reported in [15]. After these *false reassociation messages*, the expected authentication and reassociation message exchanges take place. After the reassociation reply is received by the STA it tries to transmit the next packet in the voice stream, i.e., the second packet ($data_{k+2}$) buffered during the handover interval. Somewhat

[16] A suggestion is to limit the number of channels to search on by guessing the channels where it is most likely to find an appropriate AP based upon past experience, re-use patterns or even measurements in quiet periods[13].

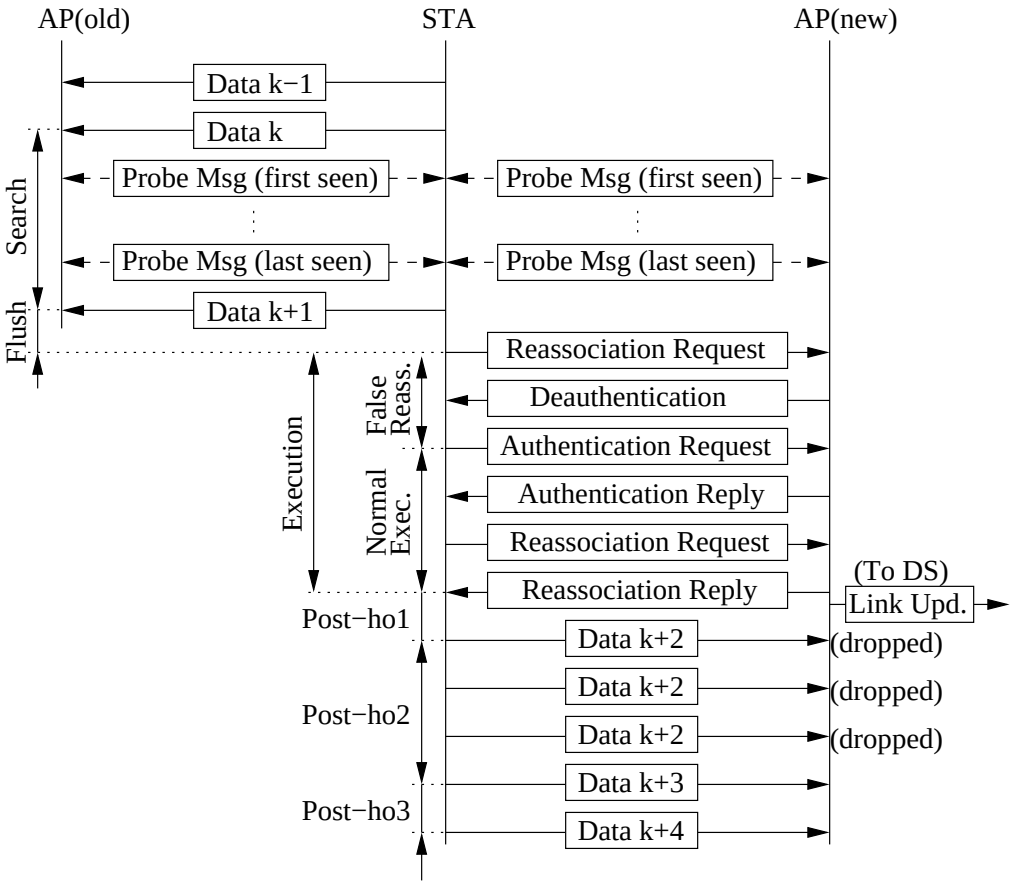

Figure 8: Handover procedure when sending upstream data for Lucent STA.

unexpectedly, this packet is ignored by the new AP, and the reason for this is that although this packet is transmitted on the frequency channel of the new AP, the packet is addressed to the old AP(!), i.e., the packet carries the BSSID of the old AP. The STA commonly retransmits this packet two times before moving to the next packet in the queue of buffered packets ($data_{k+3}$). Each of these attempts simply wastes both time and channel resources[17].

The search time was estimated as the time from the last packet ($data_k$) observed before the handover was initiated until the *flushed* data packet ($data_{k+1}$) was observed, see figure 8. Using the flushed data packet ($data_{k+1}$) to denote the end of the search phase is natural; this packet is buffered in the STA and will be transmitted as soon as possible after the search phase is over. Using the last successful data packet before the handover ($data_k$) as a measure of the start of the search phase will lead to a (non-negligible) overestimation of the search time, since it will depend on the timing between the scheduled transmission

[17] A suggestion is to let the APs promiscuously listen for data; thereby the new AP could have accepted and acknowledged the second data packet ($data_{k+2}$)[13]. This would have removed the packet loss and reduced the delay for the following packets.

	Mean [ms]	σ [ms]
T_{search}	72.5	7.8
T_{flush}	1.8	0.0
$T_{execution}$	6.7	1.6
$T_{false-reassociation}$	3.1	1.5
$T_{normal-execution}$	3.6	0.5
$T_{handover}$	81.0	7.9
$T_{post-ho1}$	0.9	0.2
$T_{post-ho2}$	7.7	1.1
$T_{post-ho3}$	2.1	0.3

Table 1: Measured delay intervals for the different handover phases for Lucent STA (upstream data at 20 ms interval; 29 handovers)

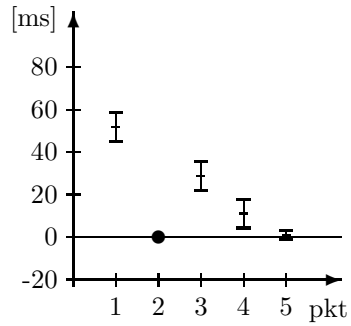

Figure 9: Measured upstream delay (mean$\pm\sigma$) for Lucent STA (upstream data at 20 ms interval; 29 handovers). The 2nd packet is dropped.

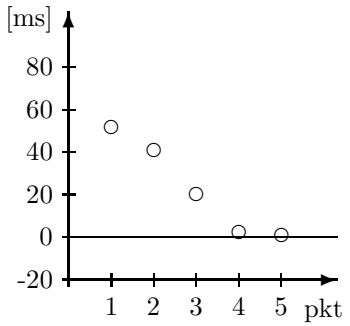

Figure 10: *Predicted* average upstream delay for Lucent STA after some *possible* enhancements as described below (upstream data at 20 ms interval).

of this data packet and the actual initialization of the search phase, which in turn is related to the data transmission interval (here 20 ms). However, using $data_k$ to indicate the start of the search phase was chosen in favor of the time of the first observed probe message, since not all probe messages were captured in this test-bed setup (Sniff only had two WLAN cards and could not listen to all channels simultaneously).

Figure 9 shows the upstream delay for *handover packets* scheduled for transmission during the handover procedure. The first packet is sent via the old AP, while the second is lost for reasons already mentioned. As one can see, only the first four packets are generally affected by the handover; the second packet is dropped while packet 1, 3 and 4 are delayed about 52 ms, 29 ms and 11 ms on average respectively.

5.1.1.1 Possible enhancements To improve the handover performance there are some possible enhancements one could try. First, if the *false reassociation request* message would not have been sent, the execution phase could have been decreased by roughly 3 ms on average.

In addition, if $data_{k+2}$ would have been sent using the BSSID of the new AP, that packet would have reached the AP 0.9 ms after the execution phase finished. Then, one could predict the end-to-end delay added for $data_{k+2}$ if it had been received correctly to be about 41 ms on average, by interpolating the delay of $data_{k+3}$ (average 29 ms, see figure 9), which was transmitted 20 ms later and arrived "post-ho2" (7.7 ms, see table 1) afterwards ($29 + 20 - 7.7 \approx 41$). Furthermore, since no retransmissions of $data_{k+2}$ would take place, $data_{k+3}$ should have arrived about 5.6 ms earlier (7.7-2.1), since the time interval "post-ho2" should become about the same as "post-ho3", see table 1. From this follows that $data_{k+4}$ would also arrive about 5.6 ms faster if the $data_{k+2}$ would have been addressed to the new AP. Figure 10 shows the predicted average delays if the false reassociation request could be avoided and if $data_{k+2}$ would have been sent using the correct address.

To enhance the handover performance even further it would have been preferable if all packets buffered during the search phase could have been sent via the old AP. For the Lucent STA it would have been beneficial if $data_{k+1}$ and $data_{k+2}$ (in addition to $data_k$) were sent via the old AP before entering the execution phase. Then the predicted average delay for these packets (2 and 3) would have been reduced by roughly the execution time (6.7 ms, see table 1) compared to the values in figure 10.

5.1.1.2 Odd cases The test-run consisted of 30 handover attempts. The data for one of these attempts was discarded, since the 3rd packet in the *post handover* phase was not captured by Sniff for this handover.

5.1.2 Sending upstream data from D-Link STA

The D-Link 650 card proves to have very different handover characteristics than the Lucent Silver card. In particular it has a much longer search phase which significantly affects the handover performance. What is interesting is not only that the timing differs, but also the set of messages exchanged is different. Figure 11 shows the messages exchanged when performing a handover when sending upstream traffic for the D-Link STA. When using the D-Link card the STA will send a power save (PS) mode management message to the AP (informing the AP that it will go to sleep) before entering the search phase.

	Mean [ms]	σ [ms]
T_{search}	188.7	31.5
T_{flush}	15.1	0.3
$T_{execution}$	7.2	1.5
$T_{false-reassociation}$	2.4	0.6
$T_{normal-execution}$	4.8	1.5
$T_{handover}$	211.0	31.7
$T_{post-ho1}$	1.1	0.2
$T_{post-ho2}$	1.0	0.3

Table 2: Measured delay intervals for the different handover phases for D-Link STA (upstream data at 20 ms interval; 29 handovers)

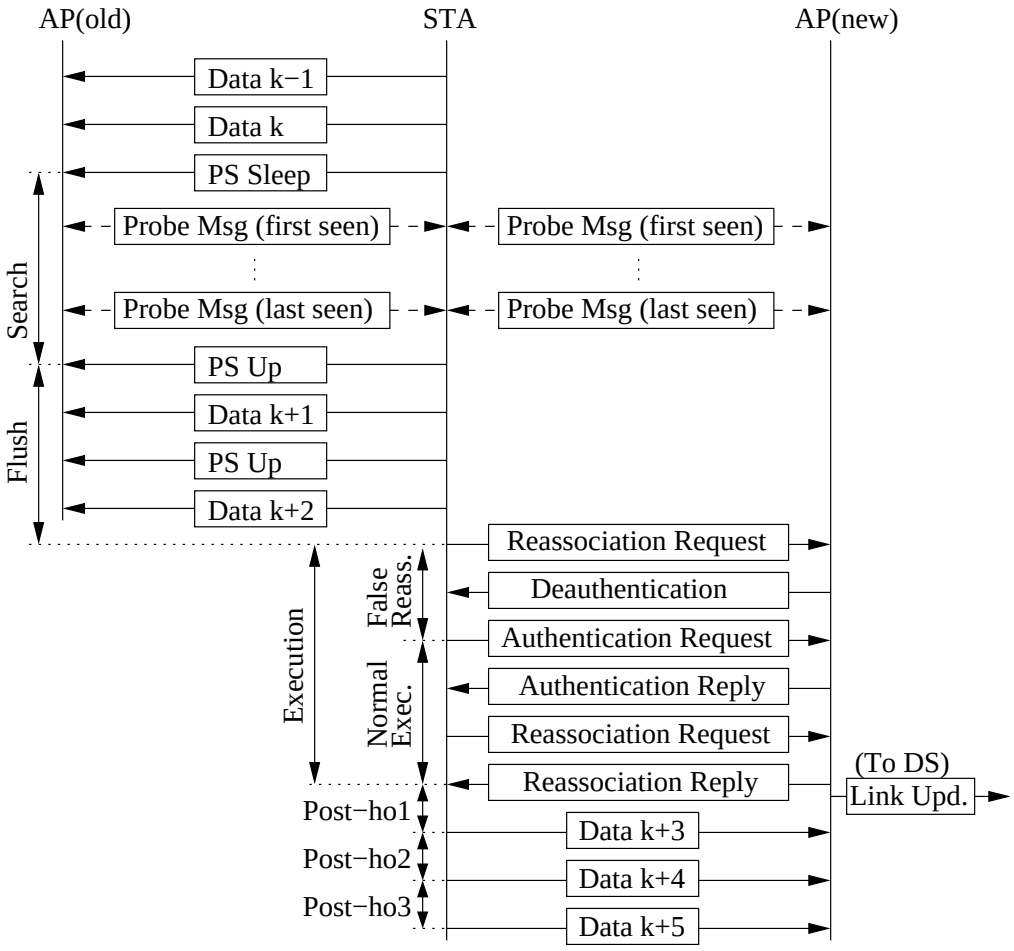

Figure 11: Handover procedure when sending upstream data for D-Link STA.

This has no major effect when the STA only transmits data traffic, but as we will see in the *downstream* case (section 5.1.4) this will make the AP buffer any packet destined for the STA. We use the time the "PS-sleep" message is observed to denote the start of the *search* phase (an alternative would have been to use the time of the last successfully transmitted packet to the old AP ($data_k$) as was done for the Lucent Silver card, but we believe that the PS message gives a better estimation of the search start time).

Similarly, after having finished scanning for new APs, the STA transmits a new PS message informing the old AP that it enters active mode again, and we use the time for this message to denote the end of the *search phase* and start of the *flush phase*. The D-Link card "flushes" two of its buffered packets to the old AP before it enters the execution phase. Just as for the Lucent case the end of the flush phase is measured as the time of the first message of the execution phase. We also note in figure 11 that the STA during the *flush phase* sends a third PS message. Just as did the second PS message, this message indicates

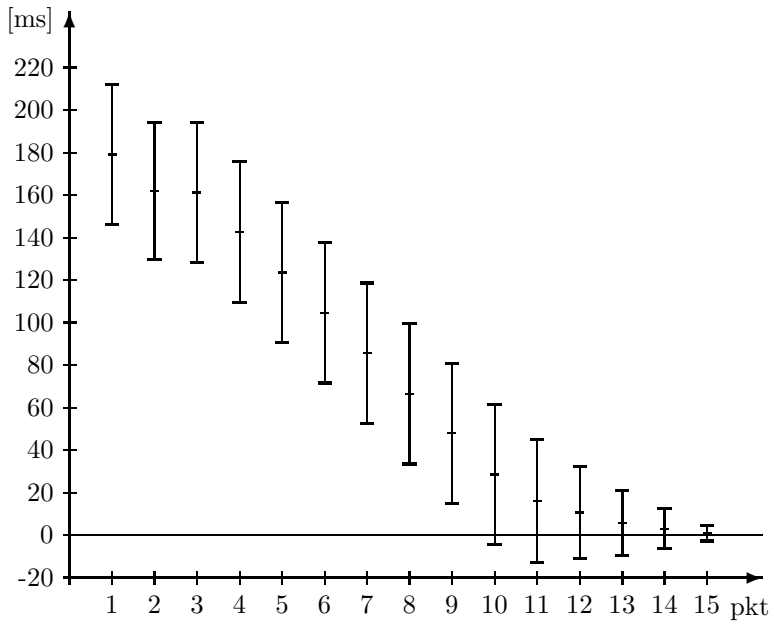

Figure 12: Measured upstream delay (mean $\pm$ σ) for D-Link STA (upstream data at 20 ms interval; 29 handovers).

that the STA enters active state.

The messages exchanged in the execution phase are the same in the D-Link and the Lucent cases; the D-Link card also (incorrectly) sends a reassociation message before authenticating, and the AP rejects this reassociation attempt. Table 2 presents the measured values for the different phases of the handover interval. The search phase is much longer than for the Lucent STA, and it is also noteworthy that its standard deviation is also high. The flush phase accounts for 15 ms of the handover interval, but what is not shown in table 2 is that the majority of this time (about 12 ms) occurs between the transmission of $data_{k+2}$ and the false reassociation request.

After the execution phase finishes, the STA transmits the buffered packets to the new AP. All packets are received correctly, although the delay introduced by the handover is significant and affects many packets. Figure 12 shows the average delay ($\pm$ the standard deviation) introduced by the handover procedure when sending upstream data at 20 ms interval.

5.1.2.1 Possible enhancements The handover interval for the D-Link card is significant. Although no packet is lost, voice packets during 180 ms (9 packets at 20 ms interval) perceive an upstream delay of around 50 ms or more on average. From figure 12 we also see that the delay of the packets decreases roughly by the transmission interval (19 ms instead of 20 ms); packets 1 and 2 follow a different curve, since they are transmitted in the *flush phase* via the old AP and are therefore not affected by the execution delay (7.2 ms) and not by the part of the flush phase used to send the false reassociation message (12 ms) either. If $data_{k+3}$ and the other buffered packets would have been transmitted via the old

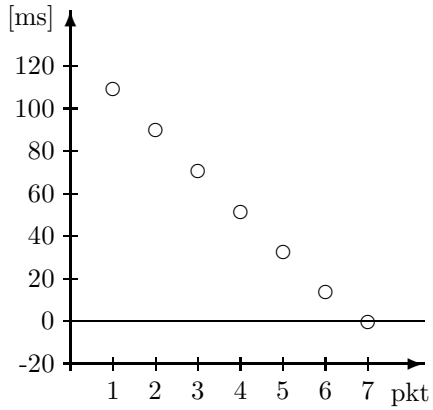

Figure 13: Measured upstream delay for the only *early search phase* observed for the 29 handovers for D-Link STA (upstream data at 20 ms interval).

AP *in a prolonged flush phase* they would therefore have perceived an average upstream delay about 19 ms ($7.2 + 12$) less than the delay shown in figure 12.

The usage of the PS Mode management messages are interesting, although of little use when sending upstream data packets. It is also a bit unclear why the STA sends two identical PS messages (instead of only one) after having finished scanning for new APs.

5.1.2.2 Odd cases The test-run consisted of 30 handover attempts. The data for one of these attempts was discarded, since the second PS packet (used to denote the end of the search phase and start of the intermediate phase) was not captured by Sniff.

In one of the remaining 29 handovers, the STA initiated the handover procedure twice. The first time the STA only performed a search phase, but never entered the execution phase. The second handover attempt, which occurred about 2 seconds later, succeeded and is included in the analysis presented earlier in this section. The *early search* phase observed here lasted for about 124 ms. Figure 13 shows the affect on the end-to-end delay for the handover data. Only six packets were affected (the first packet in the handover data was delayed 109 ms and the sixth packet was delayed 14 ms).

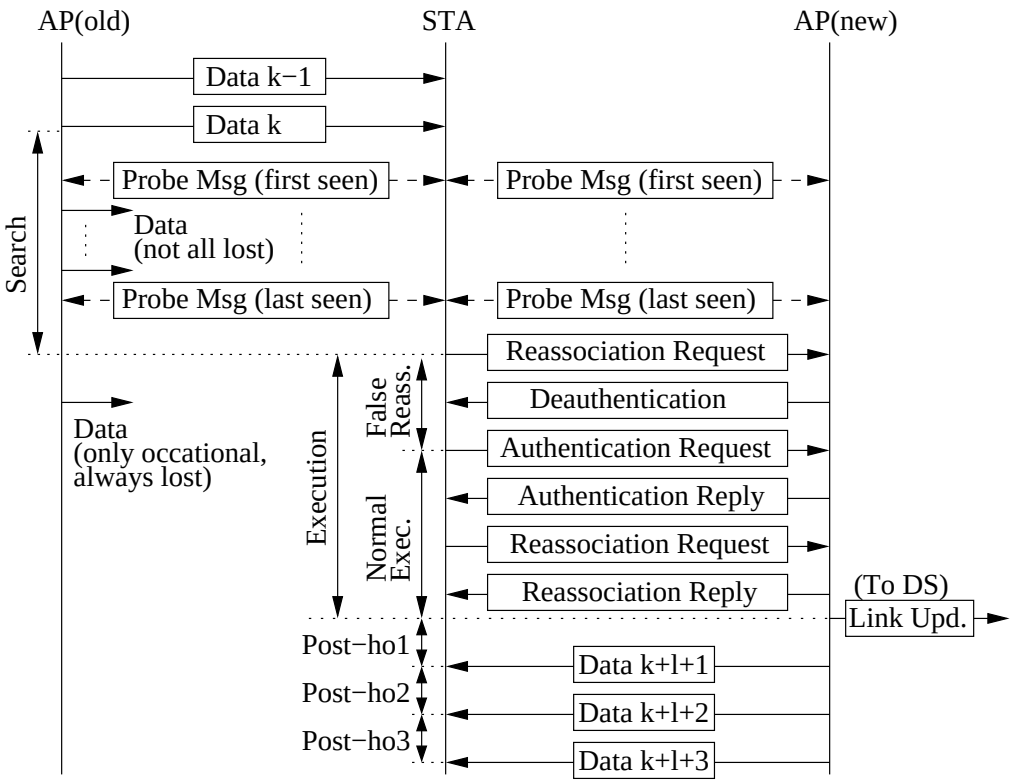

Figure 14: Handover procedure when sending downstream data for Lucent STA.

5.1.3 Downstream traffic to Lucent STA

When the Lucent STA performs a handover while receiving traffic, the old AP continues to forward arriving packets to the STA during the search and execution phases, see figure 14 (table 3 shows the corresponding measured delays). We have not considered any *flush phase* in this case, since the STA is not transmitting any packets itself (unlike in the upstream case) and it is therefore both difficult and of little use to measure a flush phase. Another difference is that the end of the execution phase is defined as the time when the *link-layer update* frame is observed, since it is this message (rather than the reassociation reply message) that redirects downstream packets to the new AP. The time for the first packet to arrive via the new AP depends heavily on the transmission interval (it should be about half the transmission interval on average). Subsequent data packets should come at an interval equal to the transmission interval. Table 3 includes the measured time intervals for the first three packets arriving after the handover finishes for a transmission interval of 20 ms.

Although most of the packets of downstream *handover data* are lost (rather than delayed as for the upstream case), these tests show that for almost every handover (26 out of 30 handovers) one or more of the packets forwarded by the AP during the *search phase* actually reached the STA. The reason for this is

	Mean [ms]	σ [ms]
T_{search}	82.3	13.2
T_{flush}	-	-
$T_{execution}$	7.2	1.9
$T_{false-reassociation}$	2.6	1.5
$T_{normal-execution}$	4.6	0.9
$T_{handover}$	89.5	13.6
$T_{post-ho1}$	7.9	5.8
$T_{post-ho2}$	19.9	1.1
$T_{post-ho3}$	19.6	3.8

Table 3: Measured delay intervals for the different handover phases for Lucent STA (downstream data at 20 ms interval; 30 handovers)

probably that the STA while searching for APs on different channels will also tune into the channel on which its old AP resides, and may therefore receive packets sent to it. Packets forwarded during the execution phase were always lost[18].

Figure 15 shows the distribution of the number of packets (l) forwarded by the old AP during the handover. As mentioned above, usually one (or more) of these packets actually reached the STA. These "lucky" packets could either be in the middle of the handover data (resulting in a "hole" in the loss burst), or at the start or end. "Lucky" packets at the start[19] or end of the handover data are indistinguishable from ordinary packets (perhaps except for some (minor) additional delay – "lucky" packets were delayed 3 ms more on average, since some of them had to be retransmitted). Figure 16 therefore shows the *perceived* handover data ($packet_{observed}$), i.e., the lucky packets at the start or end of the handover data are here ignored. Figure 17 shows the distribution of the maximum number consecutive packets lost ($packet_{max-cons}$) in a handover. In this particular test run lost packets in each handover consisted of the $packet_{max-cons}$, and zero or more "single losses". Thus, an application using a CODEC that is able to mask single packet losses would only see the loss distribution of figure 17, and in 4 of the 30 handovers (i.e., those with only single packet loss(es)) in this test run the handover would not be noticeable at all.

5.1.3.1 Odd cases The test-run consisted of 30 handover attempts and all succeeded. For two of the handovers, the STA initiated an *early search* phase about two seconds in advance of the actual handover, and both caused a loss of three consecutive packets (no lucky packets).

[18]The execution phase was relatively short, thus either zero or only one packet is sent during this phase.

[19]Packets that successfully reached the STA just before the first observed Probe message are considered normal rather than "lucky"; however, since we do not capture all probe messages it is possible that some of these packets should also have been treated as having been sent during the handover, i.e., lucky.

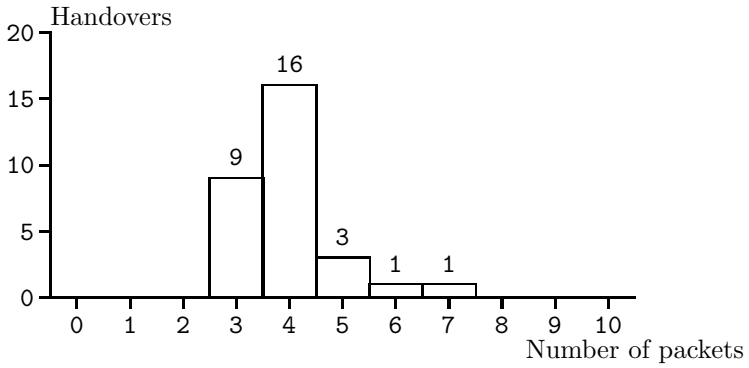

Figure 15: $packet_{all-sched}$: Distribution of packets *sent* by old AP during handover for Lucent STA (downstream data at 20 ms interval, 30 handovers).

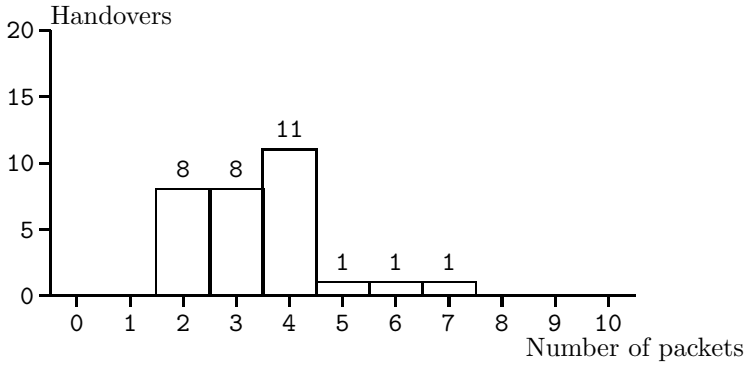

Figure 16: $packet_{observed}$: Distribution of perceived handover data (from first lost to last lost) for Lucent STA (downstream data at 20 ms interval, 30 handovers).

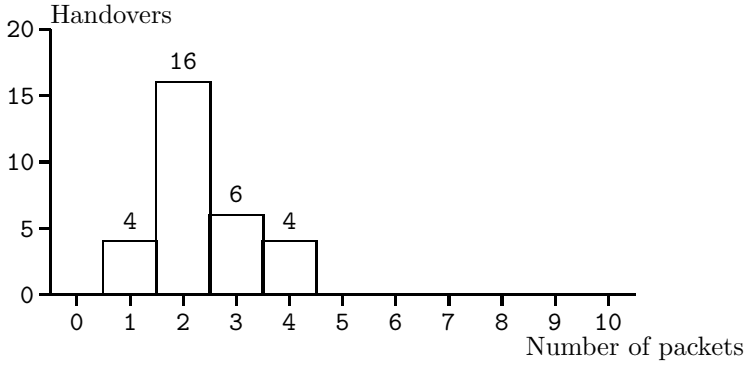

Figure 17: $packet_{max-cons}$: Distribution of longest consecutive loss burst for Lucent STA (downstream data at 20 ms interval, 30 handovers).

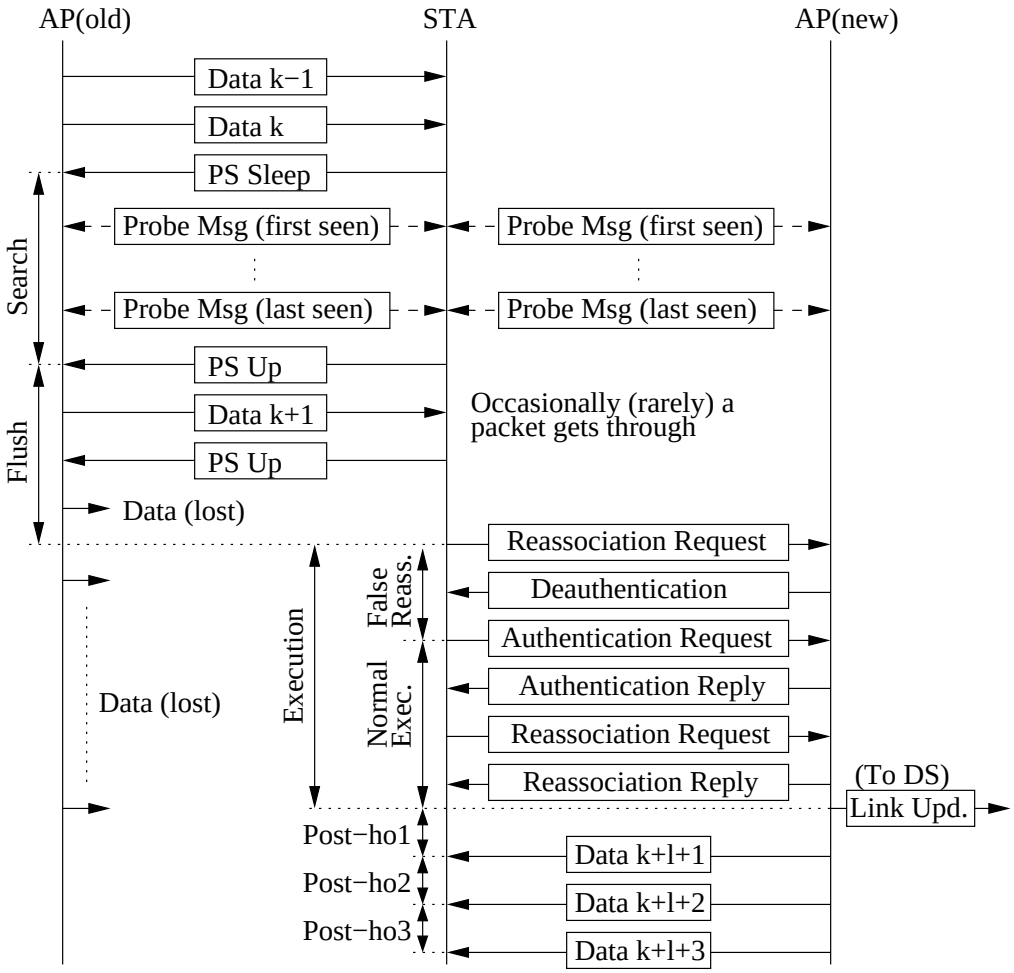

Figure 18: Handover procedure when sending downstream data for D-Link STA.

5.1.4 Downstream traffic to D-Link STA

When the D-Link STA performs a handover while receiving traffic, the old AP refrains from forwarding arriving packets to the STA during the search phase. The reason for this is that the D-Link STA informs the AP that it enters power save mode at the beginning of the search phase, i.e., it performs the PS mode trick described in section 3.1. Figure 18 shows the message exchange occurring for the D-Link STA while receiving downstream data and table 4 shows the corresponding measured delays. The long *search phase* (average 194 ms) affects the handover performance significantly.

For the D-Link case it makes sense to talk about a *flush phase* also for the downstream traffic, since the AP has buffered data that it **could** flush to the STA after the search phase. Therefore we let the second PS message denote the start of the flush phase (and the end of the search phase) just as for the D-Link upstream case (see section 5.1.2). The problem is, however, that the D-

	Mean [ms]	σ [ms]
T_{search}	193.6	28.1
T_{flush}	12.8	2.6
$T_{execution}$	7.4	0.7
$T_{false-reassociation}$	2.3	0.4
$T_{normal-execution}$	5.1	0.6
$T_{handover}$	213.2	29.2
$T_{post-ho1}$	10.4	6.3
$T_{post-ho2}$	20.1	1.1
$T_{post-ho3}$	19.9	1.1

Table 4: Measured delay intervals for the different handover phases for D-Link STA (downstream data at 20 ms interval; 29 handovers).

Link STA does not wait for any buffered packets to arrive from the AP before it enters the execution phase. Only rarely (2 out of 29 handovers) did any of the buffered packets get through to the STA, and that occurred when a packet happened to be sent from the AP between the second and third PS message (see figure 18). In these two cases it was the first buffered packet that got through and the additional delay for these packets were 162 and 215 ms respectively.

Figure 19 shows the distribution of the number of packets scheduled for transmission $packet_{all-sched}$ during the handover. Except for these rare cases when the first buffered packet managed to reach the STA, all packets arriving at the AP destined for this STA were lost. Figure 20 shows the packet loss distribution actually perceived ($packet_{observed}$) by the STA when taking the (rare) "lucky" packets into account (this happens to be the same as the maximum consecutive packet loss $packet_{max-cons}$). But since the delay for the rare lucky packets were so large, the lucky packets could in practice be considered lost as well. Figure 19 would therefore represent the handover loss characteristics experienced by most applications.

5.1.4.1 Possible enhancements The major problem for the D-Link STA is the long time spent on searching for new APs. If the search time can be reduced that would of course be beneficial, but there is at least two other approaches one could try to improve the handover performance for the D-Link STA:

- One could extend the flush phase dynamically by letting the STA dwell on the channel of the old AP to see if the old AP has any buffered data to transmit. The STA could use a (short) timer, which it resets every time it receives a packet from the old AP, and when the timer goes off the STA initiates the execution phase. The added delay would probably be similar to the D-Link upstream delay shown in figure 12. The first few packets will probably be delayed too much, but the packets being buffered later on will probably arrive in time to be useful for the application.
- Instead of searching for APs on all channels at the same time, one could search on one channel at a time. Usually one would only risk losing one packet per channel searched. By alternating between searching and listening to the old AP, an appropriate CODEC may be able to mask the

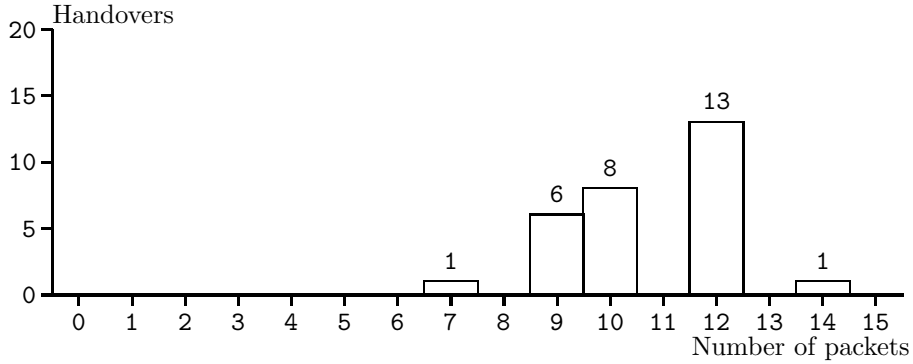

Figure 19: $packet_{all-sched}$: Distribution of packets *scheduled* for transmission by old AP during handover for D-Link STA (downstream data at 20 ms interval, 29 handovers).

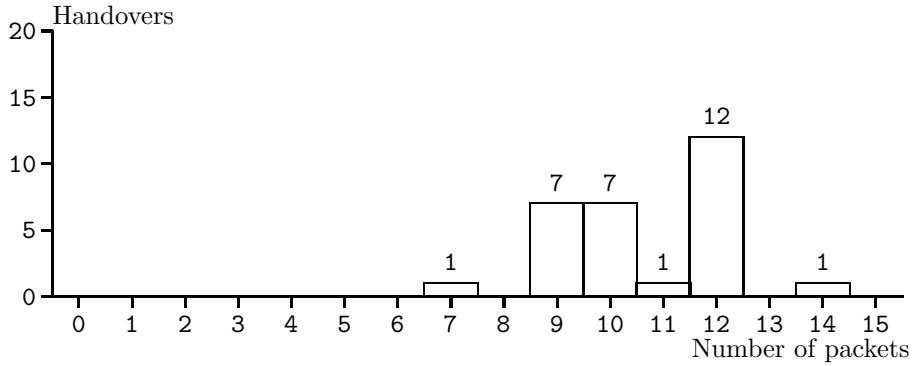

Figure 20: $packet_{observed}$ and $packet_{max-cons}$: Distribution of perceived handover data (from first lost to last lost) for D-Link STA, which in this case equals to the distribution of longest consecutive loss burst (downstream data at 20 ms interval, 29 handovers).

effect of these single losses[13]. Furthermore, if this is done in combination with the PS-mode trick no packets need to be lost at all. A drawback with this approach is that the IEEE 802.11 standard[5] specifies that all channels should be searched at once[20]. Furthermore, it may have some negative affects on the AP performance if they have to deal with lots of PS messages.

5.1.4.2 Odd cases The test-run consisted of 30 handover attempts. The data for one of these attempts was discarded, since the first PS packet (used to denote the start of the search phase) was not captured by Sniff.

	Upstream				Downstream			
	Lucent		D-Link		Lucent		D-Link	
	Mean [ms]	σ [ms]	Mean [ms]	σ [ms]	Mean [ms]	σ [ms]	Mean [ms]	σ [ms]
T_{search}	72.5	7.8	188.7	31.5	82.3	13.2	193.6	28.1
T_{flush}	1.8	0.0	15.1	0.3	-	-	12.8	0.3
$T_{execution}$	6.7	1.6	7.2	1.5	7.2	1.9	7.4	0.7
$T_{handover}$	81.0	7.9	211.0	31.7	89.5	13.6	213.9	29.2
$T_{false-reassoc.}$	3.1	1.5	2.4	0.6	2.6	1.5	2.3	0.4
$T_{post-ho1}$	0.9	0.2	1.1	0.2	7.9	5.8	10.4	6.3
$T_{post-ho2}$	7.7	1.1	1.0	0.3	19.9	1.1	20.1	1.1
$T_{post-ho3}$	2.1	0.3	10.1	0.2	19.6	3.8	19.9	1.1

Table 5: Summary of measured delay intervals for the base cases.

5.1.5 Summary and analysis of base cases

When comparing the behavior of the two types of STAs, we find both similarities and differences. The most important difference is that the length of the search phases differs significantly (see table 5), leading to differences in upstream delays and downstream packet losses for the two STAs. Another general difference is that the D-Link STA informs the AP that it enters power-save mode before initiating the search phase, while the Lucent STA behaves more in-line with our expectations. A surprising similarity is that both STAs (incorrectly) send a reassociation message to the new STA before authenticating.

If we look at the upstream traffic specifically we observed that the D-Link STA sends two packets to the old STA in the flush phase and the Lucent STA one. In addition, one of the upstream packets sent by the Lucent STA is lost while all the D-link packets are successfully transmitted. The reason for the loss is that the first packet transmitted after the handover is addressed to the old AP. The packet is retransmitted a few times before the STA gives up and sends the next (buffered) packet. If we consider the upstream handover interval ($T_{handover}$) for the Lucent STA (average 81.0 ms), which was measured as the time from the *last successfully transmitted packet before the handover* until the AP sends the reassociation reply, one would expect that 4 packets would be affected and that the upstream delay for the last of these 4 packets would be close to zero. However, as shown in figure 9 the upstream delay for the 4th handover packet is higher, and the reason for this is the retransmission of the lost packet.

For the downstream traffic the major finding was that the Lucent STA was usually able to receive one or more of the packets sent by the old AP. If we look at the downstream handover interval ($T_{handover}$) for the Lucent STA (89.5 ms), which was measured as the time from the *last successfully received packet before the handover* until the AP sends the link-layer update message, one would expect that 4 packets would be lost. The perceived handover data ($T_{observed}$), however, has a median value of 3 packets (see figure 16). In the downstream measurements we also saw that the D-link STA did not utilize the possibility of receiving the

[20]However, higher protocols (such as Mobile IP) need **not** observe this MAC limitation.

packets being buffered by the old AP during the search phase.

In table 5 we can see that the upstream and downstream search phases for the Lucent STA differed by about 10 ms. We do not know the reason for this difference, but one should note that the standard deviation for the downstream search phase had a relatively high standard deviation.

5.2 Other parameters

In this section two additional test scenarios are presented in order to complement the measurements done for the base cases. The scope for these tests was limited to only include the Lucent STA, since it showed the best performance for the base case tests.

The test-bed was located in an area with many WLAN APs and users from other WLAN networks, thus our measurements have to some extent been affected by the traffic from overlapping WLAN cells. However, in our base cases there has been no competing traffic from other terminals *associated* with our test-bed APs. Section 5.2.1 therefore presents measurements when a competing STA associated with the new AP has been uploading a large file at the time our STA performs the handover.

Section 5.2.2 covers the case when the STA is both sending and receiving voice data during the handover. This could model the situation when both parties are talking at the same time, or the case when silence detection is not performed.

5.2.1 Competing traffic

This section considers handover performance when there is another STA (associated with the new AP) competing for the media access. Figure 21 shows the modified test-bed used for this scenario. A few seconds before the STA performs the handover to the new AP (AP2 in figure 21) host H2 started to upload a large file to host H1 (to be precise, it was actually host H1 who *down-*

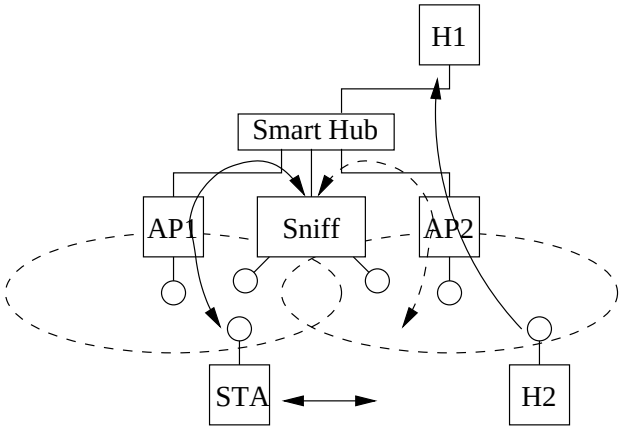

Figure 21: Test-bed layout with competing traffic.

	Upstream		Downstream	
	Mean [ms]	σ [ms]	Mean [ms]	σ [ms]
T_{search}	155.2	49.7	124.1	6.7
T_{flush}	3.6	1.2	-	-
$T_{execution}$	22.5	14.6	18.2	6.7
$T_{false-reassociation}$	4.3	2.3	6.7	2.5
$T_{normal-execution}$	18.3	14.6	11.5	5.9
$T_{handover}$	181.3	46.5	142.3	7.9
$T_{post-ho1}$	2.1	3.4	21.5	6.9
$T_{post-ho2}$	45.2	3.2	15.2	12.2
$T_{post-ho3}$	3.7	2.0	17.1	7.3

Table 6: Measured delay intervals (average and standard deviation) for the different handover phases for a Lucent STA sending either upstream or downstream data, with competing upstream bulk data transfer at the new AP (data transmission interval 20 ms; 5 handovers upstream; 10 handovers downstream).

loaded the file from H2 using the wget[21] application). This means that the STA would have to compete for the media access in the new cell with host H2 sending large TCP blocks and with AP2 sending TCP acknowledgments. Host H2 was a Dell Latitude CPx laptop with the same characteristics as the STA (Intel Pentium III, 500 MHz 128 MB RAM running Redhat 7.3). H2 had a D-Link 650 WLAN PCMCIA card and was configured to have a low sensitivity (or AP density) so that it would not "move" back and forth between the APs. Host H1 was a Fujitsu Lifebook C1020 (Intel Pentium 4, 1400 MHz 256 RAM) running Redhat 7.3.

This test was performed both for upstream traffic from and downstream traffic to the STA. Most of the downstream handovers and many of the upstream handovers observed here showed the same message exchange as when testing the Lucent STA without competing traffic (see figures 8 and 14); what differed here was that the involved delays were larger. We refer to these handovers as "normal" and the delays for the normal handovers for upstream and downstream traffic is presented in table 6 (these results will be discussed further in sections 5.2.1.1 and 5.2.1.2). But what is also of interest is that many of the handovers in the performed test-runs with competing traffic showed a different behavior then without competing traffic. There are three different actions that commonly appear when the STA is faced with competing traffic:

Early search phase An early search phase is a search phase at the old AP (AP1 in figure 21) that is not followed by any execution phase, i.e., it does not lead to a handover to the new AP (AP2). They are often followed by a new search phase leading to a handover. One should note that we already observed *early search phases* in some of the base cases (see sections 5.1.2.2 and 5.1.3.1), but then they were rare; here they occur more often.

Long flush phase In the "normal" handovers a Lucent STA (sending upstream traffic) sends one packet in the flush phase. In these tests there

[21] See the linux `wget(1)` manual page.

are several handovers (especially in the upstream case) when the STA sends/receives multiple packets between the search and the execution phases. We denote this a *long flush phase*, and it differs from the *early search phase* described above in that it actually leads to an execution phase.

Late search phase A late search phase is a search phase that the STA initiates at the new AP, some time after the handover has occurred. This happened for all cases where the handover contained a long flush phase, but was also observed at one other occasion.

Measuring the characteristics and frequency of these three actions are important, since they also affect the delay and packet loss of the audio data. Also, merely the existence of the long flush phases is interesting, since it is in-line with ideas earlier mentioned about decoupling the search and execution phase.

Section 5.2.1.1 discusses the result when a Lucent STA is sending upstream traffic when being faced with competing traffic at the new AP and section 5.2.1.2 covers the corresponding case when the STA is receiving downstream traffic.

5.2.1.1 Upstream traffic The test-run consisted of 15 handover attempts, and the characteristics of these handover attempts are summarized in table 7. Only 12 of the attempts led to a handover, and 7 of these 12 handovers contained a long flush phase. The remaining 5 handovers (cases 1-3 in table 7) contained "normal" handover message exchanges, however, 2 of these handovers also had an early search phase (case 2) and 1 of them had both an early and a late search phase (case 3). The measured delays presented in table 6 only concerns the handover part of these 5 handovers, i.e., early and late search phases are ignored there (but will be discussed further below). The associated upstream delay caused by the handover is presented in figure 22.

As one can see in table 6 the search phase is very long and has a high standard deviation. It turns out that the search phase for three of the handovers (the two from case 1 and one from case 2 in table 7) is in the order of 120 ms while it is about 210 ms for the remaining two "normal" handovers. This high variance is also reflected in the total handover interval ($T_{handover}$) as well as in the affect on the end-to-end delay (table 6). The upstream delay for the packets transmitted via the new AP are also affected by the long execution delay and in particular by the retransmission of the incorrectly addressed data

Case	#	Handover	Early search	Long flush	Late search	Odd
1	2 (13%)	x				
2	2 (13%)	x	x			
3	1 (7%)	x	x		x	
4	7 (47%)	x		x	x	
5	1 (7%)		x			
6	2 (13%)		x			x

Table 7: Characteristics of handover attempts for Lucent STA sending upstream data while facing competition at the new STA. 12 out of 15 attempts led to handover.

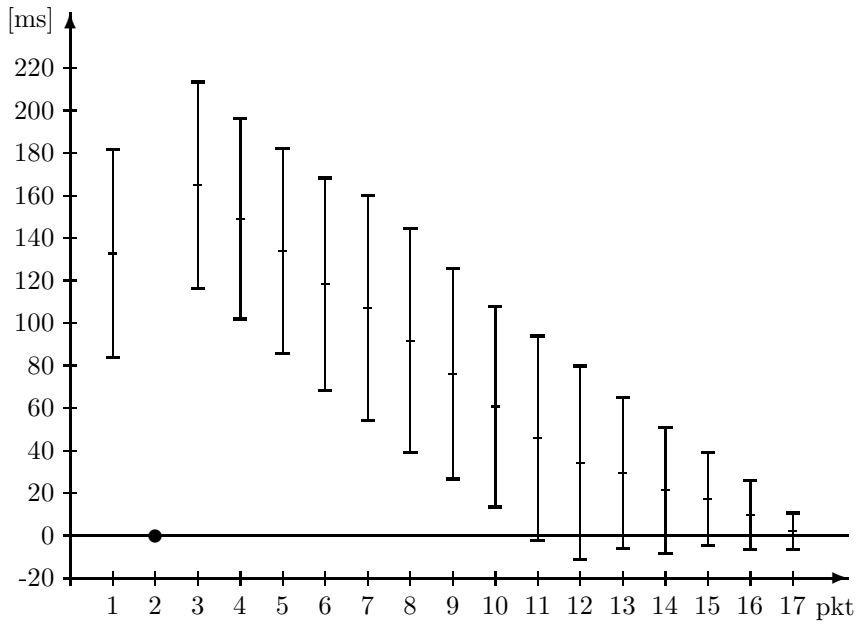

Figure 22: Measured upstream delay (mean$\pm\sigma$) for Lucent STA with competing traffic. (upstream data at 20 ms interval; 5 handovers).

packet ($data_{k+2}$). The long delay involved to retransmit $data_{k+2}$ (about 45 ms on average) not only affects the delay of the following handover data packets; it also means that at least two upstream data packets scheduled for transmission after the handover procedure finished were delayed as well.

Figure 23 shows the upstream delay for all early search phases measured, i.e., both those attempts that did lead a handover (cases 2 and 3 in table 7) and those who did not (cases 5 and 6). The early search phase commonly affected the delay for 6 upstream data packets.

The 7 handovers that had a long flush phase (case 4 in table 7) between the search and the execution phases also had a late search phase at the new AP about 2 seconds after the handover had finished. These handovers will therefore contain three delay spikes, one at the search phase of the handover (figure 24), another one at the (delayed) execution phase (figure 25), and a final one at the late search phase (figure 26)[22]. The delay spikes caused by the early and late search phases are of about the same order, although the early search phase affects one packet less on average (5 instead of 4). The delay caused by the execution phase is less, although not at all negligible. The first packet sent by the Lucent STA after execution is retransmitted and dropped of the same reasons as before, and the delay caused by this seems to be roughly in-line with the upstream "post-ho2" delay of the handovers in table 6. If the first packet

[22]The results in figure 26 is based on the late search phase in case 3 (in table 7), and six of the seven search phases in case 4 – the reason for excluding one of the late search phases in case 4 was that it had a quite different characteristic (the delay increased for the 2nd and 3rd packet).

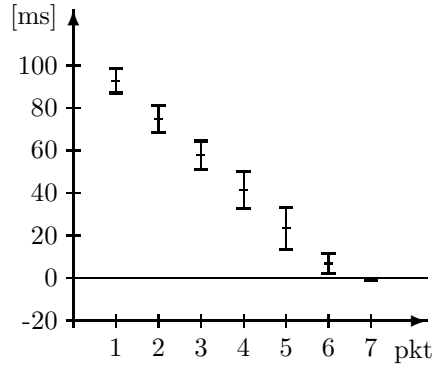

Figure 23: Measured upstream delay (mean $\pm$ σ) during early search phase of handover for Lucent STA with competing traffic. (upstream data at 20 ms interval; all early search phases: 6 handovers).

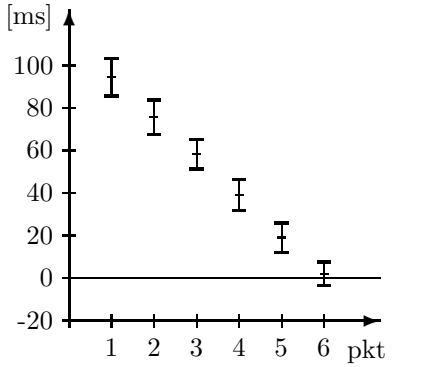

Figure 24: Measured upstream delay (mean $\pm$ σ) during search phase of handover with long flush phase for Lucent STA with competing traffic. (upstream data at 20 ms interval; 7 handovers).

after the execution phase would have been sent with the BSSID of the new AP (AP2), the delays shown in figure 25 would of course have been lower. The length of the flush phase varied from about 12 ms to 900 ms and the number of packets sent in the flush phase varied from 6 to 51.

Odd cases Two of the handover attempts which did not lead to a handover (case 6 of table 7) both showed behavior that differed significantly from the other cases. Both had an early search phase, and the result of them are included in figure 23. In both of these handover attempts the early search phase was later followed by a new, relatively long search phase affecting the delay for 16 and 19 packets respectively. In the former case (affecting 16 packets) the maximum delay was 276 ms, and there was no execution phase. In the latter case (affecting 19 packets) the maximum delay was 338 ms. In the latter case the STA and AP performed the whole execution message exchange, but of some reason the STA continued to send its traffic via the old AP, i.e., the handover never took affect.

Summary – upstream traffic with competition If we summarize the results of these measurements we find that in about half (47%) of the handover attempts in this test-run there was a long flush phase followed by a late search phase. The upstream traffic will in this case have three distinct delay spikes: one in the search phase of the handover (figure 24), a smaller one (including the loss of a packet) in the execution phase of the handover (figure 25) and some seconds later a delay spike in the late search phase (figure 26).

In a third of the cases we have a handover that looks more like the handover without competing traffic. However, the search phase is longer with values around either 120 ms or 210 ms. The other phases are also longer, and in

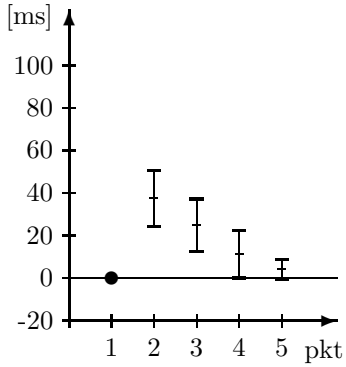

Figure 25: Measured upstream delay (mean$\pm\sigma$) during execution phase of handover with long flush phase for Lucent STA with competing traffic. (upstream data at 20 ms interval; 7 handovers).

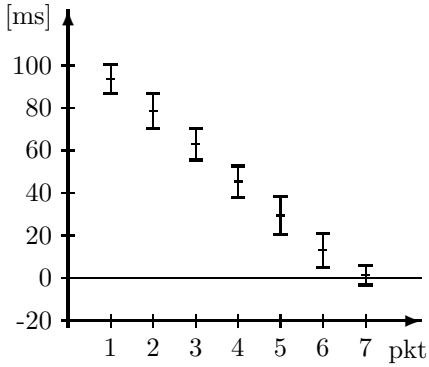

Figure 26: Measured upstream delay (mean $\pm$ σ) during *late search phase* for Lucent STA with competing traffic. (upstream data at 20 ms interval; 7 handovers).

particular the false-reassociation caused a significant upstream delay for the packets sent via the new AP (figure 22). More than half of these handovers were preceded by an early search phase generating another delay spike (figure 23), and one of the handovers had both an early and a late delay spike (figures 23 and 26).

The remaining cases (20%) did not lead to a handover, although some of them contained significant upstream delay spikes for the data.

5.2.1.2 Downstream traffic The test-run when the STA received downstream data when facing competition at the new AP also consisted of 15 handover attempts, and the characteristics of these handover attempts are summarized in table 8. Of the 12 attempts that led to a handover, 5 had "normal" handover message exchanges (case 1 in table 8), i.e., except for the delays involved the handover proceeded as shown in figure 14. Another 5 attempts had

Case	#	Handover	Early search	Long flush	Late search	Odd
1	5 (33%)	x				
2	5 (33%)	x	x			
3	2 (13%)	x		x	x	
4	2 (13%)		x			
5	1 (7%)		x			x

Table 8: Characteristics of handover attempts for Lucent STA sending upstream data while facing competition at the new STA. 12 out of 15 attempts led to handover.

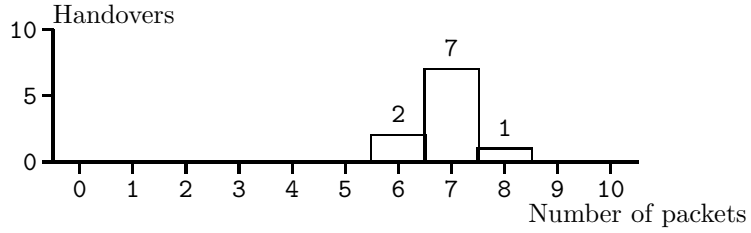

Figure 27: $packet_{all-sched}$: Distribution of packets sent by old AP during handover when Lucent STA is subject to competing traffic (downstream data at 20 ms interval, 10 handovers).

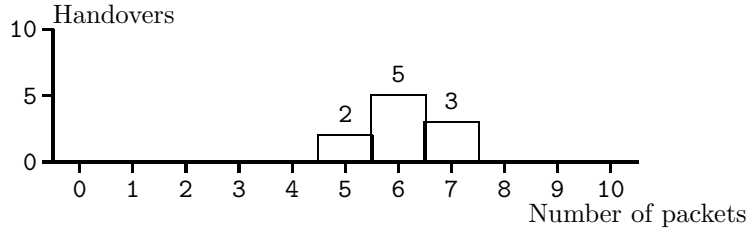

Figure 28: $packet_{observed}$ and $packet_{max-cons}$: Distribution of of perceived handover data (from first lost to last lost) for Lucent STA subject to competing traffic, which in this case equals to the distribution of longest consecutive loss burst (downstream data at 20 ms interval, 10 handovers).

an early search phase followed by a "normal" handover (case 2). The remaining two handovers (case 3) had long flush phases, and there were three handover attempts that did not lead to any handover (cases 4 and 5).

The delay measurements of the handover part of cases 1 and 2 are presented in table 6. Although the delays are significantly higher than for the corresponding measurement without competing traffic, the performance degradation is not as huge as for the upstream case. The average search phase is about 124 ms (almost 50 ms worse than without competing traffic), which is about the same order as "the best three" handovers measured when sending upstream data subject to competition. Another interesting result was that the time before the first packet was received through the new AP ($T_{post-h1} = 21.5$ ms) was rather large compared to half the transmission interval (10 ms), and that the average interval before the second packet arrived ($T_{post-h2} = 15.2$ ms) was less than the transmission interval (20 ms). We do not know the reason for this behavior.

Figure 27 shows the distribution of packets sent by to old AP during the handover ($packet_{all-sched}$) for case 1 and 2. For those handovers where lucky packets managed to get through during the handover (8 out of 10 handovers), there was only a single lucky packet at the beginning of the handover. Thus, the distributions for the perceived packet loss ($packet_{observed}$) and maximum consecutive packet loss ($packet_{max-cons}$) were the same, and they are shown in figure 28.

As presented in table 8 several of the handovers had an *early search phase*, i.e., a search phase at the old AP that was not followed by any execution phase.

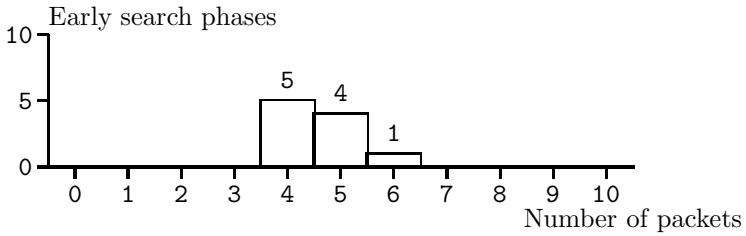

Figure 29: $packet_{all-sched}$: Distribution of packets sent by old AP during early search phase when Lucent STA is subject to competing traffic (downstream data at 20 ms interval, 10 early search phases).

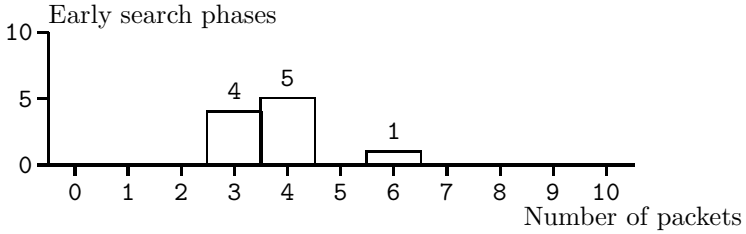

Figure 30: $packet_{observed}$: Distribution of of perceived handover data (from first lost to last lost) for Lucent STA subject to competing traffic (downstream data at 20 ms interval, 10 early search phases).

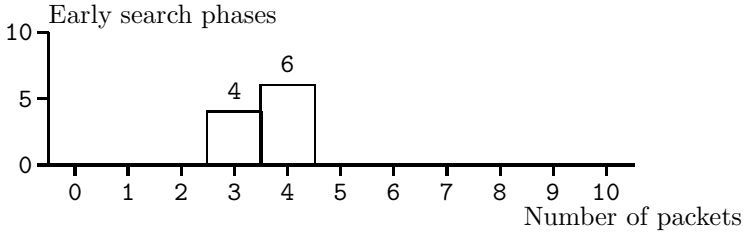

Figure 31: $packet_{max-cons}$: Distribution of longest consecutive loss burst during early search phase when Lucent STA is subject to competing traffic (downstream data at 20 ms interval, 10 early search phases).

In all there were 10 early search phases observed (the two handover attempts in case 4 in table 8 had 2 early search phases each). Figures 29-31 show the distributions of packets sent by the old AP ($packet_{all-sched}$), the perceived handover data ($packet_{observed}$) and the maximum consecutive loss burst ($packet_{max-cons}$) during the early search phase.

To understand the likely handover performance of a handover preceded by an early search phase (case 2 of table 8) we can combine the results of figures 27- 28 and figures 29-31, e.g., the observed packet loss ($packet_{observed}$) will (in median) consist of a loss burst of length 6 followed by a loss burst of length 4 two seconds later.

Two of the handovers in table 8 had a long flush phases (case 4), and about two seconds after the handover finished the STA initiated a new search phase (i.e., a late search phase) at the new AP. During the handover the search phase caused a packet loss of 4 and 5 packets respectively (no lucky packets), and the

execution phase caused a packet loss of 2 packets for both handovers. Losing 2 packets during the execution phase was a bit surprising, and it is not clear why it did not perform better (in one of the two handovers the 2nd packet loss in the execution phase actually occurred just after the execution had finished). In one of the flush phases 15 packets were sent (duration roughly 300 ms) and in the other 52 packets were sent (duration roughly 1 s).

In the late search phase following the handover for both these cases, most of the data sent by the old AP reached the STA, i.e., there were a lot of "lucky" packets. In one of the handovers 5 packets were sent ($packet_{all-sched} = 5$) and only the last packet was lost ($packet_{observed} = packet_{max-cons} = 1$); in the other case 3 packets were sent ($packet_{all-sched} = 3$) and all packets managed to get through ($packet_{observed} = packet_{max-cons} = 0$)!

Odd cases In one of the handover attempts (case 5 in table 8) the handover looked similar to the cases with long flush phases, but after the execution phase the STA remained associated with the old AP. 5 packets were lost in the search phase and 7 packets were lost during or just after the execution phase. 15 packets were sent in the flush phase, which lasted roughly 290 ms.

After this the STA initiated a new search phase (at the old AP) not leading any handover, and the data from this has been included measurements for the early search phase, see figures 29-31.

Summary – downstream traffic with competition In two thirds of the handover attempts in this test-run we observed handovers similar to the handovers without competing traffic, although the involved delays (table 6) were larger as well as the packet loss (figures 27 and 28). The perceived packet loss ($packet_{observed}$) during the handover was typically 6 packets (median). Half of these handovers were preceded by an early search phase causing packet losses as shown in figures 29-31. Thus, during those handovers the STA would first experience a packet loss ($packet_{observed}$) of typically of 4 packets (median) during the early search phase and then 6 packets (median) during the handover.

Two of the handovers (13%) had long search phases and were followed by a late search phase. The measured packet loss did not fully match our expectations: in the search phase of the handover 4-5 packets were lost, in the execution phase 2 packets were lost (more than expected), and in the late search phase 0-1 packets were lost (less than expected).

The remaining three handover attempts (20%) did not lead to a handover, but contain one or two loss bursts in the order of 4-7 packets.

5.2.2 Bidirectional traffic

To test the handover performance when the STA and Sniff were exchanging bidirectional data streams we used the "regular" test-bed setup shown in figure 5 – the only difference being that the MGEN applications at the STA and Sniff were configured to both send and receive a data stream, using different port numbers to separate the data streams in the analysis. Only one test-run was used (28 handovers), and the processing of the output was done separately for the upstream and downstream data in the same way as for the unidirectional tests, see section 5.1.1 and section 5.1.3. The message exchanges presented in figures 8 and 14 turned out to also apply for the upstream and downstream data streams of a bidirectional data stream. Table 9 shows the corresponding delay measurements.

In general one could say that neither the handover performance for the upstream nor downstream data flows are affected in any significant way just because the data is flowing in both directions simultaneously. For the upstream data we see a slight increase (about 1 ms) on the end-to-end delay (see figure 32), which is due to a somewhat longer search phase compared to the unidirectional case (see table 1). The increase of the search phase is, however, low compared to the measured standard deviation. For the downstream data the search phase is, in contrast, a few milliseconds less (4.5 ms on average) than for the corresponding unidirectional test (see table 3), but this difference is also less than the standard deviation of the measurements. For the downstream traffic one could also note that the delay for the first packet to arrive after the handover has finished ($T_{post-ho1}$) is somewhat larger than one could expect (11.9 ms). This may be due to statistical reasons, but it is likely that some of this increase is due to competition for media access from the buffered upstream data packets.

Figures 33-35 show the distributions for $packet_{all-sched}$, $packet_{observed}$ and $packet_{max-cons}$. The distributions are similar to the corresponding distributions in the unidirectional case – there are a few more long handover data sets of size 5 and 6 (see figure 33 and 34) and the maximum consecutive loss bursts are a bit more concentrated to 2 and 3 packets. However, these differences may very well be due limited statistics.

	Upstream		Downstream	
	Mean [ms]	σ [ms]	Mean [ms]	σ [ms]
T_{search}	73.6	7.3	77.7	13.9
T_{flush}	2.0	0.9	-	-
$T_{execution}$	6.5	1.6	7.1	1.6
$T_{handover}$	82.2	7.6	84.8	13.8
$T_{false-assoc}$	2.8	1.5	2.8	1.5
$T_{post-ho1}$	0.9	0.1	11.9	6.5
$T_{post-ho2}$	7.6	1.1	19.6	3.4
$T_{post-ho3}$	2.1	0.5	20.5	2.8

Table 9: Measured delay intervals (average and standard deviation) for the different handover phases for Lucent STA (bidirectional data at 20 ms interval, 28 handovers)

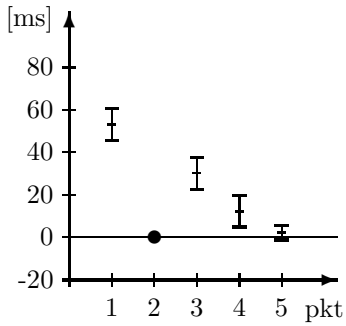

Figure 32: Measured upstream delay (mean $\pm\,\sigma$) for Lucent STA (bidirectional data at 20 ms interval; 28 handovers).

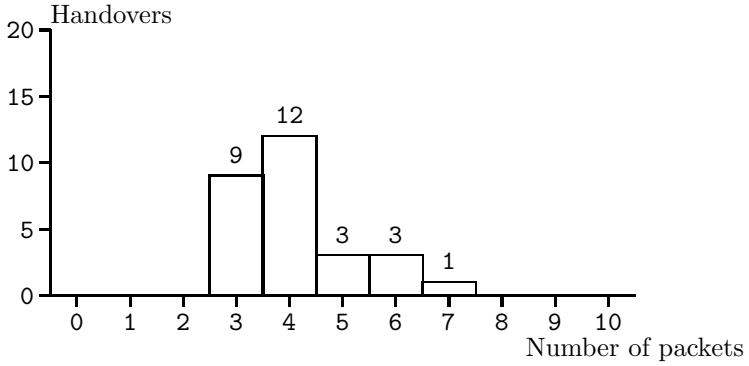

Figure 33: $packet_{all-sched}$: Distribution of downstream packets *sent* by old AP during handover for Lucent STA (bidirectional data at 20 ms interval, 28 handovers).

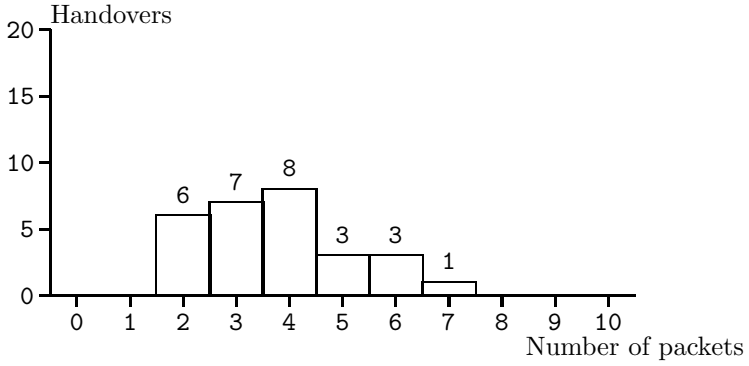

Figure 34: $packet_{observed}$: Distribution of perceived downstream handover data (from first lost to last lost) for Lucent STA (bidirectional data at 20 ms interval, 28 handovers).

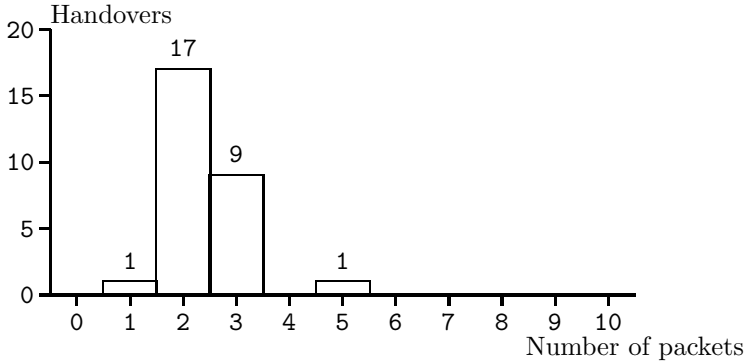

Figure 35: $packet_{max-cons}$: Distribution of longest consecutive downstream loss burst for Lucent STA (bidirectional data at 20 ms interval, 28 handovers).

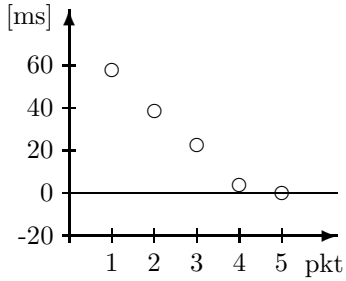

Figure 36: Measured upstream delay the only *early search phase* for the 28 handovers for Lucent STA (bidirectional data at 20 ms interval).

5.2.2.1 Odd cases The analysis of the upstream and downstream handover behavior in this section was done from a single test-run including 30 handover attempts while sending bidirectional data traffic, but only data from 28 handovers are included in the results presented above . For one of these 28 handovers the STA performed an *early search* phase about two seconds in advance of the handover. This lead to a loss of three packets for the downstream data and some additional delay for four packets for the upstream data, see figure 36.

One of the two discarded handover attempts actually did not include any handover attempt at all (the STA happened to be associated with the intended "new" AP already from start). The other discarded handover attempt had a *long flush phase*, and about 2 seconds after the handover the STA initiated a *late search phase* – a behavior quite common when the STA was subject to competing traffic, see section 5.2.1. The handover performance for this particular handover is analyzed further here.

During the search phase of the handover three downstream packets were lost, and as shown in figure 37, three upstream packets were delayed. The intermediate phase lasted for about 150 ms and included the transmission of 11 upstream and 8 downstream data packets before the execution phase was

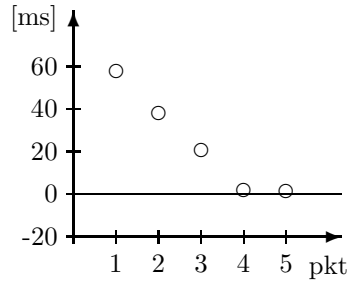

Figure 37: Measured upstream delay for the first 5 (of 11) upstream packets in the only long flush phase for Lucent STA (bidirectional data at 20 ms interval).

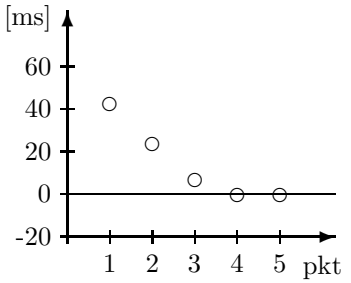

Figure 38: Measured upstream delay in the *late search phase* for the only handover with a long flush phase for Lucent STA (bidirectional data at 20 ms interval).

entered. In this case there were **no** packet losses during the execution phase for either the upstream or downstream data, and the impact on the delay was negligible. It is particularly interesting to see that no upstream data packet was lost in this case, and this should be compared to the "regular" handovers with upstream data for the Lucent card (see figures 9 and 32).

In the *late search* phase occurring at the new AP two seconds after the handover two downstream packets were delayed (the delay of the third packet was negligible), see figure 38.

6 Conclusions and future work

In this study the IEEE 802.11b handover procedure has been studied based on practical measurements. The main contribution of this work is a greater understanding of the handover behavior and performance when a STA is sending or receiving real-time voice traffic while performing the handover. We have shown how the handover affects the data flow in terms of loss and delay, but also that an ongoing data flow affects the timing of the handover mechanism – in particular we have introduced the notion of a *flush* phase between the *search* and execution phases.

We have also presented a method of how to perform IEEE 802.11b handover measurements where all equipment can be stationary and the movement of the STA is emulated by controlling the transmission power of the APs.

As has been shown in earlier studies[15, 24], we found that the search phase has a dominating effect on the handover performance and that the length of the search phase varies depending on the STA in use. What is new here is that one of the tested STAs (Lucent) were actually able to receive some of the packets sent by the AP during the search phase. We also saw a difference between the tested STAs in that the other STA (D-Link) informed the AP that it entered power-save mode before starting to search for new APs. The AP will then buffer packets for the STA, and it would be possible for the STA to let the AP *flush* these packets to the STA between the search and execution phases – this possibility was, however, not utilized by the D-Link STA.

The delay measurements of the execution phase show somewhat higher values than the measurements presented in [24], but the execution phase is still small compared to the search phase. Both STAs tested sent a *reassociation request* to the AP before authenticating, and the delay caused by this *false reassociation* message exchange may account for some of the difference compared to [24].

If we observe the effect of the handover on the upstream traffic, the data flow will experience a delay spike similar to what can happen at congested routers – packets transmitted later will catch up with the ones transmitted earlier and form a packet train. Here it occurs because audio packets scheduled for transmission during the handover interval will be buffered by the STA. It is important to observe that the delay of the packets is not uniform; instead the first packet is delayed the most and the delay of the following packets roughly decrease linearly with the amount of the transmission interval of the CODEC in use.

One interesting observation was that some upstream data packets were sent to the old AP between the search and the execution phases, instead of being buffered until the handover had finished. The D-Link STA transmitted two packets in this intermediate *flush* phase, while the Lucent STA transmitted one packet. When the Lucent STA was subject to competing traffic at the new AP it frequently sent more packets to the old AP in the flush phase.

Another finding concerning the upstream traffic was that the first packet transmitted by the Lucent STA on the channel of the new AP was addressed to the old AP. This not only led to a loss of this packet, but also an additional delay of the following data packets, which were blocked until the STA gave up

retransmitting the incorrectly addressed packet.

In our tests the data packets were generated at 20 ms interval and for the upstream traffic commonly 4 packets were affected by the handover when using the Lucent STA; the 1st was delayed 52 ms on average, the 2nd was dropped, and the 3rd and 4th packet were delayed 29 and 11 ms on average respectively. What affect this would have on the user perceived performance depends on the delay and delay variation introduced by the rest of the network, as well of the play-out buffer algorithm in use and the CODEC's capabilities to conceal packet loss. For the D-link card the handover performance was worse; the first three packets were delayed in the range 160-180 ms on average, and then the delay decreased linearly by about 19 ms per packet until the 11th packet, which was delayed by 16 ms on average.

The major difference between upstream and downstream traffic is that upstream packets are in general *delayed* while downstream packets are in general *lost*. As mentioned above, the Lucent STA (somewhat surprisingly) often managed to receive one (or more) of the packets sent by the old AP during the search phase. Thus, the number of lost downstream packets were often less than the number of packets sent by the old AP. The handover performance for the Lucent STA when receiving downstream traffic is therefore better then what can be expected when only observing the handover control and management messages. For the Lucent STA the median of $packet_{observed}$ (the first lost to the last lost packet inclusive) is 3 packets when generating data at 20 ms interval.

The D-Link STA shows worse performance due to its long search phase. The corresponding median of $packet_{observed}$ for the D-Link STA is 10 packets.

With the Lucent STA we also performed some additional tests to study the handover performance if both parties of the voice conversation were sending audio traffic simultaneously (bidirectional traffic) at the time of the handover, and the case when the STA had to compete for the media access with another STA at the new AP.

The handover performance when the STA was both sending and receiving audio data during the handover did not differ significantly from the earlier tests were the STA was either sending or receiving data. A minor difference was that the first downstream packet being transmitted after the handover had finished was delayed somewhat longer on average, and the reason for this was probably that it had to compete with the buffered upstream packets being transmitted via the new AP.

When the STA was subject to competing traffic from a bulk data transfer by another STA at the new AP, both the handover behavior and performance were affected. The duration of the search and execution phases were longer leading to longer delays or more packets being lost. We also noticed that the STA often performed a search phase some second(s) in advance of the handover. Although this *early search* phase was not part of the actual handover, it caused delay or loss of the audio data. We also observed that many of the handovers contained what we refer to as a *long flush* phase, i.e., the STA sent or received many packets via the old AP between the search and the execution phases of the handover. In our measurements we also saw that some time after the STA had finished the handover after a long flush phase it initialized a new search phase at the new AP. This *late search* phase did not lead to any new handover,

but it affected the delay and loss of the audio data.

There are several ways in which this study could be extended. One way would be to do the same kind of tests for other STAs or APs (comparing different hardware, firmware, drivers and operating systems), or to do it with different configurations in terms of numbers of APs and other STAs present. However, our main interest is instead to extend our study to concern more advanced handover scenarios, were IEEE 802.1X[7] (with adoptions for IEEE 802.11 as being specified by 802.11 Task Group I (TGi)[9]) is used to handle access control and security for roaming WLAN hosts. In our vision[23] of architecture for providing mobile VoIP, Mobile IPv6 (MIPv6)[10] will be used to handle mobility between IP subnets and its effect on the handover performance should also be studied further (both the IETF Seamoby and Mobile IP working groups[22, 16] are working on issues concerning fast handover support for IP hosts). Since mechanisms such as IEEE 802.1X and MIPv6 are likely to extend the handover interval significantly we suggest that the flush phase is used to send all buffered upstream packets via the old AP before the execution phase is entered. If the STA uses the "power-save mode trick" described in this study it could also use the flush phase to receive packets buffered by the old AP, before entering the execution phase.

To avoid additional delay of the upstream traffic during the 802.1X authentication phase another trick could be possible. While waiting for the 802.1X authentication to complete, the STA could send its data packets on the channel of the new AP, but addressed to the old AP (just like the Lucent STA does for one packet). The new AP could then forward these packets (via the wired backbone) to the old AP for further delivery. This requires that the new AP listens promiscuously[13] for packets from all associated STAs, and that the APs are configured to (securely) co-operate in this manner. This and similar ideas we would like to explore more in future work. Both the IEEE 802.11 Task Group I (TGi) and Task Group F (TGf)[8] are working on issues concerning secure and fast handover support with respect to usage of 802.1X in 802.11 environments.

To understand how the user perceived performance is affected by the delay or loss patterns presented in this study one needs to study how the play-out buffer algorithm used reacts on these patterns, if forward error correction (FEC) mechanisms can be utilized to recover some of the packet loss, and also investigate the CODEC's ability to perform loss concealment[18]. Furthermore, if it is possible to control the timing of the handover start depending on the status of the audio stream, the performance degradation can be reduced even further. E.g., if the handover could be triggered just after the transmission of an upstream data packet (or just after an upstream talkspurt) the buffering delay will be reduced. If the period of silence is long enough the handover will not be noticed at all.

If a host is using an adaptive play-out buffer mechanism to optimize its performance it will have difficulties to cope with sudden delay spikes[17], even if the packets arrive within the acceptable end-to-end delay bound for users[23]. Thus, if a handover occurs in the middle of an upstream talkspurt, it is likely

[23]Cole and Rosenbluth[2] put the critical point around 180 ms, above which an increased end-to-end delay will degrade the perceived performance more.

that the receiving host will drop most of these delayed packets in that talkspurt. Nevertheless, studying the affect of the delay and loss patterns caused by 802.11 handovers (as presented here or including 802.1X and MIPv6 support) is an interesting extension to this work.

In this study we have assumed that the STA only has one WLAN card and that this card has a single transceiver. As G. Q. Maguire Jr. has previously pointed out, a straight-forward way to improve the handover performance would be to equip the WLAN card with a second receiver (or transceiver)[13]. The STA could then passively search for APs while continuing to communicate via the old AP, thereby masking the affect of the search phase. The STA could also continue to receive any packets sent by the old AP after the STA has entered the execution phase.

If the STA were equipped with two WLAN interfaces it would even be able to associate with two APs at the same time. At the time of the handover the "unused" interface could associate with the new AP, while the "active" interface continues to send and receive audio data via the old AP. If the new and old APs reside on different IP subnets, MIPv6 could be used to redirect the downstream data to the "unused" interface (how to shift the upstream data flow from the "active" to the "unused" interface is an implementation matter local to the STA). For mobility between APs on the same LAN one could of course use MIPv6 too, although simpler methods may be desirable, e.g., the "unused" card could be used to mask the search time just as for the case with the second receiver mentioned above, however, here active scanning could be used. It remains to be seen if devices with multiple WLAN interfaces or WLAN interfaces with a second receiver will appear on the market.

Acknowledgments

I would like to thank my advisors and colleagues at KTH for helping me with this study, in particular Björn Pehrson for providing the necessary equipment, Gerald Q. Maguire Jr. for support and feedback on the draft versions of the report, and Héctor Velayos and Erik Eliasson for fruitful discussions. Erik was also involved in the early phase of the work on the IAPP implementation.

This work would not have been possible without some free software packages provided by various persons around the world, Jouni Malinen (HostAP), Lennert Buytenhek (Linux bridge), Jean Tourrilhes (Linux wireless tools) and Colin Kuskie (Perl-module for statistics[11]) to mention a few.

References

[1] Ron Bean. Linux Clock Mini-HOWTO. Linux Documention Project (LDP), November 2002. http://en.tldp.org.

[2] R.G Cole and J.H. Rosenbluth. Voice over IP Performance Monitoring. *Computer Communication Review, ACM Sigcomm*, 31(2):9–24, April 2001.

[3] A. Helal, C. Lee, Y. Zhang, and G. Richard III. An Architecture for Wireless LAN/WAN Integration. In *IEEE Wireless Communications and Networking Conference (WCNC 2000)*, pages 1035–1041, September 2000. Chicago, Illinois.

[4] Linux hostap driver. http://hostap.epitest.fi.

[5] IEEE Std 802.11, 1999 Edition (ISO/IEC 8802-11: 1999) IEEE Standard for Information Technology - Telecommunications and Information Exchange between Systems - Local and Metropolitan Area Network - Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications.

[6] IEEE Std. 8021D, 1998 Edition, IEEE Standard for Information technology Telecommunications and information exchange between systems Local and metropolitan area networks Common specifications Part 3: Media Access Control (MAC) Bridges, 1998.

[7] IEEE 802.1X-2001, IEEE Standards for Local and Metropolitan Area Networks: Port-Based Network Access Control, June 2001.

[8] IEEE 802.11f/D3, Draft supplement to IEEE Std 802.11, Recommended Practice for Multi-Vendor Access Point Interoperability via an Inter-Access Point Protocol Across Distribution Systems Supporting IEEE 802.11 Operation, January 2002. Unapproved draft, subject to change.

[9] IEEE P802.11i/D3, Draft Supplement to Standard for Telecommunications and Information Exchange Between Systems-LAN/MAN Specific Requirements-Part 11: Wireless Medium Access Control (MAC) and physical layer (PHY) specifications: Specification for Enhanced Security. Unapproved draft, subject to change.

[10] David Johnson, Charles Perkins, and Jari Arkko. Mobility Support in IPv6. IETF draft <draft-ietf-mobileip-ipv6-21.txt>, February 2003. Work in progress.

[11] Colin Kuskie. Statistics::Descriptive perl module – Module of basic descriptive statistical functions. Version 2.6, Available at http://search.cpan.org/author/COLINK/Statistics-Descriptive-2.6/Descriptive.pm. Accessed July 2003.

[12] Linux bridge support. http://bridge.sourceforge.net.

[13] Gerald Q. Maguire Jr. Private communication, Royal Institute of Technology, May 2003.

[14] D. Mills. Network time protocol (version 3) specification, implementation. RFC 1305, Internet Engineering Task Force, March 1992.

[15] Arunesh Mishra, Minho Shin, and William Arbaugh. An Empirical Analysis of the IEEE 802.11 MAC Layer Handoff Process. Technical Report CS-TR-4395 UMIACS-TR-2002-75, University of Maryland, 2002. Available at http://www.cs.umd.edu/~mhshin/paper/ACMCCR-Mishra.Shin.Arbaugh.ps (Accessed January 2003).

[16] IETF Working Group on IP Routing for Wireless/Mobile Hosts (mobileip). http://www.ietf.org/html.charters/mobileip-charter.html.

[17] Sue B. Moon, Jim Kurose, and Don Towsley. Packet audio playout delay adjustment: performance bounds and algorithms. *Multimedia Systems*, (6):17–28, 1998. Springer Verlag.

[18] Colin Perkins, Orion Hodson, and Vicky Hardman. A Survey of Packet Loss Recovery Techniques for Streaming Audio. *IEEE Network*, pages 40–48, Sep/Oct 1998.

[19] Radia Perlman. *Interconnections*, chapter 3. Addison-Wesley, 2 edition, 1999.

[20] Mark Pozzobon. Evaluation of laboratory computers for use as traffic monitors. Technical Report 020924A, Centre for Advanced Internet Architectures (CAIA), Swinburne University of Technology, Australia, September 2002. Available online at http://caia.swin.edu.au/genius/020924A/.

[21] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: a transport protocol for real-time applications. RFC 1889, Internet Engineering Task Force, January 1996.

[22] IETF Working Group on Context Transfer, Handoff Candidate Discovery,and Dormant Mode Host Alerting (seamoby). http://www.ietf.org/html.charters/seamoby-charter.html.

[23] Jon-Olov Vatn. A roaming architecture for IP based mobile telephony in WLAN environments. In *Mobility Roundtable 2003*, May 2003. Stockholm, http://www.hhs.se/cic/roundtable2003/papers/S13-Vatn.pdf. Accessed May 2003.

[24] Héctor Velayos and Gunnar Karlsson. Techniques to Reduce IEEE 802.11b MAC Layer Handover Time. Technical Report TRITA-IMIT-LCN R 03:02, ISSN 1651-7717, ISRN KTH/IMIT/LCN/R-03/02–SE, KTH, Stockholm, Sweden, April 2003. Available at http://www.it.kth.se/~hvelayos/papers/TRITA-IMIT-LCN%20R%2003-02%20Handover%20in%20IEEE%20802.pdf, Accessed April 2003.