Experimental Evaluation of IEEE 802.11b and Mobile IPv6 Hand-off Times

Mai Banh, Lawrence Stewart, Grenville Armitage
Centre for Advanced Internet Architectures, Swinburne University of Technology
Melbourne, Australia
{mbanh,lstewart,garmitage}@swin.edu.au

Australian Telecommunications Networks & Applications Conference 2004 (ATNAC2004), Sydney, Australia, December 8-10 2004

Abstract - **Mobile IPv6 (MIPv6) allows hosts to move their physical and topological attachment points around an IPv6 network while retaining connectivity through a single, well-known Home Address. Although MIPv6 has been the subject of simulation studies, the real-world dynamic behavior of MIPv6 is only gradually being experimentally characterised and analysed. In this paper we review the use of MIPv6 to support mobility between independent 802.11b-attached IPv6 subnets, and experimentally measure the how long an end to end IP path is disrupted when a MIPv6 node shifts from one subnetwork to another (hand-off latency). We also measure 802.11b hand-off independent of MIPv6. Our testbed is implemented using FreeBSD 4.x, the KAME MIPv6 stack, Cisco Aironet Access Points and NetGear 802.11b network interface cards. Using our measured hand-off latencies we evaluate the likely performance impact of MIPv6 hand-off on a common webcam application and bulk TCP data transfers.**

Keywords- IPv6, Mobile IPv6, KAME, 802.11b, FreeBSD, Mini-ITX, Mobile hand-off

I. Introduction

In today's Internet, every host is assigned a fixed address that represents both the host's identity and the host's topological location on the IP network [14]. Yet the growth of IP-based data and voice applications in the context of mobile devices (e.g. 4th generation mobile phones and PDAs) and new access technologies (e.g. Bluetooth, GPRS, ADSL, etc...) is driving a desire to support mobility at the IP level – in other words, allowing an IP host to keep on communicating with other hosts while roaming between different IP subnetworks.

Given the emerging demand for IPv6 [15,16] connectivity in non-North American markets, we are focusing on the use of Mobile IPv6 (MIPv6) [5] as a platform for supporting mobile, interactive and real-time services. MIPv6 leverages earlier work on IP mobility for IPv4 (MIPv4) [1], and functions as a network layer routing solution for uninterrupted IP connectivity. Applications on a MIPv6-enabled node can survive physical disconnection and reconnection while changing their points of attachment to the Internet, independent of the underlying wired or wireless access technologies (such as 802.11b wireless LANs, (WLANs)).

Although the industry has developed a functional MIPv6 architecture, a question remains over how the dynamic service quality of an end to end IP path degrades when a MIPv6 node shifts from one subnetwork to another (an event known as "hand-off"). This paper documents our experimental characterisation of packet loss and hand-off latency during transitions from one IP subnetwork to another in a wireless 802.11b LAN environment. We utilise the KAME project's [2] MIPv6 extensions to the FreeBSD [3] kernel, and commercial 802.11b equipment. We discuss MIPv6 hand-off components and factors that influence the hand-off procedure, and measure the actual data-link and network layer hand-off latencies in a live 802.11b network. We believe our results provide useful quantitative data for other researchers modeling and designing MIPv6 deployments.

The paper is organised as follows: Section II is a general overview of 802.11 and MIPv6, Section III covers our experimental derivation of 802.11b's contribution to hand-off delay, Section IV reviews experimental estimation of MIPv6 hand-off latency. Section V demonstrates our critique on the KAME stack's MIPv6 behavior while section VI reviews some related MIPv6 hand-off research. In section VII, we evaluate the impact of MIPv6 hand-off on end-to-end application level performance. Section VIII concludes our work.

II. Background

A. Overview of IEEE 802.11 Specification

IEEE 802.11 is commonly used to provide a bridging service between a regular Ethernet LAN and mobile hosts, or between two Ethernet LANs. The relationship between the 802.11 service and the 802.2 data link layer is shown in Figure 1. IEEE 802.11 itself defines a MAC and Physical Layer. The 802.11 MAC performs fragmentation, packet retransmissions, and acknowledgments. The MAC Layer defines two different access methods - the distributed coordination function and the point coordination function.

802.2			Data Link Layer
802.11 MAC			Physical Layer
FH	DS	IR	

Figure 1. 802.2 & 802.11 Data Link and Physical Layers

In an 802.11 wireless LAN, each cell forming the system is a Basic Service Set (BSS) controlled by a base station called an Access Point (AP) [4]. 802.11 networks support both ad-hoc and infrastructure modes of operation. An ad-hoc network, also known as an Independent BSS (IBSS), is just a collection of 802.11 nodes communicating in a peer-to-peer manner.

An infrastructure mode network contains one or more APs (one or more BSSs). A node joins such a network through association with an AP only, and frames between any two devices must travel through the AP. The name of the BSS is called the Standard Service Identification (SSID). BSSs can share the same SSID among themselves.

Multiple BSSs connected by a backbone are seen as a single 802.11 network and collectively referred to as an

Extended Service Set (ESS). A station can move between the various BSSs in an ESS without losing connectivity, re-associating with a new AP if it becomes preferable to the current one.

B. Overview of Mobile IPv6

There are three key nodes in MIPv6 [5]:

- Mobile Node (MN) - a host or a router that changes its point of attachment from one network or subnetwork to another.
- Home Agent (HA) - a router on a MN's home network that intercepts, encapsulates and tunnels packets for delivery to the MN when it is away from home.
- Correspondent Nodes (CN) – An IPv6 node connected to any reachable IP network, and in communication with a Mobile Node. The CN may or may not be MIPv6-enabled.

Communication between these nodes occurs using a Care-of Address (CoA). The CoA provides information about a MN's current point of attachment to the Internet, and is made known to both the HA and any CNs by way of Binding Update (BU) messages.

MIPv6 operation is comprised of 5 elements [5]:

- Movement detection: MN can detect its movement to a new subnet using a range of information sources – e.g. the operation of IPv6 Neighbor Discovery, Router Discovery and Neighbor Unreachability Detection (NUD) or link layer triggers.
- Obtaining a new IPv6 CoA: Stateful or stateless address autoconfiguration, and duplicate address detection (DAD) protocols allow the MN to obtain a CoA when visiting a foreign network.
- Registration of an IPv6 (CoA): MN registers its new CoA with its HA and sends BU messages to all active CNs.
- Binding Management: The binding cache maintains Binding Acknowledgements (BA), Binding Updates (BU) and Binding Requests (BR).
- Returning Home/Deregistration: If the new network prefix matches the home network, the MN deregisters using a BU with a Home Address to Home Address Binding.

MNs are allowed to acquire multiple CoAs, allowing for multiple concurrent base station connections in a MIPv6 environment, and reduction in packet loss during hand-off.

Where the CN does not itself support MIPv6 directly, a bidirectional tunnel is established between MN and HA, and regular IPv6 connectivity occurs between HA and CN. If the CN supports MIPv6 then traffic can be delivered directly between the MN and CN, bypassing the HA [7]. This process is called Route Optimisation - the MN informs the CN of the MN's CoA by sending a Binding Update to the CN, and the CN sends packets directly to the MN's claimed CoA.

C. Return Routability in MIPv6

According to RFC 3775 [5], the Return Routability (RR) Procedure enables the CN to obtain some reasonable assurance that the MN is addressable at its claimed Care-of Address as well as at its home address. The CN would then be able to accept Binding Updates from the MN for Route Optimisation. Home Test Init (HoTI), Home Test (HoT), Care-of Test Init (CoTI), and Care-of Test (CoT) are the four messages used to ensure authorisation of those Binding Updates. RFC 3775 states that the procedure requires very little processing at the CN. The Home and Care-of Test messages can be returned quickly to MN.

A HoTI message is sent from the MN to the CN via the HA to acquire a home keygen token. The HoTI conveys the MN's home address and a home init cookie that the CN must return later. The MN remembers these cookie values to obtain some assurance that its protocol messages are being processed by the desired CN. The home keygen token is formed from the first 64 bits of the MAC. In response to a HoTI message, HoT is sent to the MN at its CoA via the HA. This means that the MN needs to already have sent a BU to the HA, so that the HA will have received and authorised the new CoA for the MN before the RR procedure. The HoT contains a home keygen token, home init cookie and home nonce index. The home keygen token tests that the MN can receive messages sent to its home address. For improved security, the data passed between the HA and the MN is encrypted as it is tunneled from the HA to MN. The home init cookie ensures that the message comes from a node on the route between the HA and CN. The home nonce index allows the CN to efficiently find the nonce value that it used in creating the home keygen token.

CoTI message is sent to the CN directly to acquire the care-of keygen token. The CoTI conveys the MN's CoA and a care-of init cookie that the CN must return later. In response to a CoTI message, a CoT is sent directly to the MN's CoA. The CoT is generated with a care-of keygen token, care-of init cookie and care-of nonce index. The keygen token is formed from the first 64 bits of the MAC. The care-of init cookie from ensures that the message comes from a node on the route to the CN. The care-of nonce index is provided to identify the nonce used for the care-of keygen token.

When the MN has received both the Home and Care-of Test messages, the RR procedure completes. As a result, the MN has the data it needs to send a BU to the CN.

D. MIPv6 hand-off latency

MIPv6 performs hand-off when the MN changes from one subnetwork to another. Hand-off latency can be defined as the period that communication between MN and CN is disrupted due to the MN performing hand-off. During the hand-off process, MN cannot receive packets from CNs – packets sent during this time are simply lost. Hand-off latency can be measured as the time between the MN's last packet from the old link and the first packet from the new link. Note that this measurement focuses on the network layer hand-off - movement detection, new CoA configuration and registration.

E. Role of Router Advertisements Beacon Interval in MIPv6

Router Advertisements (RAs) are a key method used by the MN to detect it has moved subnets, and thus trigger negotiation of a new CoA. Clearly the time between RAs on a given network affects the overall network layer hand-off time.

MIPv6 specifies RA periods MinRtrAdvInterval and MaxRtrAdvInterval to be between 30 ms and 70 ms [5]. Many attempts have been made to modify the standard RA behaviors in MIPv6 so that a MN can detect its movement faster. These approaches include Fast Router Advertisement [8] (where the

Copyright © 2004, ATNAC 2004, Australia
ISBN 0-646-44190-6

router unicasts an immediate reply to a solicitation) and Fast Router Discovery with RA caching in link-layer access points [9] (where the link-layer triggers a network layer RA upon detecting link-layer hand-off).

III. IEEE 802.11 HAND-OFF

An obvious choice for early MIPv6 deployment would be to link disjoint 802.11b wireless LANs (i.e. where each 802.11b network is independent, and not linked at the Ethernet layer to form an ESS). We chose to experimentally characterise MIPv6 hand-off in an 802.11b environment. In order to eventually identify the hand-off delays due to MIPv6 we first characterised 802.11b handover delays in isolation.

A. Evaluation of IEEE 802.11b hand-off latency

Link-layer hand-off in an IEEE 802.11 wireless network occurs when a MN changes its point of connection to the network, usually characterised by a move from one AP to another. This process results in an interruption of data transmission until the 802.11b client is associated with a new AP.

The link layer hand-off process is comprised of three sequential phases: detection, search and execution [10]. The detection phase refers to the realisation that a hand-off operation is required. The search phase refers to the acquisition of the information needed to perform the hand-off. The execution phase refers to the act of carrying out the hand-off procedure.

In order to experimentally quantify the average hand-off latency of an 802.11b network we built a simple IP network consisting of two independent 802.11b LANs linked by a regular Ethernet backbone. We caused a single 802.11b client to move back and forth between the two APs, simulating the kind of link layer move that would trigger a MIPv6 hand-off event.

B. Triggering a switch between Access Points

There are three methods to trigger a switch between APs:

1. Decrease the transmission power of the AP that the MN is currently associated with. The MN will detect the degraded signal strength and switch to another AP, as long as they both have the same SSID.
2. Ensure the APs have different SSIDs, and change the BSS configured into the client when an AP switch is required.
3. Administratively shutdown the 802.11 radio interface or physically turn off the AP the MN is currently associated with.

We used methods 1 and 2.

A simple bridged Ethernet network was established, with one fixed host communicating to a wireless host that had one of two APs to choose from at any given time. Both APs were connected directly to a common Ethernet hub (Figure 2).

A VIA Mini-ITX based system running FreeBSD 4.9 [3] was used as the wireless client, with a Netgear MA401 PCMCIA 802.11b wireless network card [11] (in a PCI to PCMCIA cradle) talking to two Cisco Aironet 1200 series access points running Cisco IOS version 12.2(11) JA. A generic Intel-motherboard based host running FreeBSD 4.9 was used to sniff traffic on the wired LAN side (the 'sniffer box'). The AP's beacon interval was set to the default of 100ms. In order to 'see' the affect of switching AP, we generated IPv4 ping (ICMP) packets at 10ms intervals across the 802.11b wireless link.

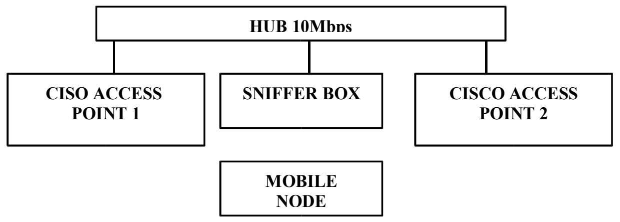

Figure 2. Testbed for 802.11b hand-offs

C. Method 1: Triggering hand-off by varying Access Point power levels

For the first approach to triggering hand-offs we configured both APs to be in the same BSS, “magicap”. The transmission power of AP1 and AP2 was respectively configured to the maximum of 100mW and minimum of 1mW for the duration of the trials.

Theoretically, when the sensitivity (or AP density) is set to high on the client side, the MN can trigger a hand-off when the signal quality of the currently associated AP drops below a certain threshold in order to avoid packet loss [10].

Our default AP density was 0 when the MN was not in range of an AP. When the MN associates with one AP, the default density changes to 1. When the default density was 1, the MN would always associate with the lowest power AP. When the default density was changed to 3 the MN would always (re)associate with the highest power AP.

The MN was set to toggle the access point density between 1 and 3 every five seconds, triggering hand-off events. Running tcpdump [12] (a packet sniffing tool) while pinging the MN every 10ms allowed us to calculate the hand-off time (from the timestamps of the last ICMP packet before hand-off and the first ICMP packet after hand-off). We ran 100 hand-off trials, yielding a mean hand-off time of 951ms. A tiny fraction (5%) of hand-offs took longer than 2s (up to 2.706s, we believe due to oddities in the wireless interface firmware search time).

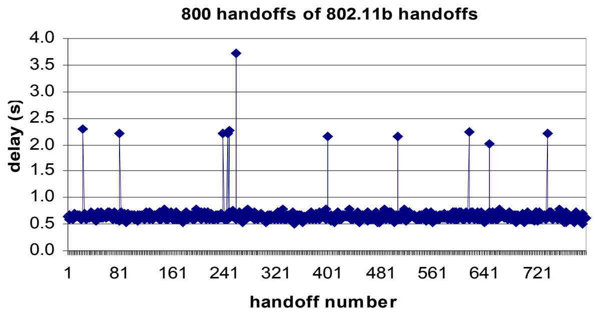

Figure 3. Hand-off times, alternating SSIDs, APs on same channel

Given the 10ms sampling interval, and the roughly 3ms RTT of our testbed, our estimates are +/- 13ms. Excluding the data points greater than 2s, we arrive at a mean 802.11b hand-off time of 864ms, ranging from 682ms to 946ms.

D. Method 2: Triggering hand-off by alternating Mobile Node's SSID association

Our alternative approach involved both APs being configured to transmit with the same signal power, but with different SSIDs -"magicap1" and "magicap2" for AP1 and AP2 respectively. Both APs were initially set to channel 10. Hand-offs were triggered by switching the client's wireless interface SSID (using FreeBSD's "ifconfig" command).

Figure 3 shows the scatter plot of 800 hand-off trials. Excluding the small number (1.34%) of samples over 2s, the mean switching time is 631ms (a minimum of 506ms and maximum 781ms). Figure 4 shows the distribution of hand-off times between 506ms and 781ms more clearly.

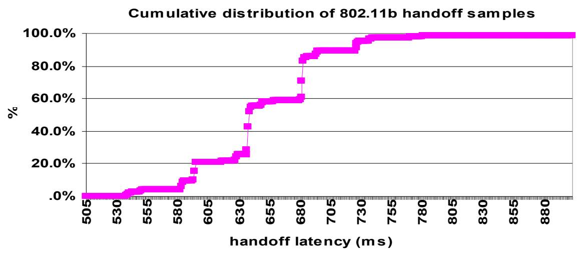

Figure 4. Hand-off times, alternating SSIDs, APs on same channel

We also re-ran these trials with the APs on different channels (AP1 on channel 4 and AP2 on channel 10). Table 1 summarises the results after excluding the few data points over 2 seconds.

	Both APs on Ch. 10	APs on Ch. 4 and Ch. 10 respectively
Mean 802.11b hand-off	631ms	667ms
Longest hand-off	781ms	825ms
Shortest hand-off	506ms	506ms

Table 1. Summary of alternating SSID hand-off times

E. Analysing our experimental 802.11b hand-off results

Method 1 deals with all 3 phases for hand-off - detection, search and execution. The 802.11b client must detect its movement, search for available channels to associate with and execute the hand-off. Method 2 eliminates the detection phase, resulting in a lower mean hand-off value (631ms compared to 864ms).

In the search phase, the MN has to scan all the different radio channels to be reassociated with an AP. The scanning can be done by listening for beacon messages from APs or by sending a probe request message on each channel and waiting on that channel for probe responses from APs. If the MN does not receive responses from APs it has to wait until a waiting timeout on each channel. The results of Mishra et al [13] find the 802.11b link layer hand-off to be in the range 58 ms to 397 ms. They also found out that the search phase was the most significant contributor to the hand-off latency. The type of wireless card firmware can have a large impact as well - we observe that the NetGear wireless NIC scans for channels in random order, where other wireless NIC firmwares can search through channels in ascending or descending order.

Our measured hand-off is generally longer than reported in other literature because we are measuring the entire time it takes for actual Ethernet level bridging to successfully resume after re-association with a new (or previous) AP. We also observed that if the 2 APs are in the same channel, the search time reduces a further 35 to 45 ms. Our primary goal was to account for the contribution of 802.11b handoff to our later MIPv6 handoff measurements, so we did not explore 802.11b-specific methods of improving link layer handoff times beyond the default settings of our commercial equipment.

IV. Measured Mobile IPv6 hand-off over IEEE 802.11b

We now experimentally determined the hand-off time when using MIPv6 over the 802.11b network.

A. The MIPv6 Testbed and Experimental Approach

Our MIPv6 testbed is shown in Figure 5. All NICs were explicitly configured to run at 10Mbps. AP1 (Access Point 1) was configured with SSID "magicap1" on channel 10. AP2 was configured with SSID "magicap2" on channel 4. We chose to trigger 802.11b network switching by changing the SSID every 20s.

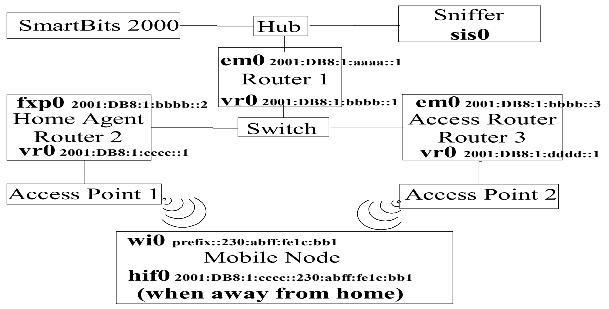

Figure 5. MIPv6 testbed

The first 2 trials saw the sniffer box used as a CN, by having it send IPv6 ping traffic at 10ms intervals to the MN and monitoring all inbound and outbound traffic with tcpdump. The hand-off time was calculated by measuring the time between the last ICMP echo response received from the MN on the current network to the first ICMP echo response received from the MN on the new network. We ran trials where the CN was both MIPv6-enabled and non MIPv6-enabled. We recorded 100 MIPv6 hand-offs for each case.

B. Hand-off with a MIPv6-enabled Correspondent Node

With the sniffer box acting as a MIPv6-enabled node, the MN establishes route optimisation with the node and thus avoids the overhead of relaying all traffic through the HA. Figure 6 shows a scatter plot of our measured hand-off times as a function of time. During these tests the RA interval was set to 30-70ms, consistent with current recommendations.

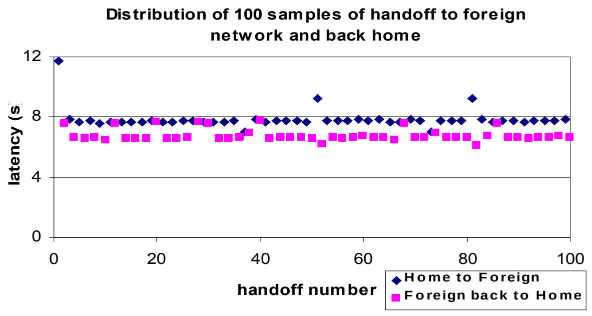

Figure 6. Hand-off times including bug times with CN as MIPv6 node

The time to hand-off from home to foreign network is plotted separately from the time from foreign back to home

Copyright © 2004, ATNAC 2004, Australia
ISBN 0-646-44190-6

network. Moving back to home is slightly faster than moving away. In this test, the average hand-off from home to foreign network took 7.770s where as the average hand-off from foreign back to home network took 6.779s.

We discovered that the KAME MIPv6 implementation had two bugs, which added three seconds to the hand-off latency in each direction. One bug in the handling of Binding Updates affected the movement of the MN away from home. A bug in the processing of Return Routability for Route Optimisation affected the movement of the MN towards home. We examine these in detail in a later section.

For the remaining analysis, we have subtracted the erroneous three seconds from our measured data to generate graphs and figures focused on the hand-off latency that is intrinsic to MIPv6 itself rather than the peculiarities of a particular KAME stack implementation. Figure 7 is a cumulative histogram of our results after adjusting for the three second bug, clearly showing the different hand-off latency in each direction.

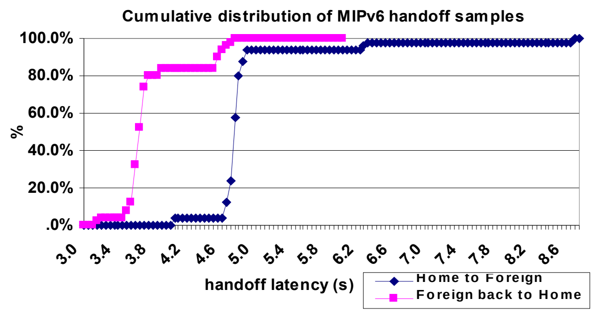

Figure 7. Hand-off times excluding bug times with CN as MIPv6 node

C. Hand-off with a non-MIPv6-enabled Correspondent Node

When a CN does not natively support MIPv6, the procedure of hand-off from home network to foreign network is similar**,** except that the CN will not reply to the MN with a CoT message when MN sends a CoTI message. The MN thus knows that the CN is not MIPv6 enabled, route optimisation will not happen, and a tunnel is established instead for communication between CN and MN via the HA.

We discovered that movement of the MN away from home triggers a similar KAME bug as when the CN is MIPv6-enabled. Hence the hand-off latencies reported here for the MN moving away from home have been adjusted down three seconds from what we actually measured.

Figure 8 and Figure 9 show our measured hand-off times in each direction – first as a scatter plot as a function of time, and then as a cumulative histogram.

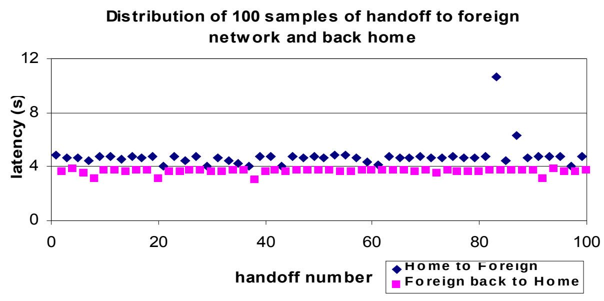

Figure 8. Hand-off times excluding bug times with CN as non-MIPv6 node

During the tests in Figure 9, the RA interval was again 30-70ms.

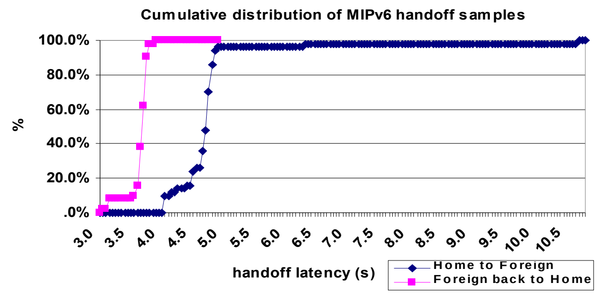

Figure 9. Hand-off times excluding bug times with CN as non-MIPv6 node

We also re-ran these tests with different RA intervals, and the results are shown in Figure 10. An interesting observation is that low RA intervals (less than one second) do not appear to make a substantial difference to hand-off time that we measured using our testbed. Our measured hand-off times based on the KAME implementation comprises a detection movement period of 3 seconds. Our results show that increasing the RA interval to 500-800ms only degrades home to foreign hand-off from 4.750 to 5.322 seconds (a difference of 0.572 seconds), while significantly reducing the amount of RA traffic (overhead) on the local networks.

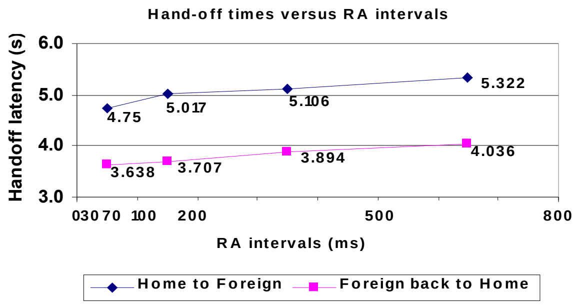

Figure 10. Hand-off times versus RA intervals

D. MIPv6 hand-off using a traffic generator as a CN

For calibration purposes, we ran a trial using a SmartBits 2000 precision traffic generator to transmit ICMPv6 ping packets at precise 10000 microsecond (10ms) intervals. This trial mirrored that of the CN acting as a non MIPv6 aware node, and obtained almost identical results: 4.770 seconds home to foreign network, and 3.660 seconds foreign to home network.

V. Critiquing the KAME stack's MIPv6 behavior

MIPv6 implemented by KAME aims to complete its hand-off procedures in the following steps:

1. Form new CoAs and start DAD: When MN attaches to a new foreign link, it starts to receive new RA messages. The MN then forms new CoAs using IPv6 address auto-configuration mechanism. DAD is performed by the MN in the next step to ensure MN's link local addresses are not duplicated (MN sends a Neighbor Solicitation message containing the tentative address).
2. Movement detection: based on the NUD mechanism [6]. MN checks reachability of default routers when it moves to foreign networks. This process takes three seconds. Movement detection and DAD for MN's link local address can be performed simultaneously.
3. Select a new CoA and send a BU to the HA: After NUD has completed, MN detaches all prefixes advertised by the unreachable routers. MN then selects a new CoA and sends a BU to the HA
4. HA receives the BU: and starts to perform DAD for the home address of the MN, if a MN moves from a home link to a foreign link, which takes one second. After DAD has completed, it returns a BA message to the MN. MN receives the BA and the hand-off is finished.

Utilising the tcpdump tool at the MN we were able to examine real MN procedures during hand-off and discover the two bugs in KAME's then-current MIPv6 implementation. The text below demonstrates the MN hand-off steps that occurred over our testbed.

A. When the CN is non-MIPv6-enabled node

MN begins to recognise it is on the new link after the L2 hand-off has occurred. The time reported in part III of this paper ranges from 506 ms to 825ms. The MN then receives RAs on its new link at a time dependent on the frequency of the RA beacons from the new network's router. With the default 30-70ms RA interval the MN will receive its first RA some random interval up to 70ms after the L2 hand-off completes. About 0.45ms after this first RA the MN completes address auto-configuration on the new link. The MN then starts to perform DAD by sending a neighbor solicitation message to see if any host is already using this auto-configured address on the new link.

In order to detect movement, MN sends three neighbor solicitations (NSs) to the old router address using its old link IP address to see if it is still reachable via the old default router. Each solicitation is sent at the interval of one second ± 0.55 ms. The first NS of this NUD is sent 0.47ms after DAD is performed. It means that the MN starts DAD and NUD simultaneously. It takes MN three seconds to complete NUD trials.

After NUD completes, the MN selects a new CoA as it has confirmed that its default router from the previous link is now unreachable. At this point, the MN is supposed to send a BU to the HA to register its new CoA. Yet we could not see any BU sent out to the HA, although there was a kernel log message that the KAME stack's BU timer has started. This is where the KAME bug occurs. It takes another three seconds (± 0.55 ms) before we actually see the MN send a new BU to register its new CoA with the HA.

Some detective work on our testbed, and discussions with KAME developers, revealed the cause. The first BU was in fact being sent immediately after the MN completed its NUD process. However, due to a bug in the ND cache the BU was being sent to the link layer address of the MN's *previous* default router. By the time the three second BU retransmission timer expires the ND cache has been properly flushed, and the second BU is correctly transmitted and the MIPv6 hand-off continues.

After another one second (± 5ms) the HA sends a BA to the MN's CoA. During this time HA performs DAD for the home address of MN. A tunnel between the HA and the MN's new default router, and hence path from CN and MN, is built within few ms. The MN is then ready to talk to the CN via the HA.

The experimental MIPv6 hand-off process consisted of:

MIPv6 hand-off = L2 hand-off time + RA delay + 3000ms + 3000ms (due to KAME bug)+1000ms = mean value of 7.75s

When the MN returns to its home network, the MN also first performs L2 hand-off and waits for a short random period to receive a new RA. As long as MN receives the first RA with MN's home prefix, it knows that it is in its home network. Therefore according to KAME implementation, the MN sends a BU to the HA to deregister its CoA immediately. However, the MN also initiates NUD (which lasts for three seconds) to confirm its movement. Hence, in this case:

MIPv6 hand-off = L2-hand-off time + RA delay +3000ms = mean value of 3.63s

B. When the CN is a MIPv6-enabled node

In this case, the MN's move processing is the same as when the CN is MIPv6 node except that there is Return Routability procedure. Due to the ND cache flushing problems at the time when NUD period ends, MIPv6 hand-off values in this case also include three seconds due to the KAME bug. Return Routability starts only few ms after the MN receives a BA from the HA. Return routability starts by a CoTI message being sent from the MN to the CN. The CN sends back a CoT message to MN within few ms. Less than one millisecond after the RR procedure, MN sends a BU to CN. Direct communication between MN and CN then begins (using new IPv6 routing headers which specify the MN's CoA as an intermediate destination in source address).

A similar procedure occurs when the MN returns home. Despite being home, the MN starts NUD for three seconds. As the MN is still in direct communication with CN it must perform the RR process to issue a new binding update to the CN after it completes home registration (including NUD). The MN then sends a BU to the CN after the necessary HoTI, CoTI, HoT and CoT message exchange. However, a bug in the KAME implementation means that RR begins immediately after the MN sends its BU sent to the HA (i.e. before home registration is complete). Thus the HA never forwards the MN's first HoTI packet to the CN for authorisation. A second HoTI, sent after a 6 second retransmission timeout period, does successfully reach the CN. As NUD completes in 3 seconds, hand-off time consequently results in a redundant period of 3 seconds. At this point the CN replies with HoT within a few ms, and roughly 0.ms later the MN sends the appropriate BU to the CN, and the foreign to home hand-off is complete.

The Table 2 below summaries our measured MIPv6 hand-off times excluding KAME code bugs.

MIPv6 hand-off including link layer hand-off	Average hand-off from home to foreign network (in s)	Average hand-off from foreign back to home network (in s)
CN is MIPv6 node (RA interval: 30ms –70ms)	4.770	3.779
CN is non MIPv6 node (RA interval: 30ms –70ms)	4.75	3.638

Table 2. Summary of MIPv6 hand-off trials

VI. Related MIPv6 Hand-off research

Cornal et al [17] investigated hand-off times using MIPL (a Helsinki University of Technology Mobile IPv6 for Linux program) over an 802.11b testbed. They do not provide much detail about the testbed setup nor the hand-off steps implemented by their version of MIPL. When the RA interval is between 0.5 seconds and 1.5 seconds, their measured hand-off times are 1.1 seconds returning to the home network and 1.8 seconds moving to foreign network. Related work by R.Hsieh et al [20], based on a simulation setup for an 802.11 network, gives their mean measured basic MIPv6 hand-off to foreign network around 5487ms.

D. Le et al [19] specify their MIPv6 implementation in great detail for WLAN mobile networks. They also used MIPL, and observed no packet losses in MIPv6 hand-off in MIPv6 WLAN networks while the CN was pinging their MN. Instead they saw only the RTT increase from 3.95ms to 89.23ms. Examining their published testbed, we suspect their MN movement only incurred L2 hand-off since the default router was still reachable when MN attached to the new link. It would appear their trials did not trigger actual MIPv6 hand-off.

N. Montavont et al [18] reported MIPv6 latency values over an 802.11b network ranging from around 300ms to 1.7s when the RA interval is 50ms. When RA interval is 1500ms MIPv6 hand-off latencies are comprised between 1.8s to 3s. Unfortunately there is little detail provided regarding the tools they used to measure hand-off latency.

Our results reflect an up-to-date KAME implementation consistent with the current MIPv6 RFC 3775 [5]. The NUD process counts most in the MIPv6 hand-off duration in KAME as it always takes 3 seconds. RFC 3775 states that “Due to the temporary packet flow disruption and signaling overhead involved in updating mobility bindings, the Mobile Node should avoid performing an L3 handover until it is strictly necessary”. The NUD process then helps to determine default or current router(s) of MN no longer reachable then helps to confirm L3 movement after MN has discovered routers and prefixes on the new link.

Specific optimisations are possible, both within each layer (link or network) and between each layer (using link layer state changes to expedite network layer awareness of the need for MIPv6 handoff). However, the focus of our work in this paper has been to measure the intrinsic limitations of a MIPv6 implementation where there are no such optimisations - indeed, our results provide further justification for tighter coupling between link layer and network layer protocols in order to significantly improve the resulting handoff times.

VII. Application performance over MIPv6 hand-off

A. Webcam performance over MIPv6 hand-off

An interesting question is how MIPv6 hand-off would affect a common webcam application where one of the parties was a Mobile Node in motion. To test this scenario we ran a two-party webcam conference between two Windows 2000 hosts through a FreeBSD 4.10-based bridge using a popular application, Yahoo!Messenger version 6.0.0.1643. The bridge repeatedly blocked and unblocked IP communication between its two network interfaces, timing the blocked state to mimic our measured MIPv6 hand-off latencies.

The bridge interrupted traffic 30 times, once every 20 seconds. At the receiving webcam we saw 30 instances where the smooth flow of video was disrupted - the duration of each disruption was noted and plotted. The cumulative distributions of conference disruption intervals due to hand-off from home to foreign network and foreign to home network are shown in Figure 11 and Figure 12 respectively.

Disruption to the actual video stream depends on how the webcam's transport and codec algorithms react to consecutive packet losses. Interruption at the IP level due to home to foreign hand-off had a mean interval of 4.75s. The webcam user at the receiving end saw the video stop for intervals ranging from 5s to 11s, with the majority of estimates sitting between 5s and 6s.

For foreign to home network hand-off (with an IP level disruption around 3.77s) the visual disruptions intervals ranged from 4s to 7s, with the majority falling between 5s and 6s.

Our test gives us an insight into the affect of MIPv6 hand-off on typical, two-party webcam scenario. It is clear that the webcam application takes between one and two additional seconds to recover from a loss of IP connectivity, compounding the time delay introduced by MIPv6 hand-off itself.

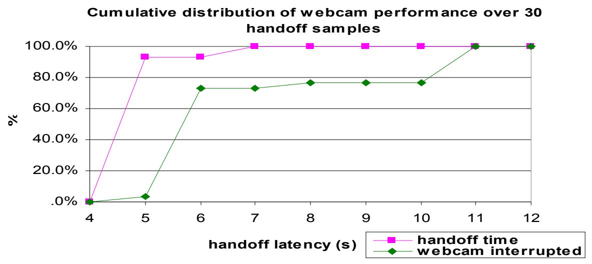

Figure 11. Webcam performance over hand-offs from home to foreign network

B. Wired LAN TCP throughput performance over MIPv6 hand-off

We also experimentally investigated the performance of bulk TCP transfers over an IP link that is affected by regular, emulated hand-off events. A FreeBSD-based bridge linked two FreeBSD 4.9 hosts over 100Mbit/sec Ethernet. We used nttcp to run a number of single-flow, bulk TCP transfers through the bridge. As with the webcam trials, the bridge was configured to regularly interrupt IP traffic flow in the same manner as would be experienced if the IP path ran through a MIPv6 link during hand-offs.

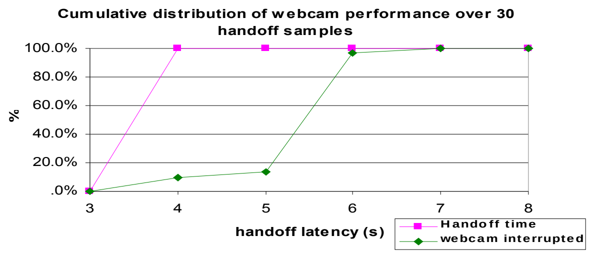

Figure 12. Webcam performance over hand-offs from foreign network back to home

Each trial run involved the unidirectional transmission of 3600Mbtes (61140 buffers of 60KBytes each) over a TCP stream while emulating hand-off events at frequency of 1, 2, 3, 4, 5, 6 and 7 hand-off per minutes. Hand-off duration was set to 4.75s for all cases and nttcp window size was always set to 16. When there were no hand-off events between the two hosts, nttcp throughput reached a mean value of 93.85 Mbits/s and the trials took approximately 5.37 minutes. Figure 13 shows the nttcp bandwidth versus hand-off rate. Data label on each plot specifies our measured throughput and the time nttcp taken to complete data transferring on each hand-off rate case. Certainly, hand-off disruptions play a relatively significant impact on TCP throughput as we can see that the more hand-off rate increases, the less throughput is achieved by nttcp. The decrease of the throughput seems to be linear until when hand-offs happen at a high rate every 8.6 seconds.

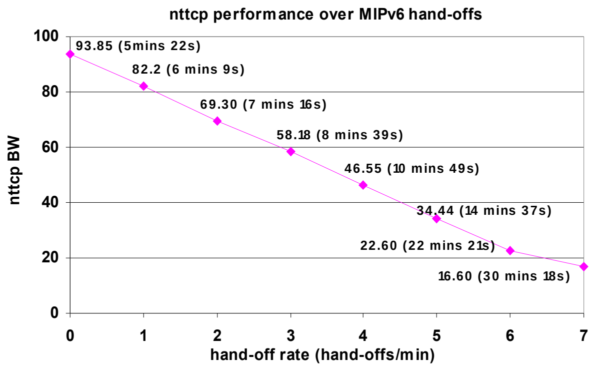

Figure 13. nttcp performance over MIPv6 hand-offs

VIII. Conclusion

By implementing an 802.11b wireless network and overlaying a mobile IPv6 network on top of it, we were able to experimentally trigger hand-off events and measure the time period during which connectivity was lost.

We found that real-world 802.11b hand-offs were typically completed in less than 700ms. The IP level disruption due to 802.11b and MIPv6 hand-off together was significantly higher - around 4.8 and 3.8 seconds depending on the direction of hand-off (home to foreign network or foreign to home network respectively). Tuning the router advertisement (RA) intervals from 30-70ms (the default) to 500-800ms did not significantly degrade these hand-off times. This suggests that short RA intervals may, in practice, not be worth the transmission overhead, particularly for resource-constrained environments (e.g. limited bandwidth or battery power).

These results clearly show that default MIPv6 would be highly disruptive to real-time and interactive applications during hand-off events, even if the underlying link-layer hand-off was instantaneous. We have also seen how simple implementation bugs can cause substantial increases in the hand-off latencies, regardless of the actual MIPv6 protocol itself.

Further research will be required to investigate how each component of the layer 3 (MIPv6) hand-off detection, configuration and registration times contribute to the overall hand-off time, and what factors can be used to reduce each component. Further work is also required to investigate packet loss and jitter during mobile IP hand-off operations. Benchmarking some of the available methods for Mobile Node fast movement detection e.g. predictive and non-predictive hand-off optimisations using RAs, fast handovers for MIP over WLANs using layer 2 triggers etc. would also provide useful information for those interested in mobile IPv6 and its applications.

References

[1] C.Perkins, "IP Mobility Support", RFC 3344, Internet Engineering Task Force, August 2002

[2] The KAME Project, http://kame.net/ (as of July 2004)

[3] The FreeBSD Project, http://www.freebsd.org/ (as of July 2004)

[4] P. Brenner, "A Technical Tutorial on the IEEE 802.11 Protocol", BreezeCom Wireless Communications, 1997

[5] D. Johnson, C. Perkins, and J. Arkko, "Mobility Support in IPv6", RFC 3775, Internet Engineering Task Force, June 2004

[6] T. Narten, E. Nordmark, and W. Simpson, "Neighbor Discovery for IPv6", RFC 2461, Internet Engineering Task Force, December 1998

[7] C. Perkins, "Mobile Networking through Mobile IP", IEEE Computer Society Tutorial. http://www.computer.org/internet/v2n1/perkins.htm (as of July 2004)

[8] J. Kempf, M. Khalil, and B. Pentland, "IPv6 Fast Router Advertisement", IETF Internet Draft, July 2004, draft-mkhalil-ipv6-fastra-05.txt

[9] J. Choi and D. Shin, "Fast Router discovery with RA caching in AP", IETF Internet Draft, July 2004, draft-jinchoi-dna-frd-00.txt

[10] 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

[11] Netgear 802.11b Wireless PC Card, http://www.netgear.com/products/prod_details.php?prodID=91 (as of July 2004)

[12] Tcpdump, http://www.tcpdump.org/ (as of July 2004)

[13] M. Arunesh, S. Minho, and A. William, "An Empirical Analysis of the IEEE 802.11 MAC Layer hand-off Process". Technical Report CS-TR-4395 UMIACS-TR-2002-75, 2002, University of Maryland.

[14] J. Postel, "Internet Protocol DARPA Internet Program Protocol Specification", RFC 791, Internet Engineering Task Force, September 1981

[15] S. Bradner and A. Mankin, "The Recommendation for the IP Next Generation Protocol," RFC 2460, Internet Engineering Task Force, January 1995

[16] S. Deering and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, Internet Engineering Task Force, December 1998

[17] T. Cornall, B. Pentland, K. Pang, "Improved Handover Performance in wireless mobile IPv6", The 8th International Conference onCommunication Systems, ICCS 2002 , Singapore, Volume: 2 , 25-28 November 2002

[18] N. Montavont, T. Noel, "Analysis and Evaluation of Mobile IPv6 Handovers over Wireless LAN", Mobile Networking and Applications (MONET), special issue on Mobile Networking through IPv6 or IPv4, Volume 8, Number 6, Kluwer Academic Publishers November 2003

[19] D. Le, D. Guo, and B. Wu, "Mobile IPv6 in WLAN Mobile Networks and its Implementation", 14th IEEE Proceedings on Personal, Indoor and Mobile Radio Communications, Proceedings PIMRC 2003, Beijing, China, Volume: 2, 7-10 September 2003

[20] R.Hsieh et al, "Performance Analysis on Hierarchical Mobile IPv6 with Fast-hand-off over End-to-End TCP", Global Telecommunications Conference, GLOBECOM '02.IEEE, Volume: 3, Pages: 2488 - 2492, Taipei, Taiwan, 17-21 November 2002

ISBN 0-646-44190-6