# L7-mobility : A framework for handling mobility at the application level

*Jean Tourrilhes*

jt@hpl.hp.com
Hewlett Packard Laboratories
1501 Page Mill Road, Palo Alto, CA 94304, USA.

*This paper introduces the concept of inter-domain mobility, to allow users to migrate their connectivity between different network domains. Most current mobility techniques are not well suited to handle inter-domain mobility. We present L7-mobility, a simple extension of current mobility practices for inter-domain mobility, which adds support for both hot and policy mobility. L7-mobility makes extensive use of our Connection Diversity framework and its Connection Manager to simplify implementation. We implemented L7-mobility in an HTTP proxy, which allowed us to evaluate the complexity, deployment and performance of this new technique.*

## 1 Introduction

The multiplication of wireless networks and services combined with the portable devices having multiple connectivity options, permits users to be mobile and still use the best network connection at all points. Switching between networks requires some form of mobility support. Most network domains start offering solutions for mobility within the domain, usually based on MobileIP [3], however little support is available for mobility across different domains.

A typical user may subscribe to a wide area provider offering cellular and public 802.11 connectivity. The same user may be able to connect at his office, accessing the company intranet via 802.11 and BlueTooth. At home, the user may have set up a private 802.11 or BlueTooth access point. The wide area provider enables seamless roaming throughout its network, so the user can remain connected while on-the-go, however when the user arrives at home, his device will not migrate to using his free private access point. As the user needs connectivity in those 3 domains, at work, at home and outside, he also needs his connectivity managed across those 3 domains.

Inter-domain mobility presents a unique set of challenges. Most existing mobility work has been to optimise the infrastructure to minimise the handoff time within this infrastructure. Inter-domain mobility require to handoff between two infrastructure that have nothing in common and may use totally incompatible mobility solutions, and need to easy to control by the user.

## 2 Applicability of current mobility techniques

Because there is a wide variety of technologies and needs, there exist many protocols to support mobility.

### 2.1 Link layer handoff

The most common techniques to handle mobility are deployed at the link layer. Cellular phone protocols allow clients to roam from one base station to another, and the 802.11 protocol deals with handoff between access points [1].

Those techniques have many advantages, they can be tightly optimised to the link layer, they are transparent to TCP/IP and any other network protocols, they are easy to deploy and they work well in the real world.

However, link layer handoffs protocols are specific to a link layer, so can't enable handoff across different link technologies (vertical handoff), and limited to a single IP subnet, so can't enable handoff across IP domains.

### 2.2 MobileIP

The IETF has defined MobileIP as the solution to handle all mobility problems in the Internet [3]. MobileIP creates a IPIP tunnel between the mobile client and its home agent to pretend to the rest of the network that the client is still located in its home network. The goal of MobileIP is to be universal and invisible to all applications and nodes.

Extensive research has been done on improving and extending MobileIP. Most of the technical issues with MobileIP are nowadays resolved, and various implementations exist. MobileIP usually can't perform as well as Link layer handoff (*section 2.1*), which is why those technologies are complementary.

The first issue with Mobile IP is that it requires some infrastructure deployment (the Home Agent). When deployed within a single domain, the network provider can take care of that. In our scenario, it is not clear which entity would deploy and manage such infrastructure, and how compatibility with the various other domains would be achieved. The user expects the mobility solution to work at his home which is typically not managed. Today, no MobileIP infrastructure to allow inter-domain mobility is deployed, despite having been standardised in 1996 [3].

The second issue is that, by forcing all traffic to go through the "home" network, MobileIP negates the real network topology, however, in many case this topology matters. The only place the user could deploy a Home Agent would be at home, which is usually behind a slow modem or broadband with slow downlink speed. Companies deploy Home Agents inside the firewall for security reasons, which means they are not accessible from the rest of the world.

### 2.3 Other TCP/IP protocols

Many other TCP/IP protocols have been proposed for mobility, such as TCP Migrate [5] ; they are similar to

MobileIP in spirit and techniques, and they also require specific infrastructure support. Cellular IP [4] is a micromobility technique enabling fast IP handoff within a single domain, and doesn't address inter-domain handoffs.

Ad-Hoc routing allows multi-hop wireless networks that can adapt to network reconfiguration. Any node can be mobile within such a network. P-Handoff is a trivial routing technique that allows two peers to perform a vertical handoff between two links they have in common [9]. Those two techniques deal only with mobility within a local wireless network, and can't handle mobility across domains.

### 2.4 Basic IP & Mobility agnostic applications

Most client applications, such as web browsing (HTTP) and e-mail (POP/IMAP), are agnostic with respect to the way they are connected to the Internet. As opposed to server applications, those applications don't need to be associated with a well known IP address.

Many people take advantage of this property to enable Basic IP mobility. While moving, they keep their device switched off. When they arrive in a new location, they switch their device on, DHCP autoconfigures the network, and they can resume the use of those mobility agnostic applications.

This is the method that people use in the real world, however, it has many limitations. This technique works only with mobility agnostic applications. The user must manually trigger the handoff (by switching off and on the network interface). Applications usually need to be restarted after the handoff, and any request that is active during the handoff fails.

### 2.5 Mobility aware applications

Mobile aware applications use explicit knowledge of the user present location and adapt to changes due to mobility. One example is SIP Mobile [6] which enables mobility for RTP applications through the SIP infrastructure.

Most often, those approaches require modifications of the server part of the application and work only in selected cases, this is probably why they are not widely deployed.

## 3 Definitions of mobility

Each mobility solution brings its own definition of mobility. The extent of mobility enabled by Basic IP mobility is limited compared to the one enabled by Mobile IP. We attempt to classify various types of mobility below.

### 3.1 IP mobility

The notion of mobility is user centric ; enabling users to move, however this definition is too vague and we need to restrict it to a network perspective. For example, for this work, we are not interested in the fact that user mobility create fading and multipath on the radio channel.

We define *IP mobility* as the impact of mobility on the TCP/IP protocols and applications. We ignore all user mobility that is transparent to TCP/IP, such as all mobility already dealt with at the link layer (*section 2.1*). This leaves mostly mobility across IP subnets and vertical handoff. IP mobility implies a change in IP routing between the mobile device and the Internet or other mobile peers.

### 3.2 Physical mobility and policy mobility

The original need for IP mobility support comes from the *physical mobility* of users in the real world. The geographical coverage of each wireless technology is limited, and networks are partitioned in IP subnets, therefore the IP connectivity available to a mobile user will change.

Our experience with Connection Diversity [11] has shown the value of *policy mobility.* If a device has multiple link layers, connectivity may be migrated at any time from one link layer to another based on link performance, application needs or other user policies, even though the user doesn't move. If a device use Co-Link [11], it may negotiate the activation of an alternate link and migrate connectivity to it. This policy mobility is not true mobility, but its consequences are exactly the same as physical mobility, the system must manage the change in IP routing resulting from the vertical handoff.

For example, a device might connect by default to an Access Point using BlueTooth, because it is lower power than the alternate 802.11 link. When the network usage exceed the capacity of the BlueTooth link, the device could negotiate the migration of connectivity on its higher capacity 802.11 link.

### 3.3 Hot mobility and cold mobility

We define the *application IP session* as the period of time when an application is using or bound to the TCP/IP stack. During an application session, either the application has an active IP connection with some remote entity, or some remote entity stores the IP address of the device.

The *application IP session* is not an explicit entity, is does not always map to TCP sockets lifetime, and depends on the overall behaviour of the application. This is not the device session either, the length of time where the device is switched on, even though in most cases switching off the device is the only way to make sure that no application IP session is active.

*Hot mobility* is IP mobility while some application IP session is active, whereas *cold mobility* is IP mobility while no application IP session is active. Basic IP mobility (*section 2.4*) can only handle cold mobility, whereas MobileIP (*section 2.2*) can handle both cold and hot mobility.

The development of smaller devices and more complex services increases the need for a permanent connection to the Internet, making hot mobility more desirable. Policy mobility also increases the need for hot mobility : even if the user is static, the system may migrate connectivity between links at any time based on various policies and the on-demand activation of those links.

## 4 Enabling mobility at the application level

Currently, the only solution for inter-domain mobility is Basic IP mobility, but this is limitative as it supports only cold mobility and requires explicit handoff. Our aim is to extend Basic IP mobility to make it more useful and user-friendly.

### 4.1 L7-mobility

The main characteristic of L7-mobility is that mobility is handled at the application layer, instead of within the network stack. Some applications are already aware of the network

topology, such as web browsers having settings for HTTP proxies. L7-mobility make applications fully aware of network topology changes and handle mobility by themselves.

Like with Basic IP mobility (*section 2.4*), applications always use a direct connection to the Internet using the IP address assigned on the interface via DHCP. The difference is that applications monitor connectivity changes and, after any handoff, automatically restarts their IP connections on the new link with the new IP address.

Obviously, L7-mobility works only for mobility agnostic applications, any application that need to be associated to a well known IP address would not work. L7-mobility can't offer low latency handoffs. Because of these limitations, it is complementary to MobileIP (one solution doesn't fit all), and both techniques need to coexist. Mobility support would be available simultaneously at all level of the networking stack, link layer, TCP/IP (via MobileIP) and at the application level. This is similar to network security support, which is available at the link layer (802.11 WEP), at TCP/IP (IPsec) and at the application level (SSL).

The difference with previous approaches to mobility aware application (*section 2.5*) is that L7-mobility doesn't require support on the server side or in the infrastructure and only addresses mobility agnostic applications.

### 4.2 Benefits for inter-domain mobility

L7 mobility is a technique that could be applied to many mobility scenario, however it mostly fits the need to inter-domain mobility and mobility in the home.

The main difference with other mobility solutions is that L7 mobility doesn't require any new infrastructure deployment and is client centric. It doesn't need any support from the infrastructure or domains, and handles inter-domain mobility transparently to each domain. It coexists with other mobility solutions and handles real Internet topologies (firewall, proxies, NAT...). It is totally contained within the client, it's easy for a user to deploy and he can have precise control on the connectivity and handoff policies.

L7 mobility fixes the main shortcomings of Basic IP mobility : it provides Hot Mobility and is invisible to the user (no need to manually restart applications). Those benefits would also apply to home mobility, because users typically don't deploy complex infrastructures in their homes.

### 4.3 The network model

Our usage model is a mobile client connecting to various network infrastructures. We assume only mobility agnostic applications ; the mobile client is mostly accessing various servers in the Internet.

The client may use a variety of wireless technologies to connect to Access Points in various IP subnets and reach the Internet (*fig. 4.3*). Those Access Points may be behind firewall or NAT, and may require the use of an application proxy. We assume no modification of the infrastructure.

The client may also connect via HomeSpot. HomeSpot are Access Points including various wireless technologies and application proxies. HomeSpots are similar to HotSpots [9],

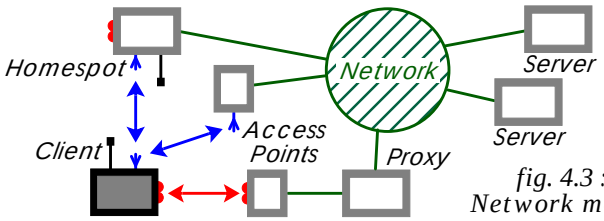

*fig. 4.3 : Network model*

however they are designed to be deployed at users's home and to not require management. We consider specific support in HomeSpot to optimise vertical handoffs.

### 4.4 The Connection Diversity framework

Various authors note that detecting network changes at the application level in a timely manner is a hard problem [6]. Another drawback of handling mobility at the application level is that it needs to be implemented in each application. Our main contribution is that we designed a complete framework solving those issues.

The application only handles the mobility part specific to them, such as restarting their IP connections and discovering remote application proxies. The application delegates all the generic mobility functionality and link specific management to the Connection Manager and interacts with it through a well defined API (*fig. 4.4*). This minimises duplication of functionality, keeps applications link agnostic and enables coordination between applications.

The framework may also include local application proxies (*fig. 4.4*). These local proxies interface between the application and the networking stack and allow an existing unmodified application to handle mobility.

## 5 The Connection Manager

The Connection Manager is the central piece of the Connection Diversity framework and where most of the functionality of L7-mobility is implemented.

### 5.1 Functionality

The role of the Connection Manager is to discover, evaluate, setup and monitor the various paths to the infrastructure on behalf of the various applications. It directly manages the various link layers and includes abstraction modules specific to each link layer.

The Connection Manager performs link discovery to find which paths to the infrastructure are available. It activates and configures link layers on-demand to enable their use, monitor them for failure, and disconnects them when idle. The Policy Manager selects the most appropriate link to connect to the infrastructure based on the current policy, applications requirements and link availability.

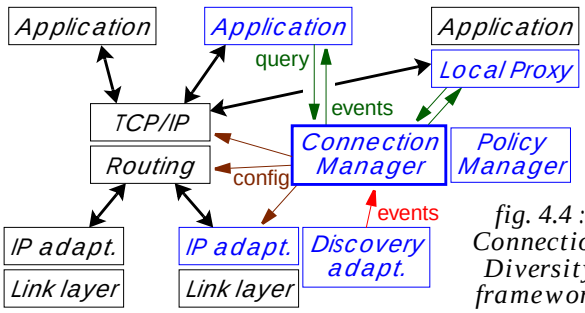

*fig. 4.4 : Connection Diversity framework*

Such Connection Manager includes many of the functionality of the Connection Manager we defined in our previous work on Connection Diversity [12]. Most mobility protocols have similar needs in term of link management, and our Connection Manager can easily support many of them. Because the Connection Manager is based on our previous work [12], we will only describe the addition and modification we did to it to support application level mobility.

### 5.2 Subnet management

The Connection Manager treats a remote communication as a local communication with a virtual neighbor that represents the entire infrastructure network [12]. The Subnet object manages a path to the infrastructure and can attach to a Peer object [9] or to an Access Point object (*section 5.3*).

The Subnet object only handles functionality common to all infrastructure paths. Once the underlying Peer or Access Point path is connected, the Subnet object triggers DHCP to get a valid IP address for the link. Then, it monitors both the DHCP process and traffic on the link to detect failures or idleness. One difference with previous types of paths is that subnet paths are not activated automatically by the demand mechanism but only at explicit application request.

### 5.3 Access Point objects

Peers are other nodes that support Connection Diversity and may offer infrastructure access, however we mostly want to deal with unmodified infrastructure. The new Access Point object represents an entity offering access to the infrastructure and which does not support Connection Diversity.

Access Point objects are specific to each link layer technology and are mostly simpler versions of the link specific Peer objects. They implement the link specific methods of the Connection Manager [13] ; they deal with AP discovery, IP adaptation and connection management (establishing link layer connections and monitor their failures).

For example, the BlueTooth Access Point object includes the code to create BNEP connections to standard NAPs [2] and process *blocked link* and *destroyed link* events [13]. The BlueTooth discovery manager discovers both Peer and standard NAPs.

### 5.4 Mobility API

The most fundamental addition to the Connection Manager is the Mobility API allowing applications to interact with it. This API is composed of various requests and events.

The *connect* request triggers the establishment of a path to the infrastructure. The Connection Manager connects the virtual peer representing the infrastructure, which trigger the selection of a path and its connection.

The *lookup* request indicates if a specific IP address belong to one of the locally discovered peers. This enables an application to properly handle local peers (*section 6.6*).

The connection manager reports all route changes through the API as *events*. This enables an application to know when a handoff has occurred and take appropriate action.

### 5.5 Mobile IP compatibility

L7-mobility and MobileIP should be used alongside, which make routing more complicated. MobileIP sets the default IP route on the IPIP tunnel to the home network, whereas L7-mobility uses a direct connection to the Internet. Different applications may use MobileIP and L7-mobility to simultaneously connect to the same IP address.

The Connection Manager uses an alternate routing table to set the direct route on an interface. It also sets a routing rule specifying that packets with a source address matching the current IP address of the interface use this alternate routing table. By default, packets use the home address of the client, so normal applications never see this alternate routing table, and therefore are routed on the IPIP tunnel. Only applications that explicitly bind to the current IP address of the interface use the alternate routing table and are routed directly to the Internet (which is the case for mobile aware applications).

Compatibility with P-Handoff [10] is not an issue as the host routes used by P-Handoff always take precedence.

## 6 The local HTTP proxy

Mobile applications need to be modified to handle mobility and interact with the Connection Manager.

### 6.1 Choice of application

We decided to try the L7-mobility concept with web browsing, which is a very popular application, and which is already mobility agnostic and topology aware.

Rather than modify a real web browser, we modified a HTTP proxy. The source code of HTTP proxies is usually simpler and well contained, and this enabled us to test a wide variety of HTTP applications, including web browsers. This is also a good model on how to retrofit existing applications with L7-mobility. The HTTP proxy runs locally on the device and provide mobility support to any HTTP application connecting to it.

### 6.2 Request management

The main functionality of the HTTP proxy is to process HTTP requests from the clients and forward them upstream (to the server or another proxy), and most of this is unchanged. The main addition is the code to handle mobility that uses the API provided by the Connection Manager.

The HTTP proxy used is Tinyproxy [8]. Tinyproxy uses a pre-forking model similar to popular Unix web servers, with one controller process and multiple worker processes.

After receiving a client request, the worker process request the establishment of the proper link and route from the Connection Manager. Then, it performs remote HTTP proxy discovery on the new link. At this point, it can forward the HTTP request upstream (to the server or the proxy).

The controller process monitors route change events from the Connection Manager. When an event indicates that the route the proxy uses is migrated to a new link, it sends a signal to the worker processes that were active. Each worker process then close its IP connections and restart it on the new link (after proxy discovery).

The proxy keeps track of the current state of the upstream route, and cache the various mobility settings while the route is active in a shared memory. Therefore, route activation and proxy discovery are only done for the first request of a session.

### 6.3 Proxy discovery (WPAD)

The client may need to use a remote HTTP proxy to connect to the Internet on some of the links, therefore the local proxy perform proxy discovery using the existing WPAD protocol (Web Proxy Address Discovery) [7]. The WPAD protocol works in two steps, the first step is the discovery of the Configuration URL, the second step is the downloading and processing of the proxy configuration file pointed by the URL (proxy.pac).

WPAD specify multiple methods for the first step, such as DHCP, SLP and well known DNS names. We use the DHCP method, which is the most efficient and most secure. The local proxy sends a DHCP-inform request to the DHCP server to get the DHCP attribute containing the Configuration URL.

The content of the proxy configuration file is a fragment of JavaScript which most often is browser specific. Our local proxy doesn't include a JavaScript engine, and proper parsing of JavaScript is not trivial, so currently it does some very crude parsing of the file to extract the proxy URL. This is in our opinion a big limitation of the WPAD protocol, as it precludes its proper use in simple HTTP clients.

If any part of the WPAD protocol fails, the proxy uses direct connection to the HTTP server (no remote proxy). When the network uses a transparent HTTP proxy, WPAD fails, the local proxy connects directly and the transparent proxy automatically acquires those connections.

### 6.4 IP restart

Most functionality added to Tinyproxy is well defined library calls. However, the restart of IP connections after handoff needs some change to the core logic of request handling. The current request needs to be interrupted and cleanly restarted without the client noticing.

Tinyproxy has a "streaming" model, where bytes are forwarded as they arrive between the client and the upstream socket ; and the complete request and reply is never present in memory. When the handoff occurs, the proxy has usually already started forwarding bytes of the reply to the client.

The current restart code supports only HTTP GET requests. During the handoff, the client socket is frozen. The old upstream socket is closed and a new one is opened. The request is sent again on the new upstream socket. From the new upstream socket, a number of bytes equivalent to what was already passed to the client is read and discarded. After that, byte forwarding is resumed to the client socket.

The fact that Tinyproxy needs to reread and discard the part of the reply already sent to the client is terribly inefficient. The use of HTTP byte range would dramatically improve performance, but this feature is only available with HTTP 1.1 and Tinyproxy supports only HTTP 1.0. To be able to support POST and dynamic requests, Tinyproxy would need to be modified to cache the requests. The integration in the actual HTTP client would solve this issue, as the client always has the full context necessary to resend the request.

### 6.5 HomeSpot optimisations

Closing and reopening IP connections adds overhead, therefore avoiding it is a worthwhile optimisation. One of the scenarios we target is policy mobility (*section 3.2*) when connecting to a HomeSpot (*section 4.3*). A typical HomeSpot includes multiple wireless interfaces and an HTTP proxy to control user access and offer local services.

The P-Handoff protocol allows vertical handoff without breaking IP connections on one-hop links [10]. To be able to use P-Handoff, the IP connection must end at the HomeSpot, and of course the HomeSpot must support P-Handoff.

We have therefore implemented the following optimisation : if the connection is routed via a Peer (as opposed to an AP - *section 5.2*), and if the proxy discovers a remote HTTP proxy on this Peer, the proxy uses P-Handoff. The behaviour of the proxy is slightly changed, it must use the Global IP address of the client as the source address and ignore route changes between links to the same peer.

This clearly breaks our requirement to not modify the infrastructure, but adding Connection Diversity support to the HomeSpot has other advantages. The client can now use Co-Link to discover the other links of the HomeSpot and their configuration [11]. Because Peers have unique identities, the client can cache the result of proxy discovery and skip proxy discovery when reconnecting to a known HomeSpot.

### 6.6 Peer to peer support

We also modified the local HTTP proxy for better handling of peer-to-peer connections [12].

The proxy verify if the target of the IP connection is one of the local peer by querying the Connection Manager. If it's the case, it bypass the infrastructure connection and proxy discovery, and always uses a direct connection, which is managed automatically by the Connection Manager using the On-Demand mechanism [12]. The proxy monitors route changes for peers so that it can properly fallback such peer connections to the infrastructure mode when the peer is no longer in range.

The proxy also checks if the hostname part of the URL is an ad-hoc name [12], and in those case returns a redirect to the client with the global name of this peer. Ad-Hoc names are short-lived with only local scope, this technique prevents the web browser to keep them in the UI and prevents the user to bookmark or e-mail them.

## 7 Implementation and results

Our claim is that L7-mobility is easier to deploy, but this need to be verified through a complete implementation.

### 7.1 Implementation and integration

The implementation of L7-mobility is based on our previous implementation of the Connection Diversity framework [12]. The Connection Manager, the Discovery Managers and the internal APIs have been extended to properly deal with Subnets and BlueTooth Access Points. The

Mobility API is implemented via a Unix sockets, both requests and events are described as XML fragments.

The HTTP proxy used is Tinyproxy 1.5.3 [8]. Most of the code to support mobility was written as a few well encapsulated library calls. For example, a simple call request the link establishment from the Connection Manager and return the new proxy configuration, another simple call monitor route events and propagate signals.

The main integration challenge is handling the restart of IP connections (*section 6.4*), and this would be different for each application. Integration in the a proxy is more difficult because of the need to hide the handoff from the client.

Our additions increased the code of Tinyproxy by only 20 kB (~2000 lines), so our approach definitely minimize the complexity and overhead on the application side.

### 7.2 Performance

Performance is not the main goal, therefore we only verified that performance was acceptable. This is the breakdown of a typical handoff from a BlueTooth Access Point to a 802.11b HomeSpot :

```
1) BlueTooth link breakage detection : 700 ms
2) 802.11b link and monitoring setup :  98 ms
3) DHCP to configure 802.11b link :   1789 ms
4) Proxy API processing time :           6 ms
5) WPAD, DHCP to get Config. URL :     109 ms
6) WPAD, query proxy.pac via HTTP :     10 ms
7) Parsing, connect to upstream proxy :  4 ms
Total elapsed time :                  2716 ms
```

Those numbers are to take with a grain of salt. First, they don't include the time needed to reread and discard the part of the HTTP reply already read, which depend on its size and can be very large. So, it would correspond to the performance of clients implementing HTTP byte-ranges (*section 6.4*).

Components were not optimised for speed, the DHCP time may be further reduced. Performance also depend on the links themselves, for example DHCP over BlueTooth is even slower (2-3 seconds).

A policy handoff would obviously not require delay to detect the link breakage, so would not require step 1. Vertical handoff with the same HomeSpot would use P-Handoff [10], so would not require step 3 to 7, and therefore be much faster.

### 7.3 Deployment

We tested L7-mobility with a standard BlueTooth Access Points (PAN NAP), with and without a remote HTTP proxy, with a HomeSpot supporting IrDA, BlueTooth and 802.11, and with a direct PPP link. Proper operation was observed in all those cases, and we tested handoff between interfaces of the HomeSpot and between the HomeSpot, the Access Points and the PPP link. We properly tested recovery from various failures modes, such as connection and DHCP failures.

The testbed was much simpler than a usual MobileIP testbed, as L7-mobility can directly use the existing infrastructure without having to change it.

## 8 Conclusions

Most mobile users can't use MobileIP for inter-domain mobility, but only BasicIP mobility. This doesn't offer hot mobility, which is needed to enable policy mobility and full usage of the various wireless links of a mobile device.

L7-mobility extends BasicIP mobility and makes applications themselves handle mobility, which adds support for hot mobility and automate handoffs. Like BasicIP, L7-mobility is limited to mobility agnostic applications, so is designed as complementary to MobileIP. L7-mobility doesn't depend on any infrastructure support, therefore is ideally suited for inter-domain mobility and mobility in the home.

To ease implementation, we defined a complete framework : a Connection Manager handles all the low level mobility tasks and link management, allowing mobile aware applications to be simple and link agnostic. Applications receive mobility events through a standard API.

Our implementation of L7-mobility in a HTTP proxy shows that it is relatively easy to integrate and can be deployed in the real world. The tradeoff is that handoff is slow, except in the case of the HomeSpot where it can use P-Handoff. Such a proxy allows to retrofit existing applications with L7-mobility without modifying them.

## 9 References


[1] *IEEE 802.11 : Wireless LAN medium access control (MAC) and physical layer (PHY) specifications.* IEEE.

[2] Bluetooth SIG. *Personal Area Networking Profile (PAN)*. http://bluetooth.org.

[3] C. Perkins. *IP Mobility Support*. RFC 2002.

[4] A. G. Valko. *Cellular IP - A New Approach to Internet Host Mobility*. ACM Computer Communication Review, January 1999.

[5] Alex C. Snoeren and Hari Balakrishnan. *An End-to-End Approach to Host Mobility*. Proc. of MobiCom 2000.

[6] Henning Schulzrinne and Elin Wedlund. *Application-Layer Mobility Using SIP*. ACM SIGMOBILE MC2R, Vol. 4, Number 3, July 2000.

[7] P. Gauthier, J. Cohen, M. Dunsmuir and C. Perkins. *The Web Proxy Auto-Discovery Protocol*. Work in Progress.

[8] Steven Young, Robert James Kaes and al. *tinyproxy-1.5.3.tar.gz*. http://tinyproxy.sourceforge.net/.

[9] D. Das, G. Manjunath, V. Krishnan, P. Reddy. *HotSpot! - a service delivery environment for Nomadic Users System Architecture*. HP report HPL-2002-134.

[10] Jean Tourrilhes & Casey Carter. *P-Handoff : A framework for fine grained ad-hoc vertical handoff*. Proc. of PIMRC 2002.

[11] Jean Tourrilhes & Venky Krishnan. *Co-Link configuration : Using wireless diversity for more than just connectivity*. Proc. of WCNC 2003.

[12] Casey Carter, Robin Kravets & Jean Tourrilhes. *Contact Networking: A Localised Mobility System*. Proc. of MobiSys 2003.

[13] Jean Tourrilhes. *On-Demand BlueTooth : Experience integrating BlueTooth in Connection Diversity*. To be published.