A Lightweight Macro-Mobility Framework

Karel De Vogeleer, David Erman, Markus Fiedler, and Adrian Popescu

Blekinge Institute of Technology, Karlskrona, Sweden
{kdv,der,mfi,apo}@bth.se

Abstract. This paper presents the design of a lightweight framework to provide vertical handover for macro-mobility on handheld devices. The framework is designed for mobile-controlled handover and does not require modification of the Internet infrastructure. The framework enables users to control the entire vertical handover process so handover decisions are driven by user preferences rather then ISP considerations. UDP tunneling is utilized as the basis for seamless roaming. Support nodes participate in mobility management by keeping track of the mobile users. We elaborate on a proof-of-concept implementation targeted to be deployed on the Android platform.

Key words: Mobility Management, Seamless Handover, Always Best Connected (ABC), Vertical Handover, Multihoming, Seamless Roaming

1 Introduction

Mobility is one of the prominent factors that describe computer devices today. Moreover, mobility in wireless networks becomes increasingly popular especially with the introduction of smart phones. Market studies show that the market share of smart phones continuously increase even in times of economic crisis.

In this paper we propose a seamless handover framework that enables mobility of users from one network to another. *Multihoming* is an important prerequisite in our framework, *i.e.,* maintaining multiple interfaces or connections over which Internet Protocol (IP) based communication is enabled, to exclude a potential single point of failure. *Multihoming* together with our framework enables users to participate in *seamless handover*. *Seamless handover* is defined as a handover in which no change in service capability, security, or quality is noticeable [10]. The technologies the framework support are technologies using IP addressing, *e.g.,* 3G, WLAN or fixed Ethernet. We refer to *vertical handover* when we are talking about migrating to another network that can be administrated by another domain. The reason for conducting a vertical handover is usually to improve Quality of Experience (QoE) towards users or to maintain connectivity. Efficient network selection, security, flexibility, transparency with reference to access technologies and provisioning of Quality of Service (QoS) are the most common parameters taken into account when performing handovers. This paper, however, does not cover the decision making mechanisms for deciding when to perform a vertical handover.

We differentiate between *proactive* and *reactive* handover. *Proactive* handover is used when referring to handovers initiated before network connectivity is lost. The initiating trigger is usually generated by an artificial intelligence mechanism, typically a decision engine. *Reactive* handover occurs when connectivity is lost and as an attempt to re-establish connectivity a vertical handover is initiated. Proactive handover thus is more suited to support seamless mobility than reactive handover. Though, proactive handover is not always possible in unpredictable circumstances, *e.g.,* weak signals while driving through a tunnel. Proactive algorithms are also known as Make-Before-Break (MBB) whereas reactive algorithms can be referred to as Break-Before-Make (BBM).

The most known mobility frameworks developed by the IETF are Mobile IPv4 (MIPv4) [13] and Mobile IPv6 (MIPv6) [6]. Unfortunately, MIPv6 is not adequate to support fast handover [7] and MIPv4 is not widely enabled between administrative domains even though it was preliminary proposed in 1996 [12]. Moreover, in the 90's numerous network architectures for mobility support were proposed. Yet almost 20 years later, Internet Service Providers (ISPs) seem reluctant to adopt mobility features in their infrastructure. Some possible reasons for this could be that they are not willing to invest additional money to extend their infrastructure, that no commonly accepted standard for handover is defined, or the fear of losing costumers for other ISPs. Also the diversity of existing access networks, the lack of interoperability of vendor equipment and the lack of techniques to measure and assess the performance challenge handover solutions. The adaptation reluctance of ISPs is a good motivation to look for alternative mobile-centered handover solutions if we want to benefit from mobility today.

To support mobility in the TCP/IP protocol stack limitations must be solved that incorporate cross-layer cooperation and awareness of concerned layers with regards to mobility. The congestion control mechanism, for example, in TCP is inadequate to perform well in mobile environments. The mechanism is unable to differentiate between packet loss as a result of link properties or as a result of handover performance degradations. Also, improper design of application towards mobile environments contribute to mobility issues.

Vertical handover solutions can be classified in several ways [10]. One approach is to divide the solutions from the point of view they tackle the problem, *i.e.,* from a user's or the network's point of view. User-centric, or mobile-controlled, handovers have the advantage that the user has full control over the handover mechanism. Whereas network-controlled handovers are usually faster in signaling because of, *e.g.,* traffic prioritization abilities, and place a smaller burden on the mobile device's resources. The ongoing research can also be divided into three other groups: handover in homogeneous, heterogeneous and IP-backbone networks. While efforts by IEEE focus on the first two groups, the third group is where the Internet Engineering Task Force (IETF) is active. 3GPP provides a unified mobility concept that primarily focuses on QoS support and MBB handover. Special Working Groups (WGs) are also active within the IETF which focus on auxiliary enhancements for mobility support.

Furthermore, solutions can also be classified according to their situation in the OSI reference model. The most notable vertical handover solution on the Data Link Layer is put forth as the IEEE 802.21 which is known as the Media Independent Handover (MIH) framework for seamless handover in heterogeneous networks [5]. MIH provides a framework pertaining methods and procedures for managing handovers irrespective of media. Mobile nodes and the network collect measurements on which handover decision are made. The MIH core exposes its facilities as an Application Programming Interface (API) to higher layers as a unified interface which is intended to work on any access technology. The framework presented in this paper fits the view of the MIH's core.

Overviews of existing mobility frameworks covering existing frameworks and vertical handover mechanisms can be found in [2, 8, 1] and others.

The remainder of the paper is organized as follows: we define the requirements of our framework in section 2 and the architectural design of our framework is presented in section 3. Section 4 reports the current status of our implementation efforts. We conclude the paper in section 5 where we state our future actions and present the conclusion of this paper.

2 Requirements

The vertical handover framework presented is designed to meet the requirements of the PERIMETER project and is granted by the EU STREP FP7. PERIMETER's main objective is to establish a new paradigm for user centricity in advanced networking architectures [11]. PERIMETER tries to tackle the seamless mobility problem from a user-centric perspective. Therefore seamless mobility can be achieved by actual user needs rather than business considerations. By deploying the vertical handover architecture we provide PERIMETER a framework for "Always Best Connected (ABC)" in a multiple-access multiple-operator environment. The presented framework provides a toolbox for vertical handover. It is up to the user or other authorized entities to command the framework to decide when to commence a vertical handover.

Transparency, user-centricity and deployability on handheld devices are the requirements for PERIMETER and thus for the framework. Transparency has a two-fold meaning in the sense that the framework must be transparent for applications, so legacy applications are able to use the framework. We will see further that this introduces an extra level of complexity. Secondly, the framework must be transparent to the end-user. The framework is not supposed to drain recourses of the phone that it operates on. Although handheld devices become more powerful, recourses as battery life, bandwidth and computational power are limited. When we take these requirements into account we are limited by possible solutions. Because of the mobile-controlled handover feature of the framework we must minimize the dependency on the Internet infrastructure. As a consequence a software-based solution suits the project needs.

The main difficulty in vertical handover lies in the fact that commonly used connection-oriented network protocols are not designed to deal gracefully with

mobile environments. Transmission Control Protocol (TCP) is a well-known example. A connection in TCP is defined as the sender and receiver IP address and port pair. If one of these duplets change during communication the connection will fail. Exactly this might happen during vertical handovers. Port numbers are not likely to change during handovers as opposed to the IP address. The current Internet architecture does not allow migrating from one network to another, neither intra-domain nor inter-domain, retaining the same IP address in networks where MIPv4 or MIPv6 is not enabled. A system is therefore desired by which the end-user can easily swap interfaces, *i.e.,* IP address, without interrupting ongoing services. This is contradictory to the idea of fixed interfaces in connection-oriented transport layer protocols. These protocols were originally designed without the consideration of mobile environments. A solution here is to deploy a transport protocol that can cope with mobile environments, *e.g.,* Stream Control Transport Protocol (SCTP). Yet when one must support legacy applications changing the transport layer protocol is not an option.

3 Architectural Design

We now describe the architectural design of a lightweight macro-mobility vertical handover framework.

As a start to solve the vertical handover problem we must provide (legacy) applications a fixed point to which they can *bind* to. The introduction of a fixed virtual network interface solves this problem. The virtual interface has a fixed IP address that does not change during the up-time of the device. This is in contrast with network interfaces that can come up, go down and change IP address on-the-fly in mobile environments. The technology of a virtual interfaces is well known and used. A popular implementation is, *e.g.,* the TUN-TAP driver used in for example the VPN project OpenVPN [3]. We can only operate the virtual address when both communication ends are using and maintaing the virtual address. Traffic going through the virtual interfaces without any additional measures are however invalid on the Internet. Therefore the traffic must be tunneled when leaving virtual network devices. Tunneling isolates the virtual interface from physical interfaces.

The tunnel virtually connects two end-nodes of a tunnel in such a way that they perceive that they are physically connected to the same network. Seamless roaming or seamless mobility is achieved by changing or relaying the tunnel's enter and/or exit point to another interface. This action changes the tunnel header and does not change the data encapsulated by the tunnel. This way, seamless mobility is achieved.

Tunnels have also drawbacks; the introduction of extra computational overhead and the inducuction extra data traffic. For User Datagram Protocol (UDP) tunnels each packet is appended by an IP header (20 B minimum), UDP header (8 B), and a tunnel header. The size of the latter is arbitrary, in our framework we use this space for measurement data on tunnels. Furthermore, performing a

vertical handover on the same access technology between two different networks is impossible without interrupting the service.

3.1 Roaming Strategies

Seamless roaming is achieved by means of tunneling data. Tunnels can be created, deleted and relayed, henceforth referred to as operations. Four different roaming strategies are elaborated. The roaming strategies apply to all the tunnel operations and define a manner how tunnels are managed and data is multiplexed into tunnels. We can distinct four different roaming strategies.

- Destination-oriented strategy: tunnels are created per destination. All outgoing data is grouped per destination and sent over the appropriate tunnel that leads to the destination. Tunnels are deleted when the destination of the tunnel disconnects or the tunnels are unused for a predefined amount of time. However, this strategy has limited flexibility when it comes to service and application differentiation. Only per-destination QoE can be targeted.
- Application-oriented strategy: data is assigned to a tunnel per application. This allows for service differentiation between connections belonging to different applications. Data from multiple applications are treated separately but can be sent over the same tunnel. Similarly, relaying connections happens per application. Even though this strategy might introduce duplicate tunnels to destinations, it is possible to provide application-differentiated roaming. This means concretely that the QoE per application can be maintained.
- Protocol-oriented strategy: tunnels are created per protocol, *i.e.,* data is bundled per protocol and then tunneled to the proper destination. Here it is also possible that multiple tunnels are established towards a destination. Transport layer protocols as well as application layer protocols are taken into account by the protocol-oriented roaming strategy. The protocol-roaming strategy offers the advantage to roam protocol streams independent of where the data emanates. Thus the protocol oriented roaming strategy is able to provide service differentiation.
- Service-oriented strategy: as the strategy's name reveals, data is treated per service. To enable this an extra level of information is needed to identify particular data as a specific services. In this strategy data is bundled per service and multiplexed into the proper tunnels. As a result QoE per service can be maintained.

The vertical handover framework presented in this paper currently utilizes an application-oriented strategy. This strategy has the most straightforward implementation. An application is bound to a socket and usually doesn't share the socket among others. Traffic coming from a particular socket is then handled per application's preference.

3.2 Mobility Management

Mobility Management is a very important issue in mobile environment. In extreme cases, a small physical movement can result in a change of network con-

nectivity. Hence the user might disappear and appear somewhere else in the network landscape if for example the physical IP address changes. When physical IP address changes randomly it is difficult to locate the user. By introducing a third party we can avoid this problem. Users report to the third party on which IP address they are available. The third party is henceforth referred to as the *Location Server* and has similar functionalities as the Home Subscribers Database (HSS) in the IP Multimedia System (IMS) architecture. An entry in the *Location Server* comprises of two compulsory elements and one optional element: a virtual IP address and a physical IP address that the user is reachable at and optionally a string identifier of the user. The later could be a DNS name or SIP identifier. Users in the system are expected to keep there entry in the *Location Server* up-to-date. Users report typically there location to the *Location Server* when they boot or turn off there device and after hand over.

The flow diagram of the signaling between two PERIMETER enabled nodes is shown in figure 1 and elaborated in the following itemization. Here we assume that *User A* initiates all the operations to *User B*. The *Location Server* supports both peers when needed.

- Setting up a connection to a user means concretely setting up a tunnel. The tunnel set-up procedure starts with retrieving the destination user's entry from the *Location Server*. Then a request is sent to the destination user to set up a tunnel with given settings. This request can be sent over any network interface available. Once the other side has parsed the request, configured its side of the tunnel and activated the tunnel, the destination sends an acknowledgement back to the initiating user. The acknowledgment is sent outside of the tunnel. Upon reception of the acknowledgement, the initiating user activates the tunnel at his side.
- Conducting handover is essentially the relay of a tunnel over another interface. The concept of relaying a tunnel is achieved by creating a second tunnel, the second tunnel might be using a different interface. Then all new traffic is forwarded into the new tunnel and the old tunnel is eradicated. The main concern in the relay of tunnels is the prevention of data loss, in case data might be residing inside the old tunnel when it is being shut down, and the reachability of the other side during handover for signaling and data exchange. The latter can only be indirectly influenced by performing the handover as fast as possible, for reactive handover, and by trying to initiate a handover at the right point in time before coverage vanishes completely *i.e.,* proactive handover. To circumvent the data-loss problem the old tunnel is only deactivated when the new tunnel is configured and running. In case of reactive handover data that is queued for sending is buffered while a new communication channel is being set up.
- Finally, tunnels are deleted when they are not being used anymore or when the destination or source user disconnects. The principles of data-loss discussed in above item applies to deleting tunnels as well. We do not want to loose data that might still reside in the tunnel upon deletion. Therefore we only

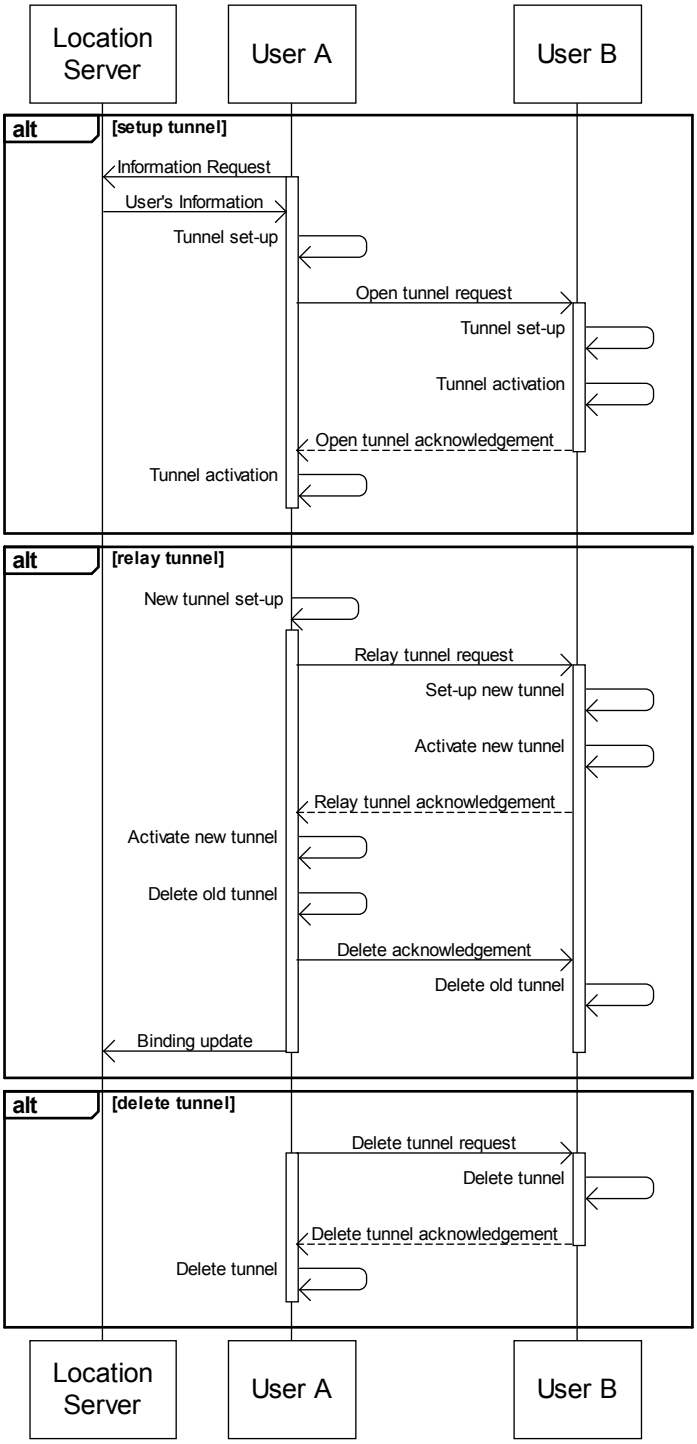

Fig. 1. Signaling scheme between mobile users operating the vertical handover framework and/or a location server.

tear down the tunnel when we are sure that both sides of the tunnel are fully aware of the operation.

In the part the structure is presented of the messages exchanged in the operations request described above. Acknowledgements are used to confirm or reject a previously received operation request. For the operations and acknowledgements, the content of their message body have a common structure containing the following fields:

1. sender-ip: sender's virtual IP address;
2. receiver-ip: receiver's virtual IP address;
3. tunnel-id: a Universal Unique ID (UUID) [9] to identify the tunnel;
4. seq-number: sequence number of the message;
5. operation: the operation to be performed [set-up, delete, relay, acknowledge]

The following fields are appended for operation requests only:

1. sender-ip: physical IP address of the sender;
2. receiver-ip: physical IP address of the receiver:
3. sender-port: port on which the sender is active;
4. receiver-port: port on which the receiver is active;

Two additional fields are appended for acknowledgement messages:

1. ack: can take *accepted* or *rejected* as value;
2. error: when the request is rejected this field contains the reason for rejection.

We will now elaborate more on the architectural design including the components that build up the framework.

3.3 Architecture of the Handover Framework

The Vertical Handover Architecture describes the design of the framework and is depicted in figure 2. The framework comprises five concrete blocks: the *Tunnel Farm* and *Mobility Manager*, the *Tunnel Manager*, the *Network Interface Manager* and the *Tunnel Catcher*. Additionally, a dedicated interface is exposing the frameworks functionalities to the outside world. The Vertical Handover framework is a passive element in the sense that it only acts upon external triggers, *i.e.,* no internal event generators are present.

Vertical Handover Interface: The *Vertical Handover* Interface is the relay between the Vertical Handover framework and the other processes wanting services from the framework. Incoming calls from the framework are multiplexed into the Vertical Handover framework. Similarly, outgoing calls to the controlling processes emanating from the framework are also managed.

Mobility Manager: The *Mobility Manager's* responsibility is to maintain the node's entry at the *Location Server*. At start-up of the framework, the *Mobility Manager* registers the framework at the *Location Server* and lists the physical interfaces on which it is currently active. If during the lifetime of the PERIMETER user physical network interfaces are activated or deactivated, then the *Mobility Manager* will report this to the *Location Server*.

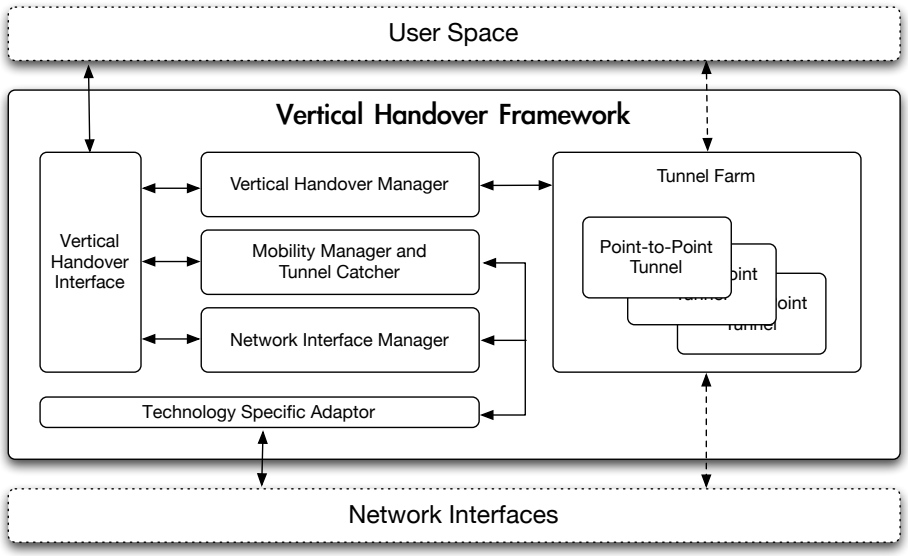

Fig. 2. Block scheme of the vertical handover framework. Solid lines represent control data flows whereas dashed lines depicts data traffic flows.

Tunnel Farm: The *Tunnel Farm* is a module that maintains a set of UDP tunnels. The farm's duties are creating, deleting and relaying tunnels. These three operations together with the intelligence implemented in the *Vertical Handover Manager* provide seamless roaming. The operations on tunnels are triggered by different events:

- Creating Tunnels: tunnels are created when data is about to be sent or when data is about to be received. This is a critical operation, as data must not be sent before a tunnel is established at both sides of transmitter and receiver. Therefore, a queue must be implemented to temporally buffer data until the tunnel is completely initialized. The *Tunnel Catcher* implements a mechanism that detects incoming tunnels and outgoing data. When such an event occurs the data is buffered and the *Vertical Handover Manager* will set up a new tunnel.
- Deleting Tunnels: when tunnels are not being used they are deleted. Either a timer expiration or a protocol connection closing signal can be the trigger to delete a tunnel.
- Relaying Tunnels: tunnels are only relayed on demand by the *Vertical Handover Manager*. Though, relaying is in essence a combination of creating and deleting tunnels. Therefore the tunnel relay operation is transparent to the *Tunnel Farm* as creating and deleting tunnels is commanded by the *Vertical Handover Manager*.

Tunnel Catcher: A negotiation must be held between the users who want to set up a communication channel. The negotiation pertains the configuration

settings of the tunnel, in particular the entry point and the exit point of the tunnel. Therefore a binary signaling protocol is deployed to serve as a communication substrate between mobility aware users and central administrative instances, *e.g.,* the the *Location Server*. The *Tunnel Catcher* manages the signaling protocol and cooperates with the *Tunnel Farm* that maintains the tunnels. Communicating with the *Tunnel Farm* happens through the *Vertical Handover Manager*.

When switching from one interface to another, *i.e.,* during handover, network addresses change. Therefore control messages, that are sent by the *Tunnel Manager* and the *Tunnel Catcher*, are sent over *UDP*. This approach avoids delays in connection set-ups that might affect the seamlessness of the handover. As a result random changes of interfaces are not an issue.

If, during handover signaling, no replies to request or acknowledgement are received within a specific time frame, the request is assumed to be lost and a retransmission is requested. The time frame for retransmission must be rather short because in order to conduct seamless roaming one must react quickly. The sequence number of the control messages then play an important role to detect possible duplicate requests.

Vertical Handover Manager: The *Handover Manager* directs the vertical handover procedure requested by any concerned instance. It decomposes the handover commands and delegates the work to the other blocks in the framework. Communication between these blocks is relayed through the Vertical Handover Interface.

Network Interface Manager: The *Network Interface Manager* deals with managing all the interfaces. Its duties are, *e.g.,* bringing interfaces up and down, configuring interfaces, logging into networks etc. These actions are technology-specific, and therefore, the Network Interface Manager relies upon the facilities offered by the Technology Specific Adaptors.

Technology Specific Adaptor: The *Technology specific adaptors* responsibilities are translating common operation offered by access technologies into a common API so that the vertical handover framework can communicate with interfaces in a unified manner.

4 Implementation

A preliminary description of the implementation of the framework is described in this section.

The current implementation of the framework is designed to run on Google's Android [4] platform. The motivation behind that is that the operating system is accessible, *i.e.,* open-source network stack to implement our framework, and the operating system is Linux based. The latter allows us to run framework on any Linux based device. Android also provides a useful set of tools to manage

handheld devices recourse's. Applications in Android run inside a Dalvik virtual machine which is based on Java technology. To deploy our framework inside a virtual machine is however not desirable because this would introduce unnecessary process delays and be disadvantageous for the seamlessness of handovers. Therefore we opt to implement the framework mostly in kernel. This approach minimizes changes to the original path that data would take when traversing the network stack and optimizes computational requirements.

The framework including the virtual interface is implemented as a Linux kernel module. The virtual interface is partly based upon the GRE tunneling driver. All outgoing traffic passing through the virtual driver is multiplexed into the appropriate tunnel according to the active roaming strategy. Outgoing traffic is also forwarded by the virtual interface to the higher laying infrastructure. Communication between the framework, residing in the kernel, and the controlling process in user-space happens through a UDP socket interface.

5 Conclusion and Future Work

In this paper we presented a lightweight framework for vertical handover. UDP tunnels are utilized as substrate for vertical handover. When migrating from one interface to another tunnels are relayed accordingly. This does not affect the data that is traveling through the tunnels as these events are transparent for the end user. We elaborated on implementation and design details and we discussed different strategies in relaying tunnels depending upon the kind of service that is offered towards the user of the mobility framework.

We are currently conducting experiments and assessing the performance analysis accordingly to show the viability of the framework. Preliminary outputs show that the framework is running properly, though some parts in the low-level code can be optimized to yield better performance, *e.g.,* table lookups can be accelerated.

Furthermore, the current design of the framework is prone to erroneous messages. These could be originating from malicious entities and other causes. A simple security mechanism is already present in the control signaling under the form of a sequence number. However this does not exclude any type of attack. Security issues and erroneous messages must be addressed in the future.

References

1. Mohammed Atiquzzaman and Abu Ahmed Reaz. Survey and classification of transport layer mobility management schemes. In *16th International Symposium on Personal Indoor and Mobile Radio Communications*, Berlin, Germany, September 2005.
2. Marc Emmelmann, Sven Wiethoelter, Andreas Koepsel, Cornelia Kappler, and Adam Wolisz. Moving towards seamless mobility – state of the art and emerging aspects in standardization bodies. *Springer's International Journal on Wireless*

Personal Communication – Special Issue on Seamless Handover in Next Generation Wireless/Mobile Networks, 2007.
3. M Feilner. *OpenVPN: Building and Integrating Virtual Private Networks*. Packt Publishing, 2006.
4. Google. http://www.android.com/, 2009.
5. IEEE. Draft ieee standard for local and metropolian area networks: Media indipendant handover. *P802.21/D8.0*, December 2007.
6. D. Johnson, C. Perkins, and J. Arkko. Mobility Support in IPv6. RFC 3775 (Proposed Standard), June 2004.
7. T. Johnson, R. Prado, E. Zagari, T. Badan, E. Cardozo, and L. Westberg. Performance evaluation of reactive and proactive handover schemes for ip micromobility networks. In *Wireless Communications and Networking Conference, 2009. WCNC 2009. IEEE*, pages 1–6, April 2009.
8. Deguang Le, Xiaoming Fu, and Dieter Hogrefe. A review of mobility support paradigms for the internet. *IEEE Communications Surveys and Tutorials*, 8(1-4):38–51, 2006.
9. P. Leach, M. Mealling, and R. Salz. A Universally Unique IDentifier (UUID) URN Namespace. RFC 4122 (Proposed Standard), July 2005.
10. J. Manner and M. Kojo. Mobility Related Terminology. RFC 3753 (Informational), June 2004.
11. Perimeter. http://www.ict-perimeter.eu/, 2009.
12. C. Perkins. IP Mobility Support. RFC 2002 (Proposed Standard), October 1996. Updated by RFC 2290, obsoleted by RFC 3220.
13. C. Perkins. IP Mobility Support for IPv4. RFC 3344 (Proposed Standard), August 2002. Updated by RFC 4721.