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 than ISP considerations. UDP tunneling is used 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.

Keywords: Vertical Handover, Always Best Connected (ABC), Multihoming, Seamless Roaming, Mobility Management.

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 suggest a seamless handover framework that enables mobility for 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. 3G, WLAN and fixed Ethernet are well-known examples of IP based substrates. *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]. We refer to *vertical handover* when migrating from one technology to another. Networks accessed through these technologies can be administrated by different domains. 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 process for adequately deciding when and where to perform a vertical handover.

We differentiate between *proactive* and *reactive* handover. *Proactive* handover is used when referring to handovers completed before network connectivity is

P. Chatzimisios et al. (Eds.): MobiLight 2010, LNICST 45, pp. 567–578, 2010.

lost. The initiating trigger is usually generated by a mechanism employing artificial intelligence , 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 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 are referred to as Break-Before-Make (BBM). Proactive handover is thus better suited to support seamless mobility than reactive handover.

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 deployed even though it was initially 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. Possible reasons for this are that they are not willing to invest additional money to extend their infrastructure, no commonly accepted standard for handover is yet defined or the fear of losing costumers to 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 adoption reluctance of ISPs is a good motivation to look for alternative mobile-centered handover solutions if we want to benefit from mobility today.

Limitations must be solved in order to support mobility in the TCP/IP protocol stack. These limitations include incorporation of cross-layer cooperation and awareness of concerned layers with regards to mobility. For example the congestion control mechanism 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 applications 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 the 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. Network-controlled handovers are usually faster in signaling, *e.g.*, because of traffic prioritization abilities, and place a smaller burden on the mobile device's resources. Vertical handover frameworks can also be divided into three other groups: handover in homogeneous, heterogeneous and IP-backbone networks. While the efforts done by IEEE focus on the first two groups, the third group is where the Internet Engineering Task Force (IETF) is active. The 3rd Generation Partnership Project (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 Open System Interconnection (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 that are used to make handover decisions. The MIH core exposes its facilities as an Application Programming Interface (API) to higher layers. The unified interface is intended to work on any access technology. The framework presented in this paper fits the view of the MIH core.

Overviews covering existing vertical handover frameworks and mechanisms can be found in [1,2,9] 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 in section 3. Section 4 reports on 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 Handover Requirements

The vertical handover framework presented is part of PERIMETER, a STREP project granted by EU FP7. PERIMETER's main objective is to establish a new paradigm for user centricity in advanced networking architectures [11]. PERIMETER approaches the seamless mobility problem from a user-centric perspective point of view. 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.

Transparency, user-centricity and deployability on handheld devices are the needs of 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. This introduces an extra level of complexity. Also, the framework must be transparent to the end-user. The framework is not supposed to drain resources of the device that it operates on. Although handheld devices become more powerful, resources as battery life, bandwidth and computational power are limited. When we take these requirements into account, we are limited to certain solutions. Because of the mobile-controlled handover feature of the framework we must minimize the dependency on the Internet infrastructure. Well-known and studied handover technologies, *e.g.*, MIPv4 and 3GPP, thus do not fit our requirements. As a result, a software-based solution is needed.

The main difficulty in vertical handover lies in the fact that commonly used connection-oriented network protocols, *e.g.*, Transmission Control Protocol (TCP), are not designed to deal gracefully with mobile environments. A connection in TCP is defined as the sender's and receiver's IP address and port pair. If one of these duplets change during the communication, the connection will

fail. This might even 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 technology to another, neither intra-domain nor inter-domain, retaining the same IP address in networks when MIPv4 or MIPv6 are not enabled. A system is therefore desired by which the end-user can easily swap interfaces, *i.e.*, IP addresses, without interrupting ongoing services. This is contradictory to the idea of fixed interfaces used in connection-oriented transport layer protocols. These protocols were originally designed without the consideration of mobile environments. A solution can be 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 challenging.

Furthermore, the framework must be able to cope with Network Address Translations (NATs), Port Address Translators (PATs) and firewalls. Telecom companies and ISPs use NATs and PATs technologies throughout their networks and retail products. A part of these only seem to cope with UDP and TCP connections. For ISPs it is also usual to allow only communication emanating from within their networks. This is a severe impediment for pure Peer-to-Peer (P2P) technologies. All these limitations create a complex environment for designing a straightforward handover framework.

3 Architectural Design

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

A virtual fixed network is created on top of the existing communication channels to provide (legacy) applications a fixed point to which they can *bind* to. Applications are connected to the virtual network through a virtual interface. The virtual interface has a fixed IP address that does not change during the uptime 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 concept of virtual interfaces are 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 maintaining the virtual address space. Traffic going through the virtual interfaces without any additional measures are however invalid on the Internet. We therefore tunnel the traffic going through the virtual network devices over real interfaces. A tunnel virtually connects two end-nodes in such a way that they perceive they are physically connected to the same network. Seamless roaming or seamless mobility is then achieved by transmitting the tunneled data over one of the physical interfaces. This action alters the tunnel header, yet it does not change the data encapsulated by the tunnel.

The proposed framework utilizes User Datagram Protocol (UDP) tunneling. UDP packets are prone to vertical handover, NATs and PATs are able to cope with them.

Tunnels have also drawbacks; the introduction of extra computational overhead and the extension of packet headers and results in less data per PDU. For UDP tunnels each data unit is preceded by an additional IP header (20 B minimum), UDP header (8 B), and a tunnel header. The size of the latter is arbitrary. Our framework uses this space for QoS measurement purposes on the 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. The roaming strategies apply to all tunnel operations and how tunnels are managed and data is multiplexed into tunnels. We differentiate between 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. 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. Data from multiple applications are treated separately but can be sent over the same tunnel. Similarly, relaying connections happen per application. Even though this strategy might introduce duplicate tunnels to destinations, it is possible to provide application-differentiated handover. 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 its proper 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 handover protocol streams independent of where the data emanates. Thus the protocol-oriented roaming strategy is able to provide protocol differentiation.
- *Service-oriented strategy*: data is treated in this case per service. To enable this, an extra level of information is needed to identify particular data as a specific service. 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 does not share the socket among others. Traffic emanating from a particular socket is then handled as per each application's preference.

3.2 Mobility Management for PERIMETER

Mobility Management is a very important issue in mobile environment. In extreme cases, a small physical movement can result in a change of network connectivity. Hence the user might disappear and appear somewhere else in the network landscape. As a result the physical IP address will change. Unpredictable changes of physical IP address makes it difficult to locate users. By introducing a third party we can solve this problem. Users report to the third party at which IP address(es) they are available. The third party is henceforth referred to as the *Location Service*. The service has similar functionalities as the Home Subscribers Database (HSS) in the IP Multimedia System (IMS) architecture.

The *Location Service* can be, *e.g.*, a Distributed Hash Table (DHT) in which all mobile nodes are participating or could be implemented as a server operated by a Mobile Virtual Network Operator (MVNO). A distributed network of servers hosted by the mobile users at their home may also provide a solution. Many other configurations are possible.

An entry in the *Location Service* 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 latter can for example be a DNS name or SIP identifier. Users in the system are expected to keep their entry in the *Location Service* up-to-date. Users report typically their location to the *Location Service* when they boot or turn off their device and after handover.

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

- Setting up a connection to a mobile user means concretely setting up a tunnel. The tunnel set-up procedure starts with retrieving the destination user's entry from the *Location Service*. A request is sent to the destination user to set up a tunnel with given settings. This request can be sent over any available network interface. 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. All messages are sent outside of the tunnels. Upon reception of the acknowledgement, the initiating user activates the tunnel at his side.
- Performing a handover is essentially the relay of a tunnel over another physical interface. The relay of a tunnel is achieved by creating a second tunnel. All traffic is forwarded into the new tunnel and the old tunnel is removed. The main concern in the relay of tunnels is the prevention of data loss. Data might be residing inside the old tunnel when it is being shut down. The reachability of the other side must be maintained during handover for signaling and data exchange. To address the latter the handover must be carried out as fast as possible. For *proactive handover* the handover must be initiated at the right point in time before connectivity vanishes completely. 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

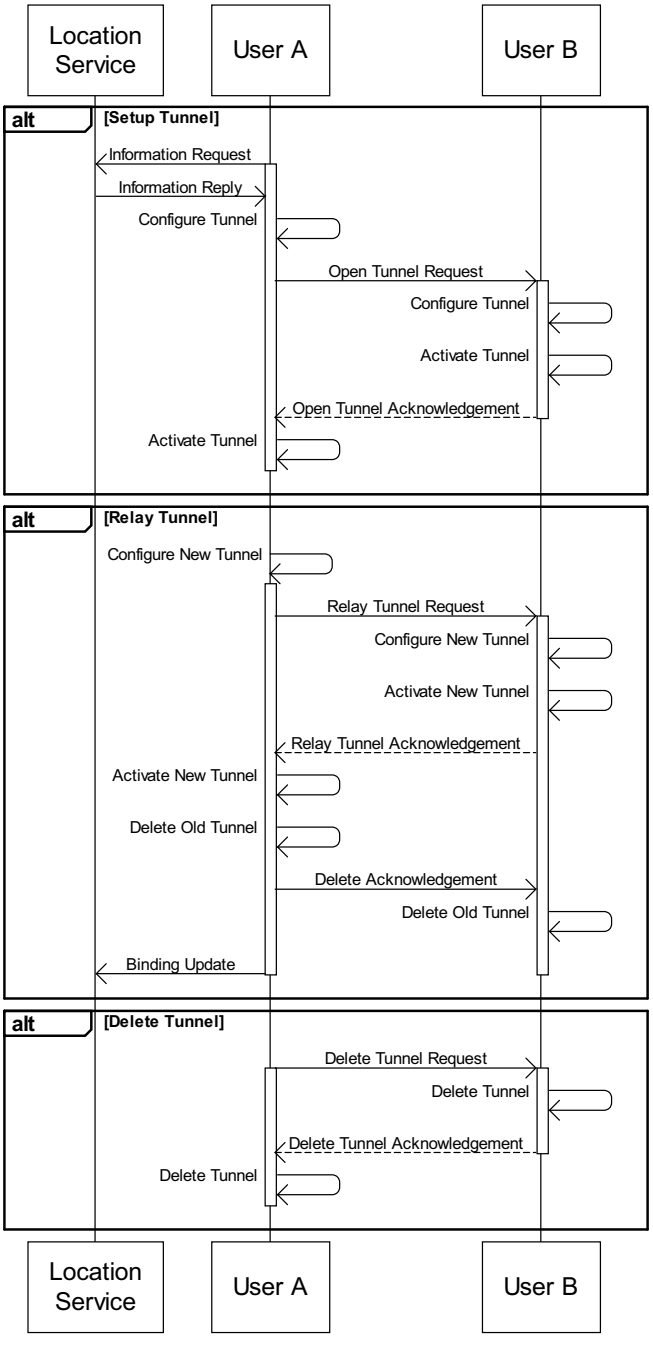

Fig. 1. Signaling scheme between the location service and mobile users operating the vertical handover framework

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

The structure of the control messages are presented in the following enumeration. *Control messages* are exchanged during an operation 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: Universal Unique ID (UUID) to identify the tunnel [9];
4. seq-number: sequence number of the message;
5. operation: operation to be performed [set-up, delete, relay, acknowledge].

The following fields are appended for operation requests only:

1. sender-ip: sender's physical IP address;
2. receiver-ip: receiver's physical IP address:
3. sender-port: sender's port number;
4. receiver-port: receiver's port number.

Two additional fields are appended for acknowledgement messages:

1. ack: status of request [accepted, rejected];
2. error: reason for acceptance or rejection (optional).

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 *Vertical Handover Manager*, the *Network Interface Manager* and the *Tunnel Catcher*. Additionally, a dedicated interface is exposing the frameworks functionalities to the outside world and an interface is assisting network interface calls. 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.

Tunnel Farm: The *Tunnel Farm* is an entity that maintains a set of UDP tunnels. The farm's duties are to create, delete and relay tunnels. These three operations together with the intelligence implemented in the *Vertical Handover Manager* provide seamless handover. The operations on tunnels are triggered by different events:

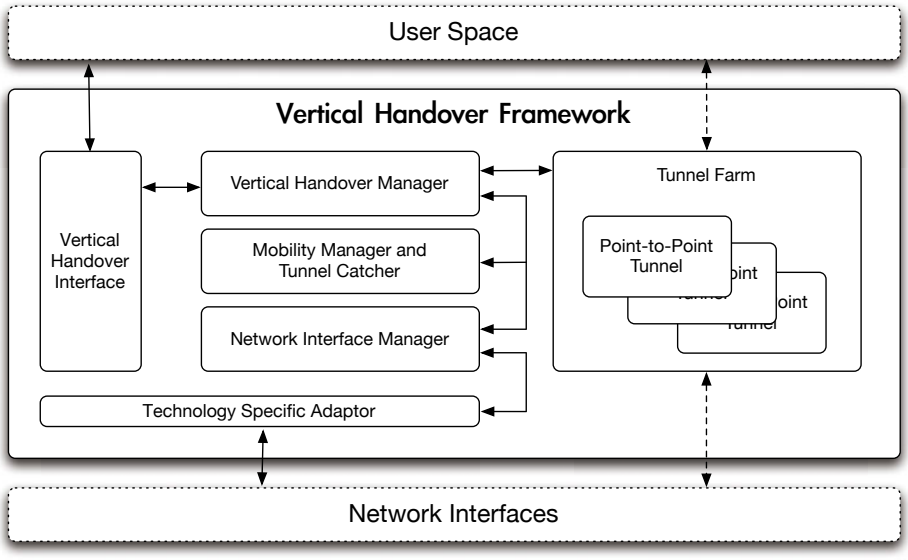

Fig. 2. Block diagram of the vertical handover framework. Solid lines represent control message flows whereas dashed lines depict user data flows.

- 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 cannot be sent before a tunnel is established at both sides of transmitter and receiver. Therefore, a queue must be implemented to temporarily 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 direct the configuration of 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 directed by the *Vertical Handover Manager*.

Vertical Handover Manager: The vertical handover procedures requested by any concerned instance is directed by the *Handover Manager*. The manager 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.

Tunnel Catcher: A negotiation must be held between the users who want to set up a communication channel over the mobility framework. 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 *Location Service*. The *Tunnel Catcher* manages the signaling protocol and cooperates with the *Tunnel Farm* that maintains the tunnels. Communication with the *Tunnel Farm* happens through the *Vertical Handover Manager*.

When switching from one interface to another, *i.e.*, during handover, network addresses change. To avoid connection problems the UDP protocol is used to communicate between handover frameworks. Using UDP avoids delays in connection set-ups, *e.g.*, as induced by TCP, that might affect the seamlessness of the handover.

If, during handover signaling, a request is not replied or acknowledged within a specific time frame, the request is assumed to be lost. A retransmission of the request is then initiated. The time frame for retransmission must be short because in order to conduct seamless handover one must react quickly.

Mobility Manager: Responsibilities involve maintaining the node's entry at the *Location Service*. The manager is mostly active during startup and closing of the mobility framework and during handover.

Network Interface Manager: All network interfaces are managed by the *Network Interface Manager*. 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 adaptor's* responsibility is translating common operations offered by access technologies. A common API is exposed so that the vertical handover framework can communicate with all interfaces in a unified manner.

Vertical Handover Interface: User-space utilities to configure the behavior of the handover system can communicate with the framework through the *Vertical Handover Interface*.

4 Implementation

The current implementation of the framework is designed to run on a Linux machine with special focus on Google's Android [4] platform. The motivation is that the operating system is accessible, *i.e.*, has an open-source network stack to implement our framework, and the operating system is Linux based. Android also provides a useful set of tools to manage the handheld devices resource's.

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

The framework including the virtual interface is implemented as a Linux kernel module. The virtual interface is partly based upon the IPIP 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 layers in the network stack. Communication between the framework, residing in the kernel, and the controlling process in user-space happens through a UDP socket interface or the `/proc` file system.

Preliminary experimental outputs show that the framework is working properly, though some parts in the low-level code could be optimized to yield better performance, *e.g.*, IP address lookups in tables can be accelerated.

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 technology to another tunnels are relayed accordingly. This does not affect the data that is traveling through the tunnels as these events are transparent to 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. The results will then be compared to other vertical handover frameworks and the findings will be published accordingly.

Furthermore, the current design of the framework is prone to erroneous messages and does not handle wrongly formatted commands very well. These could be emanating 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 and commands must be addressed in the future.

Acknowledgement

We would like to thank PERIMETER, a STREP project, granted by EU FP7, and MOBICOME, a EUREKA project, granted by VINNOVA for the funding of our research in vertical handover technologies. We also thank the anonymous reviewers for their valuable input.

References

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