Network Selection Box: An Implementation of Seamless Communication

Stefan Chevul[1], Lennart Isaksson[1], Markus Fiedler[1], Peter Lindberg[2], and Roland Waltersson[2]

[1] Dept. of Telecommunication Systems
School of Engineering
Blekinge Institute of Technology
371 79 Karlskrona, Sweden
{stefan.chevul, lennart.isaksson, markus.fiedler}@bth.se
[2] Saab Communication,
SE-351 80 Växjö, Sweden
{peter.lindberg, roland.waltersson}@saabgroup.com

Abstract. During recent years, it has become evident that mobility functions will have a profound impact on current and future wireless networks. Users expect service connectivity anywhere and anytime without having to think about the underlying communication systems used at that particular moment in time.

On this background, this paper presents a ready-to-deploy implementation of a mobility framework that supports seamless communication and represents an important enabler for adaptive applications through its simple QoS feedback mechanism. The framework selects the best available network through a decision algorithm that takes advantage of both experience with different network types for certain types of services and a link performance monitoring concept. The impact of the proposed framework on performance in terms of processing and throughput overhead is also discussed.

1 Introduction

The emergence of mobile networks has led to mobile end-users who expect access to information sources from anywhere at anytime. Such access should preferably be implemented in a seamless way: the user should be able to use a service without even having to think about which network technology is used at the moment. If a change of network technology is necessary, for instance due to the fact that a user leaves the coverage area of a Wireless Local Area Network (WLAN) hot spot and has to be connected via General Packet Radio Service (GPRS) instead, that change should happen more or less "on the fly", *i.e.* during ongoing communication without breaking the session. Thus, with *seamless communication*, we mean roaming between different types of networks or network operators in a handover-fashion, preserving connectivity to the selected service as far as possible and minimizing the performance degradation perceived by the application.

The main goal is to be Always Best Connected (ABC). This is defined as always connected according to a decision based on different static and dynamic criteria aiming at optimizing the perception of a certain type of service.

Unfortunately, the Internet Protocol (IP) is not designed to deal gracefully with mobility. In IP, the point-of-attachment to the Internet is uniquely identified by the node's IP address. Thus, the IP address is tied tightly to the network where the device is located. In order to keep the Internet connectivity, the mobile device has to change its IP address every time its point-of-attachment to the Internet changes. This makes it impossible for the node to maintain transport layer connection during a location change. In other words, the connectivity breaks and the corresponding data transmission has to be re-initiated.

The Internet Engineering Task Force (IETF) has proposed Mobile IP (MIP) [1] as a solution for mobility in the future all-IP networks [2]. Although MIP has been standardized, it is hardly implemented in the Internet. The main reasons are the limited number of IPv4 addresses and MIP mechanism issues such as the triangular routing, frequent and long distant registration updates, and single-point of failures [3]. MIP also requires additional network elements to be introduced in the network, *e.g.* Foreign Agent and Home Agent *etc.* This has a direct impact on the cost of deploying a MIP enabled network. Floroiu *et al.* [4] reports of a case study of seamless handover in MIP for WLAN/GPRS. The results show that the Round Trip Time (RTT) is one of the important parameters indicating performance degradation. Seamless handover was only considered between two different types of wireless technologies.

In this paper, we present an implementation and evalutation of a new mobility management concept that supports seamless communication independetly of MIP. The concept is called Network Selection Box (NSB). Ideally, a NSB has several networks to select from *e.g.* General Packet Radio Service (GPRS), Universal Mobile Telecommunications System (UMTS), Wireless Local Area Network (WLAN), Digital Audio Broadcast (DAB), *etc.* Based on knowledge of the requirements of an application in terms of performance and cost and accessibility, the NSB will choose an appropriate network that fulfills these requisites at the lowest possible expense. In case an available network cannot meet the requirements but still provides connectivity, it is considered as as limited appropriate. Such a limitation might be known beforehand [5] – the service needs more than the network in question can offer – but it might also arise during operation, *e.g.* due to bad transmission conditions. The latter is revealed by more or less continuous monitoring of the different network paths. The NSB achieves seamless communication by hiding the change of current point-of-attachment to the Internet from the application through a virtual network interface (TAP), thus making the handover transparent for the application. Furthermore, through control messages to the application, the NSB provides a simple Quality of Service (QoS) feedback mechanism. One of the side effects of the NSB is the overhead in terms of more data to be sent and more processing to be performed. In some way this can be considered as the price of achieving seamless communication in the existing access networks without needing to make any changes in these networks.

The remainder of the paper is organized as follows. In Section 2 foundation of mobility issues are described together with a proposed solution. The architecture of the proposed mobility framework is described in detail in Section 3, while Section 4 presents a performance assessment of the NSB implementation in order to reveal its impact on the data streams as such. Finally Section 5 presents conclusion and outlook.

2 Foundations of Mobility and Seamless Communication

The main challenge when it comes to mobility is the fact that IP protocol does not support any mobility by itself. This is reflected in the way IP addresses are used, mainly as topological locators and network interface identities. Thus, every time a mobile node changes the point of attachment to the Internet, it changes its IP address. *E.g.*, a switch from WLAN to any cellular network, such as GPRS or UMTS, results in a change of IP address. Most transport protocols, *e.g.* Transmission Control Protocol (TCP), can not handle IP address changes without breaking the communication session.

In order to attain seamless communication, the communication session must be maintained as far as possible to ensure minimal packet loss and performance degradation perceived by the application. This can be achieved by hiding the change of the IP address from the application, *i.e.* making the handovers transparent to the application. The NSB adopts a virtual network interface (TAP) for this purpose, see Figure 1. The TAP device consists of a driver running in kernel mode and origins from the open source VPN project OpenVPN [6]. In

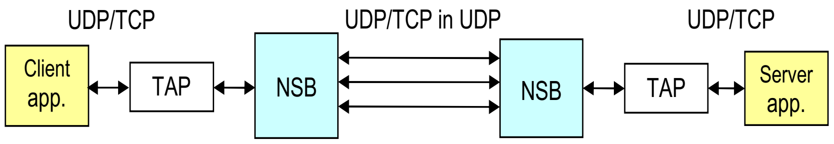

Fig. 1. Virtual Network Interface in the NSB

most respects, the TAP device works as any network interface; it will show up in Microsoft Windows XP as just another network connection, to which an IP address may be assigned. By allocating a low metric to the TAP device, we ensure that any application will bind its sockets to the TAP, unless if the application specifically states another network interface. At the sender, the original packet is encapsulated in a new User Datagram Protocol (UDP) datagram or Transmission Control Protocol (TCP) segment before sending it through the physical network interface to the Internet, as depicted in Figure 2. Thus, with the aid of the TAP device, a tunnel is implemented and the change of IP address is hidden from the application. On the receiver side, the outer header (by default a UDP header) is removed, and the original datagram sent by the application

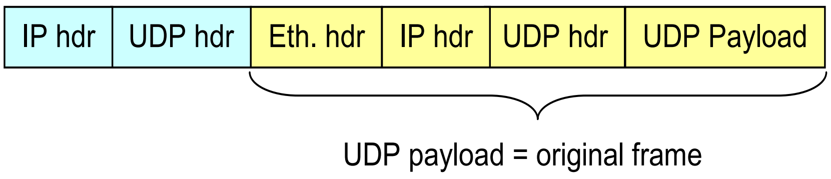

Fig. 2. Encapsulated packet for tunnelling purpose

is recovered. The application socket bound to the TAP device will receive the datagram seamlessly.

3 Network Selection Box (NSB) Architecture

The NSB is implemented as client-server architecture; the building blocks are shown in Figure 3. The NSB consists of the following building blocks: TAP device, NSB Controller, Network Selector, Link Monitor, Link Parameter database and Roaming Policy.

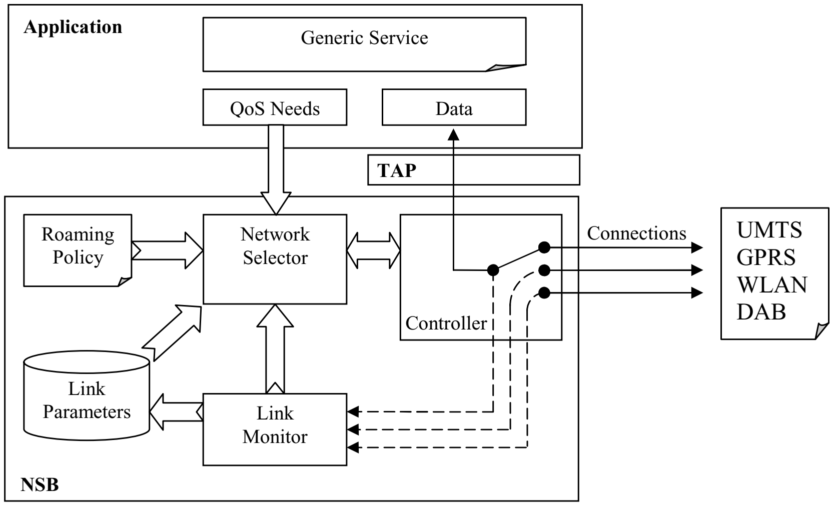

Fig. 3. Building blocks of the NSB

The *Controller* is responsible for sending packets, receiving packets, sending and reacting to control messages, and executing handovers between the different available wireless networks. The Controller on the server side is also responsible for allocating virtual IP addresses to clients and, for each client, it keeps a list with available networks (and their perceived quality) that packets may be sent over. This database is updated by control messages sent from clients.

The *Network Selector* determines the best network to use at the moment. For this purpose, a simple decision algorithm is implemented. The decision algorithm decides for each packet to be sent which network to choose. This decision is based upon recent measurements of link performance in terms of RTT, packet loss and time-outs. This task is carried out by the *Link Monitor.* These values are stored in the *Link Parameter database* on the server and in a separate database on the client. Based on own measurements [7,8], an a-priory network selection is derived [5].

The *Roaming Policy* depends on general scanning and predefined priorities depending on the service selected. The predefined priorities are also based on current status of the connection. The general scanning also depends on distinction of rank, which means choosing the best technology in advance. For example, an a-priory for streaming service is associated with the following list of priorities: WLAN (1), UMTS (2), and GPRS (3). Now let us imagine that there is no WLAN connectivity, and that GPRS experiences a high RTT. In this case the network selector decides that UMTS is currently the best link to send packets over, although UMTS initially had a lower priority than the WLAN network.

3.1 Implementation and Communication Management

Both NSB Client and Server were implemented on desktop computers, running the popular Microsoft Windows XP. C# .NET was used as the programming language.

The NSB uses blocking, synchronous send and receive operations, running in different threads. On the client, one socket is created for each network. All sockets are UDP sockets. UDP is better suited for encapsulation of IP since UDP provides a connectionless transmission medium for IP, thus avoiding nesting one reliability layer into another *i.e.* TCP over TCP that essentially produces a whole level of redundancy.

To assure that packets are sent on the intended interface, a route is created in the routing table for each added network, with the same metric value as the other networks. In this way, Windows will not route packets in an undesirable way.

Prior to sending each packets, the NSB server refers to the tables in the Controler and the `evaluateSend()` method in the Network Selector to find the best network to use. The NSB client uses a similar table and the `evaluateSend()` method. In this way, the NSB can be seen as a multiplexer.

The NSB uses control messages for communication management between NSB client and server. Furthermore, the control messages are also used to monitor network performance. All control messages use the same UDP sockets as the data traffic with the exception of the `APPPREF` message, which uses a separate port. Table 1 lists the control messages.

The two control messages `REGISTER` and `ACKREGISTER` are used for address allocation. The client sends a `REGISTER` request together with information about his first network. The server answers with an `ACKREGISTER` message containing the virtual IP. In this sense the server operates much like a Dynamic Host Configuration Protocol (DHCP) server, although the address lease times are

Table 1. NSB control messages

Message	Direction	Interface	Explanation
`REGISTER`	Client – Server	NSB – NSB	Register with server and get a virtual IP.
`ACKREGISTER`	Server – Client	NSB – NSB	Response to register request.
`ADDNET`	Client – Server	NSB – NSB	A new network is available.
`REMOVENET`	Client – Server	NSB – NSB	A network was lost.
`IAMALIVE`	Client – Server	NSB – NSB	Client is alive and wants to keep his virtual IP.
`BYE`	Client – Server	NSB – NSB	Client logging of, release virtual IP.
`NETSTAT`	Both ways	NSB – NSB	Network statistics.
`ACKNETSTAT`	Both ways	NSB – NSB	Response to `NETSTAT` message.
`APPPREF`	App.	App. – NSB	For controlling NSB settings and QoS feedback.

generally infinite. The client will keep this address until it disconnects with the control message `BYE`.

`ADDNET` and `REMOVENET` are sent by the client when a new network is connected or when a connection is lost. `ADDNET` contains the virtual IP of the client, and information about the new network. The `REMOVENET` message is not that critical, as the server's monitor will discover a network loss anyway within time. It is sent to speed up the server's adoption to the new situation.

`IAMALIVE` is used by the client to hold on to a virtual IP address. The server expects all clients to send this message periodically, otherwise it assumes that the client has crashed without notification and releases the client's virtual IP so that others can use it. When the client NSB shuts down normally, it sends a `BYE` message to tell the server that it does not need its virtual IP anymore.

Applications have that possibility to control the NSB by using the `APPPREF` control message. The message contains information about the requirements of the application in terms of *e.g.* performance and cost. This information is used by the Network Selector to find the most suitable network to use. Applications do not need to control the NSB as the NSB implements automatic network selection. `APPPREF` are also used by the application in order to enquire QoS status of the network connection used at the moment.

Through control messages `NETSTAT` (issued by the sender and includes a time stamp) and `ACKNETSTAT` (issued by the receiver upon reception of a `NETSTAT` message) the application-level RTT is measured. If no `ACKNETSTAT` message has been received for a predefined period of time, it is considered lost. A specific network is classified as unavailable when a predefined number of `ACKNETSTAT` are lost. According to our experience, this value is set to five. However, the value might be further tuned for special network and application scenarios.

In addition, the `NETSTAT` and `ACKNETSTAT` packets contain information about how many packets that were sent by the sender and received at the receiver side in the last time period. In this way, packet loss can be measured.

4 Performance Evalutation of the NSB

The NSB can be viewed as a middleware, therefore its impact on the network performance as perceived by the application must be evaluated. We investigate whether the NSB is transparent from a performance viewpoint or it is a bottleneck that introduces loss and throughput reductions. To this end measurement of streaming traffic is considered. The measurements are produced by using a cooperative UDP tool, consisting of a traffic generator and a receiver. Both generator and receiver have to run simultaneously at each end in order to implement coordinated and comparative measurements. The generator, developed in C#, is trying to send UDP datagrams of constant length as regularly as possible with a sequence number inside each datagram. The datagrams are not sent *back-to-back*, but spaced in order to yield a certain load, hence exact timing on the sender side is important. The minimal time the packets are spaced is hereafter called *inter-packet delay*. The software tries to keep this value as closely as possible. Since the original *C#* time stamp resolution is limited to 10 milliseconds, specific coding was necessary to improve the time stamp resolution to one thousand of a millisecond. This was achieved by using *performance counters* in conjunction with the system time to provide smaller time increments. To this aim the `kernel32.dll` functions `QueryPerformanceCounter` and `QueryPerformanceFrequency` were used. References [9] and [10] discuss the time stamping issue in detail.

4.1 Measurement Setup

For the measurements, two scenarios were considered: the first is called the *downlink scenario* (*cf.* Figure 4), in which the client is connected to a base station (BS) and the server to the Internet via 100 Mbps Ethernet. The second one is called the *uplink scenario* (*cf.* Figure 4), in which the server is connected to a BS and the client to the Internet via 100 Mbps Ethernet. The scenario names are based on the direction of the data traffic in reference to the BS. The encapsulation used by the TAP device, *cf.* Figure 2, leads to an overhead of 42

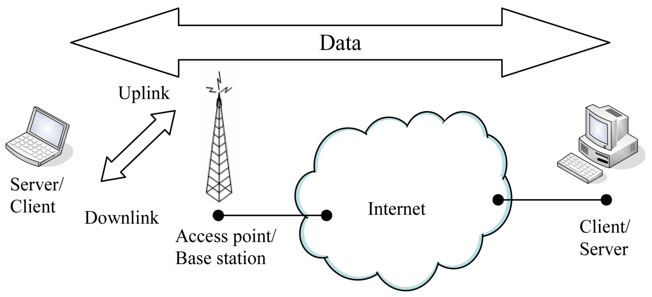

Fig. 4. Uplink and downlink scenario

bytes that potentially can lead to undesirable fragmentation. Hence, the tool takes as a parameter the desired packet length in order to yield comparable load on the link.

4.2 Measurements and Results

The first case presented is a modestly loaded UMTS uplink measurement where the NSB is not used. By choosing a payload size of 158 bytes and a nominal inter-packet delay of 100 ms, the server transmission rate was set to 12.64 kbps which is matched by the averaged throughput at the application for both server and client. The plots from the time domain are displayed in Figure 5. Jitter at the senders sleep function is hardly visible, *cf.* Figure 5 (top). Figure 5 (second from top) shows very small jitter in the senders send function, three orders of magnitude smaller as compared to the jitter at senders sleep function. The inter-packet delay at the client side, *cf.* Figure 5 (bottom), displays distinct burst deviations that seem to originate from the UMTS channel itself. The client does not indicate any packet loss.

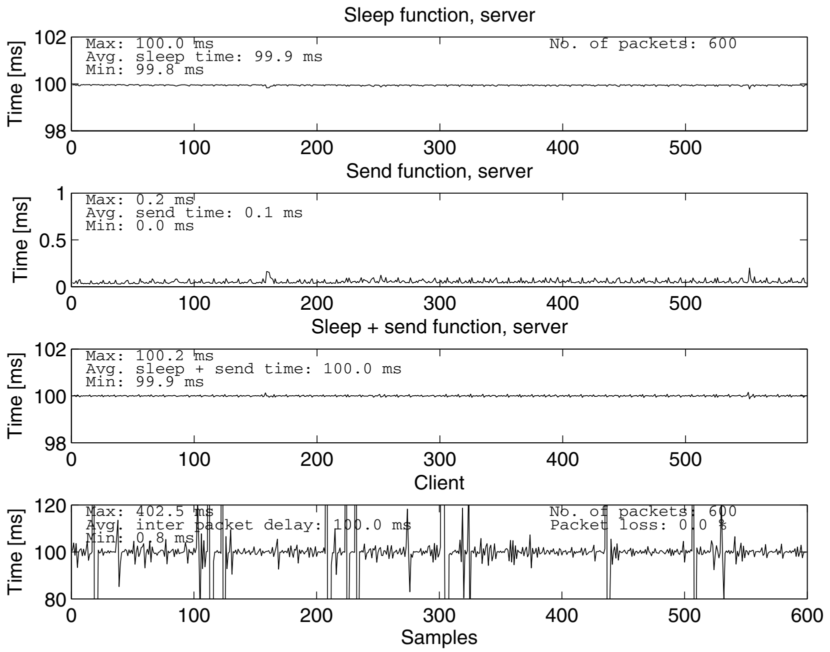

Fig. 5. Timeplot UMTS uplink without NSB and with 100 ms inter-packet delay

In the second case, the NSB is activated and compensation for the introduced overhead is done by decreasing the payload by the size of the overhead, *i.e.* 42 bytes. Thus, the payload is set to 116 bytes and the frame size on the link layer is kept the same, *i.e.* 200 bytes. The nominal inter-packet delay is also maintained at 100 ms. Figure 6 indicates a rather small jitter at the client side. Distinct burst deviations are identified. As in the previous case, these bursts seem to originate from the UMTS channel itself, rather than from the NSB. Still the client does not indicate any packet loss. Table 3 summarizes the statistical results from all measurements, including the uplink measurements.

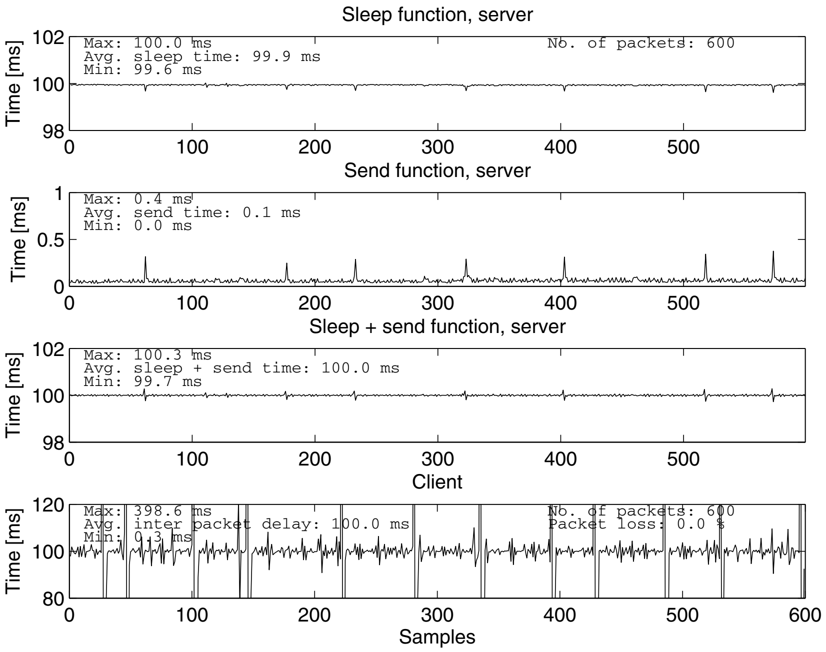

Fig. 6. Timeplot UMTS uplink with NSB and 100 ms inter-packet delay

The UMTS downlink case without NSB is depicted in Figure 7. Here the payload size is set to 1158 bytes and a nominal inter-packet delay to 100 ms. The server transmission rate was set to 92.64 kbps, which is matched by the averaged throughput at the application for both server and client. Figure 7 shows that the mean inter-packet delay at the client side is slightly smaller than the mean inter-packet delay at the server, displaying distinct burst deviations. The client does not experience any packet loss. Figure 8, shows the UMTS downlink case when the NSB is activated. The payload is set 1116 bytes in order to compensate for the overhead introduced by the NSB. The frame size of the link layer amounts to 1200 bytes. The nominal inter-packet delay is also maintained at 100 ms. No packet loss is perceived by the client. However, Figure 8 (bottom) displays several distinct burst deviations indicating that more jitter is perceived by the client. Table 3 summarizes the statistical results of the downlink measurements.

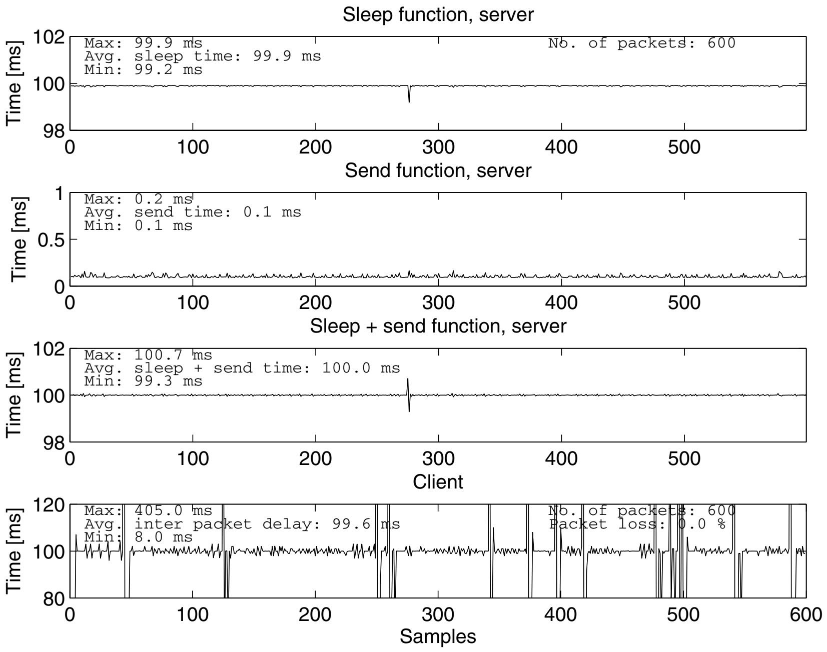

Fig. 7. Timeplot UMTS downlink without NSB and 100 ms inter-packet delay

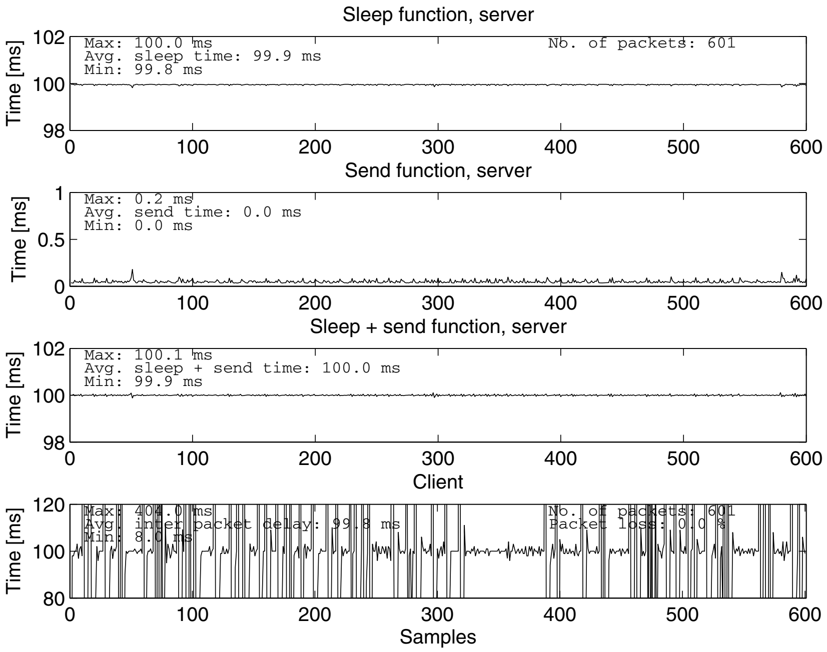

Fig. 8. Timeplot UMTS downlink with NSB and 100 ms inter-packet delay

The next case is a stress test of the NSB. To this aim, the UMTS network is replaced with a Local Area Network (LAN) 10 Mbps link. The payload size is set to 1458 bytes with no NSB used and 1416 bytes with NSB activated. Thus, the frame size at the link layer is kept at 1500 bytes. When the NSB is not used, the jitter perceived by the client is rather high together with some moderate amount of packet loss, *cf.* Figure 9. On the other hand, when NSB is activated, the client still perceives rather high jitter while there is no packet loss, *cf.* Figure 10. Table 3 summarizes the statistical results from all measurements.

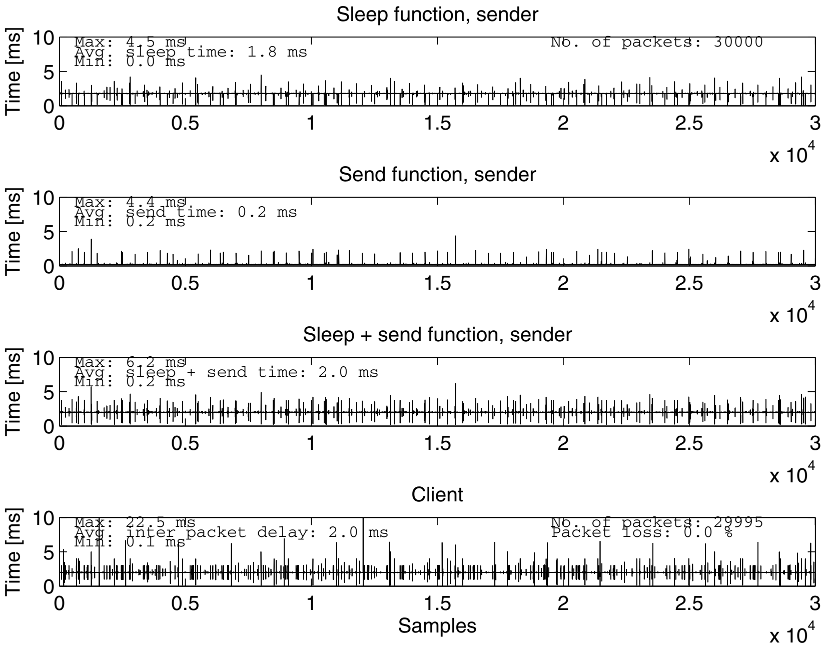

Fig. 9. Timeplot LAN without NSB and 2 ms inter-packet delay

4.3 Handover Delays

Handover delays between WLAN and UMTS were investigated. Two scenarios were considered: the first addresses handover without backup network, in which the client first has to detect the network loss, then make a decision regarding which network to connect to, and finally establish the connection. This means that when connecting to the UMTS network, the time for dialling is included in the delay. In the second scenario, handover with a backup network is considered. In this scenario the UMTS network is already connected, thus the delay does not include time for dialling although it includes the time to detect loss of network connection. Network loss can be detected either by the loss of `ACKNETSTAT` control messages or through the socket API reporting failed a send. Hence, choosing

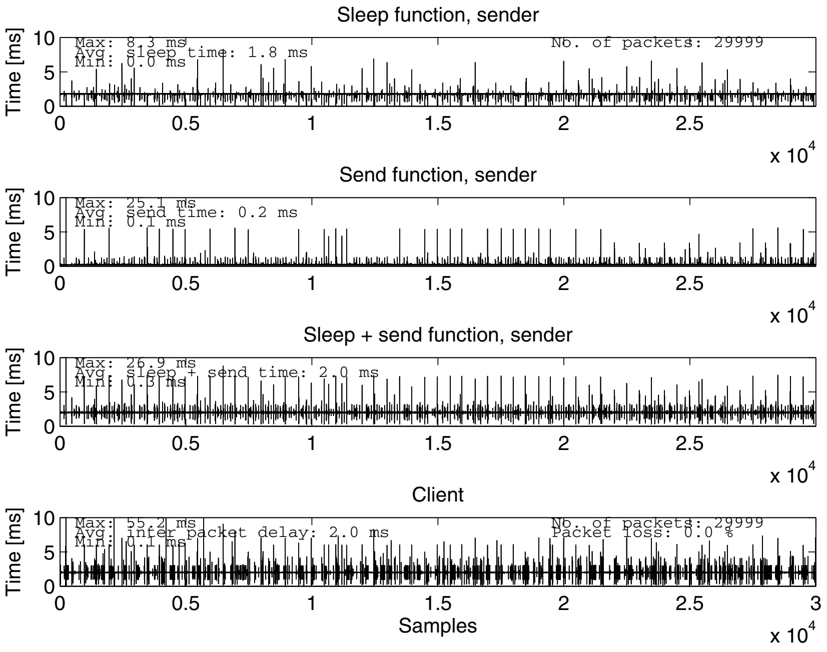

Fig. 10. Timeplot LAN with NSB and 2 ms inter-packet delay

a different parameter value for the number of missed `ACKNETSTAT` might reduce the time for detecting a network loss. Network loss can also be detected through the Windows Management Instrumentation (WMI). This method was used in the early stage of the NSB development but was abandoned due to severe memory leakage in the WMI that was not resolved by Microsoft.

From Table 2 it can be observed that the handover delay when a backup UMTS network exists is considerably shorter as compared to the case when no backup network exists. The longer delay partly arises from the additional time it take to connect to the UMTS, *i.e.* the time it takes for the modem to dial the UMTS connection.

The handover delay can be further reduced by decreasing the times for detecting network loss, which can be achieved by choosing different parameter values for the number of missing `ACKNETSTAT` control message.

Table 2. NSB handover delay between WLAN and UMTS networks

	min [s]	mean [s]	max [s]
without backup network	6.98	9.13	10.26
with backup network	2.18	3.10	4.00

4.4 Relative Overhead vs. Frame Size Ration

The encapsulation used by the TAP device, *cf.* Figure 2, implies an overhead of 42 bytes. This has a minor impact on the application-perceived throughput hence the MTU of application is reduced in order to avoid fragmentation on the link layer. The overhead also results in larger frame size on the link layer. On the other hand, large frames at the link layer have a positive effect on the efficiency.

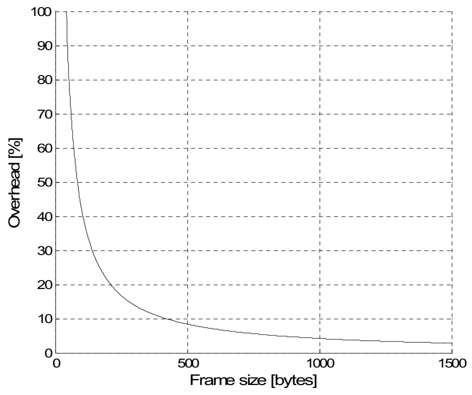

Fig. 11. Relative overhead vs. frame size ration

Table 3. Overview of the NSB performance assessment

		UMTS uplink IPD 100 ms		UMTS downlink IPD 100 ms		LAN IPD 2 ms	
		excl. NSB	incl. NSB	excl. NSB	incl. NSB	excl. NSB	incl. NSB
sleep function, server [ms]	min	98.8	99.6	99.2	99.8	0.0	0.00
	mean	99.9	99.9	99.9	99.9	1.8	1.8
	max	100.0	100.0	99.9	100.0	4.5	8.3
	stddev	0.0	0.0	0.0	0.0	0.1	0.2
send function, server [ms]	min	0.0	0.0	0.1	0.0	0.2	0.1
	mean	0.1	0.1	0.1	0.0	0.2	0.2
	max	0.2	0.4	0.2	0.2	4.4	25.1
	stddev	0.0	0.0	0.0	0.0	0.1	0.3
sleep + send function, server [ms]	min	98.9	99.7	99.3	99.9	0.2	0.3
	mean	100.0	100.0	100.0	100.0	2.0	2.0
	max	100.2	100.3	100.7	100.1	6.2	26.9
	stddev	0.0	0.0	0.0	0.0	0.2	0.3
client [ms]	min	0.8	0.3	8.0	8.0	0.0	0.0
	mean	100.0	100.0	99.6	99.8	2.00	2.0
	max	402.5	398.6	405.0	404.0	22.5	55.2
	stddev	32.1	37.1	34.9	82.2	0.3	0.5

In Figure 11 the ratio between the overhead and frame size is plotted. When using frames larger than 450 byte the overhead is less than 10 % of the frame size.

Table 3 summarizes the statistical results from the measurements. We recognize that the client perceives more jitter than the server. This behaviour has also been observed in [7,8,5]. However, the jitter seems to originate from the physical layer rather than from the NSB. In fact even when considering different load scenarios, we have not seen any considerable differences between using the NSB or not doing so.

5 Conclusion and Outlook

We have described and evaluated a ready-to-deploy mobility framework that supports seamless communication and represents an important enabler for adaptive applications through a simple QoS feedback mechanism. The framework is called NSB, and ideally has several networks to select from *e.g.* GPRS, UMTS, WLAN, *etc.* The design of the NSB has been described in detail, where the network selection is based on measured network performance.

Performance evaluation of the framework indicates that the NSB is transparent to the upper layers, in terms of throughput, although rather small jitter at the receiver has been identified. The fact that no loss occurred during the measurements is partly due to that the server uses a blocking send function. Hence, in case the server transmits datagrams too fast, the send function itself holds packets until they can be sent, which can be considered as some kind of force feed-back. No performance implication was found besides the tunnel-typical overhead of 42 bytes.

Furthermore, measurements of handover delays between WLAN and UMTS networks indicate that rather short handover delays can be achieved when a backup network such as UMTS was available. The handover delays are highly dependent on how fast a network loss can be detected and whether there exists backup network connectivity.

Future work includes a refined roaming strategy that will take advanced performance monitoring into account and faster handover by optimizing the corresponding parameter setting. Also additional measurements of one-way delays and TCP goodput are considered.

References

1. C. Perkins. IP Mobility Support for IPv4. Technical Report IETF RFC 3344, August 2002.
2. L. Morand and S. Tessier. Global Mobility Approach with Mobile IP in All IP Networks. In *IEEE International Conf. on Communications (ICC)*, pages 2075–2079, May 2002.
3. J-.W. Lin and J. Arul. An efficient fault-tolerant approach for Mobile IP in wireless systems. In *IEEE Trans. on Mobile Computing*, volume 2, pages 207–220, July-Sept. 2003.

4. J.W. Floroiu, R. Ruppelt, D. Sisalem, and J. Voglimacci. Seamless handover in terrestrial radio access networks: a case study. *IEEE Communications Magazine*, 41(11):110–116, Nov. 2003.
5. M. Fiedler, L. Isaksson, S. Chevul, P. Lindberg, and J. Karlsson. Measurements and Analysis of Application-Perceived Throughput via Mobile Links. In *Proceedings of the 2005* 3^{rd} *Performance Modeling and Evaluation of Heterogeneous Networks (HET-NETs)*, July 2005.
6. OpenVPN. URL: http://openvpn.net/.
7. S. Chevul, J. Karlsson, L. Isaksson, M. Fiedler, P. Lindberg, and L. Strandén. Measurements of application-perceived throughput in DAB, GPRS, UMTS and WLAN Environments. In *Proceedings of RVK'05*, Linköping, Sweden, June 2005.
8. L. Isaksson, S. Chevul, M. Fiedler, J. Karlsson, and P. Lindberg. Application-Perceived Throughput Process in Wireless Systems. In *Proceedings of ICMCS'05*, Montreal, Canada, August 2005.
9. S. Chevul. *On Application-Perceived Quality of Service in Wireless Networks.* Licentiate Dissertation Series No. 2006:11. Blekinge Institute of Technology, 2006.
10. S. Chevul, L. Isaksson, M. Fiedler, and P. Lindberg. Measurement of Application-Perceived Throughput of an E2E VPN Connection Using a GPRS Network. In *Wireless Systems and Network Architectures in Next Generation Internet, Second International Workshop of the EURO-NGI Network of Excellence*, pages 255–268, Villa Vigoni, Italy, 2005.