Ipv6 Web Services Experiment-HOWTO

Lead Editor: Patrick Mandic, USTUTT

25/02/2005

Status: Final

SIXTH FRAMEWORK PROGRAMME

PRIORITY IST-2002-2.3.1.18

Grid for complex problem solving

Proposal/Contract no.: 004293

License

THE WORK (AS DEFINED BELOW) IS PROVIDED UNDER THE TERMS OF THIS CREATIVE COMMONS PUBLIC LICENSE ("CCPL" OR "LICENSE"). THE WORK IS PROTECTED BY COPYRIGHT AND/OR OTHER APPLICABLE LAW. ANY USE OF THE WORK OTHER THAN AS AUTHORIZED UNDER THIS LICENSE OR COPYRIGHT LAW IS PROHIBITED.

BY EXERCISING ANY RIGHTS TO THE WORK PROVIDED HERE, YOU ACCEPT AND AGREE TO BE BOUND BY THE TERMS OF THIS LICENSE. THE LICENSOR GRANTS YOU THE RIGHTS CONTAINED HERE IN CONSIDERATION OF YOUR ACCEPTANCE OF SUCH TERMS AND CONDITIONS.

1. Definitions

a. **"Collective Work"** means a work, such as a periodical issue, anthology or encyclopaedia, in which the Work in its entirety in unmodified form, along with a number of other contributions, constituting separate and independent works in themselves, are assembled into a collective whole. A work that constitutes a Collective Work will not be considered a Derivative Work (as defined below) for the purposes of this License.
b. **"Derivative Work"** means a work based upon the Work or upon the Work and other pre-existing works, such as a translation, musical arrangement, dramatization, fictionalization, motion picture version, sound recording, art reproduction, abridgment, condensation, or any other form in which the Work may be recast, transformed, or adapted, except that a work that constitutes a Collective Work will not be considered a Derivative Work for the purpose of this License. For the avoidance of doubt, where the Work is a musical composition or sound recording, the synchronization of the Work in timed-relation with a moving image ("synching") will be considered a Derivative Work for the purpose of this License.
c. **"Licensor"** means the individual or entity that offers the Work under the terms of this License.
d. **"Original Author"** means the individual or entity who created the Work.
e. **"Work"** means the copyrightable work of authorship offered under the terms of this License.
f. **"You"** means an individual or entity exercising rights under this License who has not previously violated the terms of this License with respect to the Work, or who has received express permission from the Licensor to exercise rights under this License despite a previous violation.

2. Fair Use Rights. Nothing in this license is intended to reduce, limit, or restrict any rights arising from fair use, first sale or other limitations on the exclusive rights of the copyright owner under copyright law or other applicable laws.

3. License Grant. Subject to the terms and conditions of this License, Licensor hereby grants You a worldwide, royalty-free, non-exclusive, perpetual (for the duration of the applicable copyright) license to exercise the rights in the Work as stated below:

a. to reproduce the Work, to incorporate the Work into one or more Collective Works, and to reproduce the Work as incorporated in the Collective Works;

b. to distribute copies or phonorecords of, display publicly, perform publicly, and perform publicly by means of a digital audio transmission the Work including as incorporated in Collective Works.
c. For the avoidance of doubt, where the work is a musical composition:
    i. **Performance Royalties Under Blanket Licenses**. Licensor waives the exclusive right to collect, whether individually or via a performance rights society (e.g. ASCAP, BMI, SESAC), royalties for the public performance or public digital performance (e.g. webcast) of the Work.
    ii. **Mechanical Rights and Statutory Royalties**. Licensor waives the exclusive right to collect, whether individually or via a music rights society or designated agent (e.g. Harry Fox Agency), royalties for any phonorecord You create from the Work ("cover version") and distribute, subject to the compulsory license created by 17 USC Section 115 of the US Copyright Act (or the equivalent in other jurisdictions).
d. **Webcasting Rights and Statutory Royalties**. For the avoidance of doubt, where the Work is a sound recording, Licensor waives the exclusive right to collect, whether individually or via a performance-rights society (e.g. SoundExchange), royalties for the public digital performance (e.g. webcast) of the Work, subject to the compulsory license created by 17 USC Section 114 of the US Copyright Act (or the equivalent in other jurisdictions).

The above rights may be exercised in all media and formats whether now known or hereafter devised. The above rights include the right to make such modifications as are technically necessary to exercise the rights in other media and formats, but otherwise you have no rights to make Derivative Works. All rights not expressly granted by Licensor are hereby reserved.

4. Restrictions. The license granted in Section 3 above is expressly made subject to and limited by the following restrictions:

a. You may distribute, publicly display, publicly perform, or publicly digitally perform the Work only under the terms of this License, and You must include a copy of, or the Uniform Resource Identifier for, this License with every copy or phonorecord of the Work You distribute, publicly display, publicly perform, or publicly digitally perform. You may not offer or impose any terms on the Work that alter or restrict the terms of this License or the recipients' exercise of the rights granted hereunder. You may not sublicense the Work. You must keep intact all notices that refer to this License and to the disclaimer of warranties. You may not distribute, publicly display, publicly perform, or publicly digitally perform the Work with any technological measures that control access or use of the Work in a manner inconsistent with the terms of this License Agreement. The above applies to the Work as incorporated in a Collective Work, but this does not require the Collective Work apart from the Work itself to be made subject to the terms of this License. If You create a Collective Work, upon notice from any Licensor You must, to the extent practicable, remove from the Collective Work any reference to such Licensor or the Original Author, as requested.
b. If you distribute, publicly display, publicly perform, or publicly digitally perform the Work or Collective Works, You must keep intact all copyright notices for the Work and give the Original Author credit reasonable to the medium or means You are utilizing by conveying the name (or pseudonym if applicable) of the Original Author if supplied; the title of the Work if supplied; and to the extent reasonably practicable, the Uniform Resource Identifier, if any, that Licensor specifies to be associated with the Work, unless such URI does not refer to the copyright notice or licensing information for the Work. Such credit may be implemented in any reasonable manner; provided, however, that in the

case of a Collective Work, at a minimum such credit will appear where any other comparable authorship credit appears and in a manner at least as prominent as such other comparable authorship credit.

5. Representations, Warranties and Disclaimer

UNLESS OTHERWISE MUTUALLY AGREED TO BY THE PARTIES IN WRITING, LICENSOR OFFERS THE WORK AS-IS AND MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND CONCERNING THE MATERIALS, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, INCLUDING, WITHOUT LIMITATION, WARRANTIES OF TITLE, MERCHANTIBILITY, FITNESS FOR A PARTICULAR PURPOSE, NONINFRINGEMENT, OR THE ABSENCE OF LATENT OR OTHER DEFECTS, ACCURACY, OR THE PRESENCE OF ABSENCE OF ERRORS, WHETHER OR NOT DISCOVERABLE. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO SUCH EXCLUSION MAY NOT APPLY TO YOU.

6. Limitation on Liability. EXCEPT TO THE EXTENT REQUIRED BY APPLICABLE LAW, IN NO EVENT WILL LICENSOR BE LIABLE TO YOU ON ANY LEGAL THEORY FOR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL, PUNITIVE OR EXEMPLARY DAMAGES ARISING OUT OF THIS LICENSE OR THE USE OF THE WORK, EVEN IF LICENSOR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

7. Termination

a. This License and the rights granted hereunder will terminate automatically upon any breach by You of the terms of this License. Individuals or entities who have received Collective Works from You under this License, however, will not have their licenses terminated provided such individuals or entities remain in full compliance with those licenses. Sections 1, 2, 5, 6, 7, and 8 will survive any termination of this License.
b. Subject to the above terms and conditions, the license granted here is perpetual (for the duration of the applicable copyright in the Work). Notwithstanding the above, Licensor reserves the right to release the Work under different license terms or to stop distributing the Work at any time; provided, however that any such election will not serve to withdraw this License (or any other license that has been, or is required to be, granted under the terms of this License), and this License will continue in full force and effect unless terminated as stated above.

8. Miscellaneous

a. Each time You distribute or publicly digitally perform the Work, the Licensor offers to the recipient a license to the Work on the same terms and conditions as the license granted to You under this License.
b. If any provision of this License is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this License, and without further action by the parties to this agreement, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.
c. No term or provision of this License shall be deemed waived and no breach consented to unless such waiver or consent shall be in writing and signed by the party to be charged with such waiver or consent.
d. This License constitutes the entire agreement between the parties with respect to the Work licensed here. There are no understandings, agreements or representations with respect to

the Work not specified here. Licensor shall not be bound by any additional provisions that may appear in any communication from You. This License may not be modified without the mutual written agreement of the Licensor and You.

Document History

Contributors: Patrick Mandic (USTUTT-RUS), Ignaz Müller (USTUTT-HLRS), Robert Piotter (USTUTT-HLRS)

Version	Date	Authors	Sections Affected
0.1	21.12.2004	Patrick Mandic	Initial version
0.2	11.01.2005	Robert Piotter	Added installation information
0.3	13.01.2005	Patrick Mandic	Added Mobile IPv6 configuration
0.4	27.01.2005	Patrick Mandic	Added DNS Server
0.5	02.02.2005	Patrick Mandic	Added IPSec
0.6	24.02.2005	Patrick Mandic, Robert Piotter, Ignaz Müller	Tests description. Table of compatibilities.

Table of Contents

List of Figures

List of Tables

1. Introduction

This document describes the Akogrimo "hello world"-testbed developed by the USTUTT. This testbed has the objective of analyzing the compatibility between WS tools in IPv4, IPv6 and MIPv6 combined environments. In Section 2 we make a description of the testbed and what we want to achieve with it in general terms. Section 3 describes how to install the basic software to provide each node with the necessary basic WS and network infrastructure and how to configure the network to give connectivity between nodes based on IPv6, MIPv6 and IPv4. On Section 4 we describe the concrete applications and tests that we developed and carried out. Configuration files and code files are made available on request.

This experiment will be further extended in the near future in order to experiment with WSRF.NET and Globus Toolkit 4 and also with a SIP based context manager.

2. Overall description

The basic idea of this testbed is to have a configuration like the one that follows:

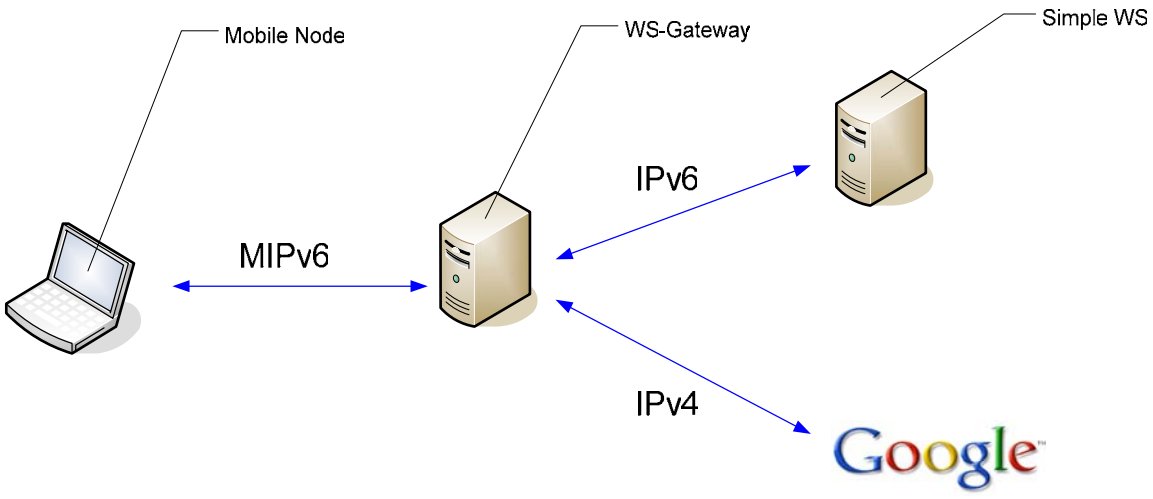


Figure 1: Testbed scheme

The Mobile Node (MN) accesses the WS-Gateway with WSs by making use of the MIPv6 protocol. The WS-Gateway Implements a simple WS in order to test its performance by having it communicating with another simple WS and Google using respectively IPv6 and IPv4.

The implementation of this test-bed regarding the network architecture and the operating systems used for each node can be observed in the following picture:

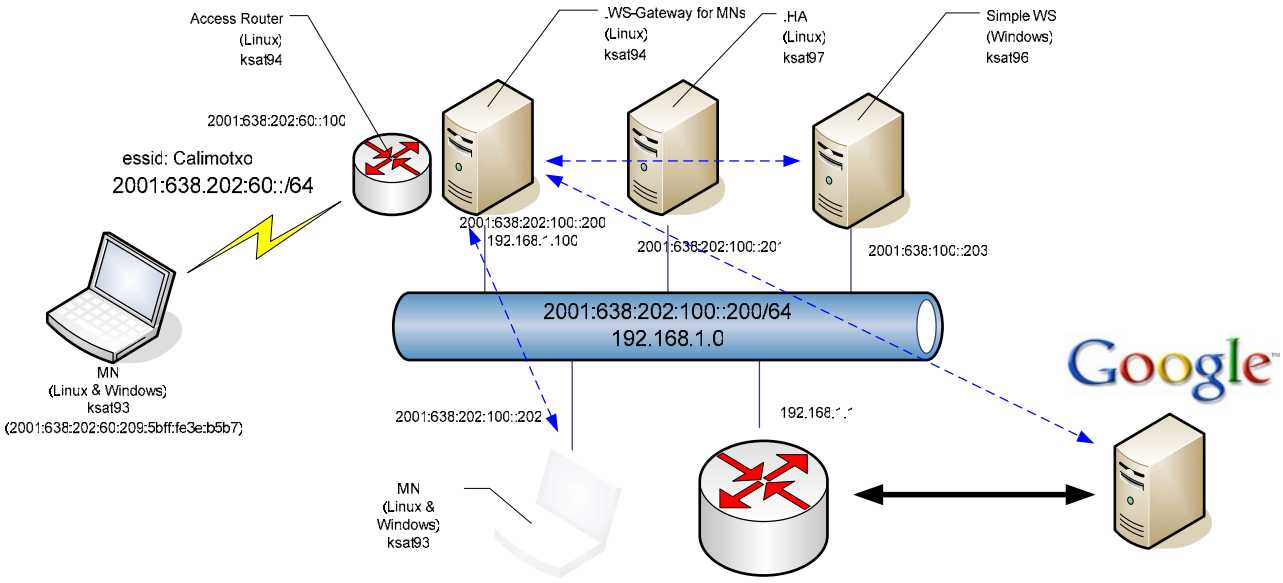


Figure 2: complete testbed Map

The MN:

As a MN we are using a PDA and a PC that has been configured with dual boot in order to test the performance of the WSs running over MIPv6 in different different architectures, Windows XP, Pocket PC 2003 and Linux (In the annex D you can see a complete drawing with the PDA). Notice that the MN has two addresses depending on whether he is at home or roaming -which are called Home Address (HoA) when the node is at home and Care-of address (CoA) when the node is roaming. This, however, is only contemplated at network layer and is not visible for the rest of the layers. That means that **the only address that applications like WSs will need to handle and be aware of is the HoA address**, that is, the address of the mobile node in his home network (2001:638:202:100::202). No matter if the MN is in his Home network or roaming, the applications will always use the HoA to communicate with him, thus not being aware of its location. The MN is drawn as a shadowed figure in its home network representing that, even though his physical location is somewhere else, the MN is virtually always at home (from the application point of view). The schema in *Figure 1* is represented in *Figure 2* with blue arrows.

The Home Agent (HA):

The HA is part of the MIPv6 structure. Its task is to collect the packets sent to the MN while this is roaming and send them by means of a tunnel to the MN's current Care-of Address.

The Access Router (AR)

The Access Router (AR) – which is being used by the MN - is co-located on ksat94 together with the WS-Gateway. This was done simply to reduce the number of machines in the test-bed and therefore has no other conceptual interest. One interface of the AR looks at the 2001:638:202:60:: network where the MN is and the other at the 2001:638:202:100:: network.

The WS-Gateway:

The WS-Gateway gets requests from the MN and respectively connects to google WSs and to the simple Windows WSs using IPv4 and IPv6. Notice that the WS-Gateway communicates with the MN – from the application point of view – using the MN's address 2001:638:202:100::202.

Regarding the creation and management of the network a complete set of addresses in both protocols (IPv6 and IPv4) was developed to ease things like software installation from Internet, web navigation and so forth by using IPv4.

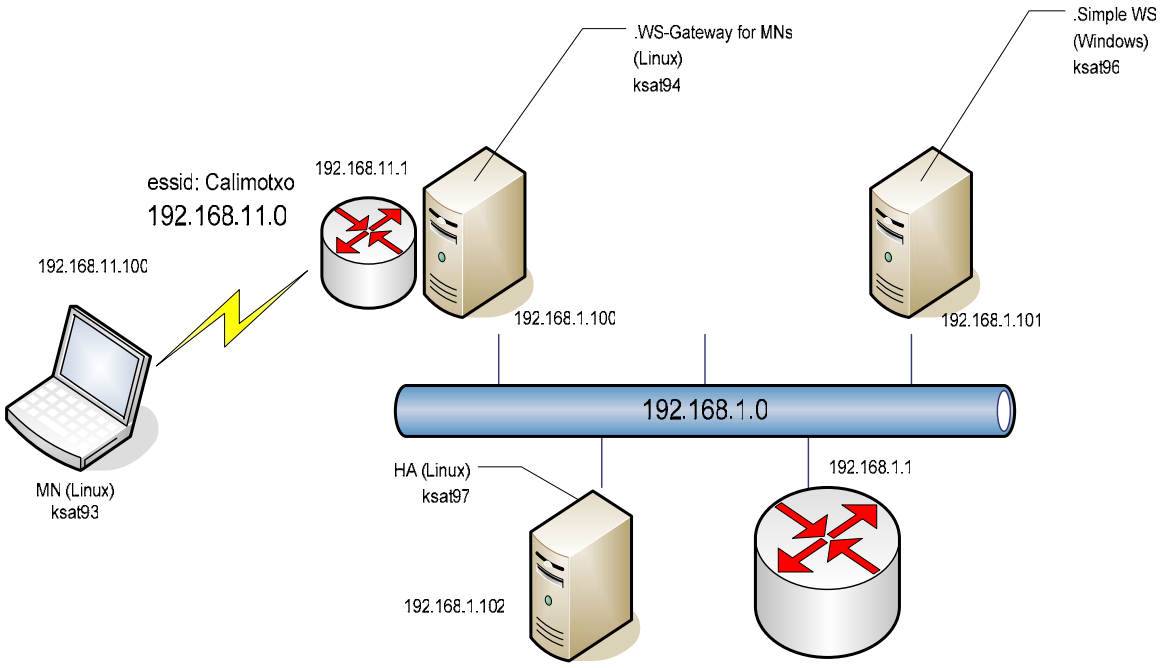


Figure 2: IPv4 Testbed Map

This other drawing represents the map of IPv6 addresses.

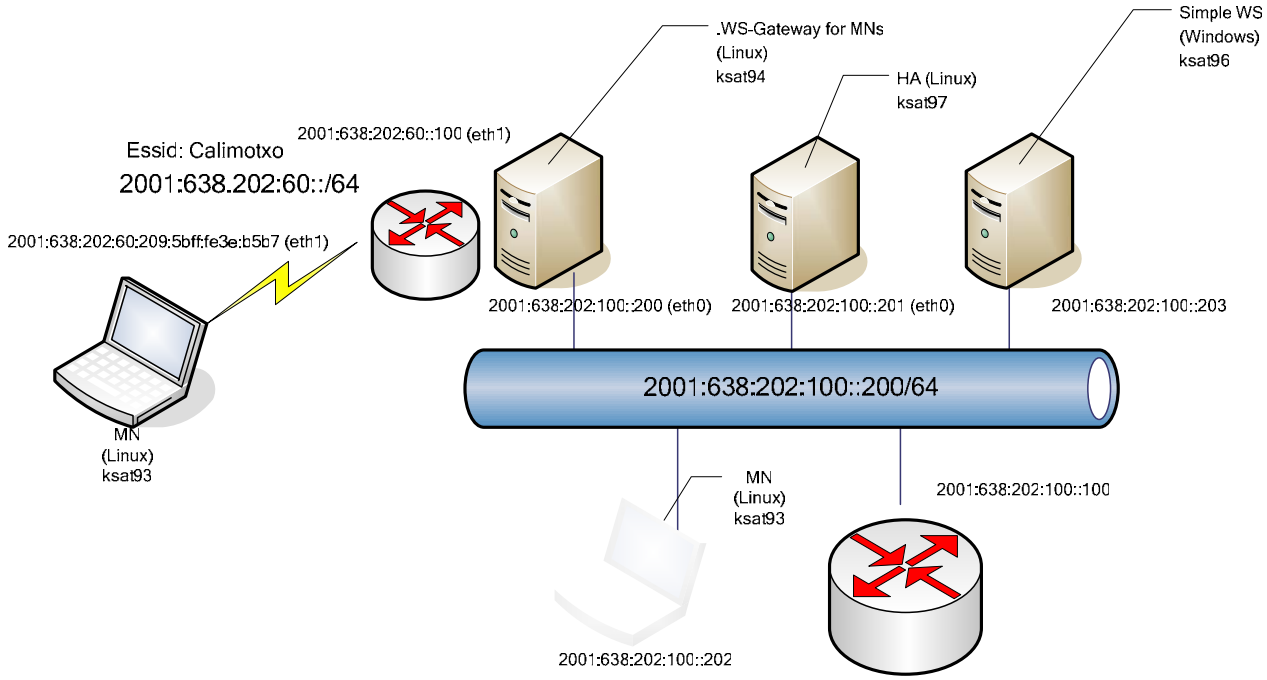


Figure 3: IPv6 testbed Map

3. Installing

3.1. MN Linux

Name	Ksat93
OS	Mandrake 10.0 Linux
Kernel	Linux kernel 2.6.8.1
Java	Java 1.5.0_01
Openssl	Libopenssl0.9.7-devel-0.9.7c Openssl-0.9.7 (for IPSec)
SOAP	Axis1.1
XML Parser	Xerces 2.6.2
Mobile IPv6	MIPL-2.0
Wireless	wirelesstools-26

3.1.1. Installation procedure

I suggest to read the IPv6 Linux Howto if you are not familiar with IPv6 (http://www.tldp.org/HOWTO/Linux+IPv6-HOWTO/)

3.1.1.1. Install OS

Install Mandrake (10.0).

3.1.1.2. Install kernel with mipl support.

1) Install kernel

- Download kernel to */usr/src* (http://www.kernel.org/pub/linux/kernel/v2.6/linux-2.6.8.1.tar.bz2)

- Download mipv6-2.0-rc1-linux-2.6.8.1.patch.gz

- Patch the kernel: copy the patch in the kernel directory and run

```
# patch -p1 --dry-run < mipv6-2.0-rc1-linux-2.6.8.1.patch
```

If no error arises, then run

```
# patch -p1 < mipv6-2.0-rc1-linux-2.6.8.1.patch
```

```
# make menuconfig
```

In the bscw you can find my complete .config file with the options I used. The .config has to have this options:

```
CONFIG_EXPERIMENTAL=y
    CONFIG_SYSVIPC=y
    CONFIG_PROC_FS=y
    CONFIG_NET=y
    CONFIG_INET=y
    CONFIG_IPV6=y
    CONFIG_IPV6_MIP6=y
    CONFIG_XFRM=y
    CONFIG_XFRM_USER=y
    CONFIG_XFRM_ENHANCEMENT=y

 #The Home Agent and Mobile Node also need:

    CONFIG_IPV6_TUNNEL=y
    CONFIG_IPV6_ADVANCED_ROUTER=y
    CONFIG_IPV6_MULTIPLE_TABLES=y

 #The Mobile Node also needs:

    CONFIG_IPV6_SUBTREES=y

 # Warning!  You shouldn't turn on MIPv6 debug messages, because a huge
 #amount messages will be generated, but you can do it by defining:

    CONFIG_IPV6_MIP6_DEBUG=y

 #For IPsec support you need at least:

    CONFIG_INET6_ESP=y

  #If you plan to use KAME IPsec tools you also need:
```

```
CONFIG_NET_KEY=y
```

The mipl package comes with a simple script that you can use to check this configuration:

```
# ./chkconf_kernel.sh /usr/src/linux-2.6.8.1
```

Now we are ready to compile the kernel.

```
# make
# make modules_install
# make install
# lilo –v
```

2) Install MIPL user space tools:

- Download http://www.mipl.mediapoli.com/software/download/mipv6-2.0-rc1.tar.gz
- (Install openssl if not done yet: libopenssl0.9.7-devel-0.9.7 and openssl-0.9.7)
- Specify the kernel headers to include:

```
# export CPPFLAGS=-I/usr/src/linux-2.6.8.1/include ./configure
```

- And compile

```
#  ./configure
# make
# make install
```

An executable should have appeared in the sbin folder */usr/local/sbin/mip6d*

3.1.1.3. Installing Java:

- http://java.sun.com/j2se/1.5.0/download.jsp. Click on download SDK for other platforms and then choose the Linux RPM self-extracting file.

- run the downloaded file and install the resulting rpm.

```
# ./j2sdk-1_5_0_01-linux-i586-rpm.bin
# urpmi j2sdk-1_5_0_01-linux-i586.rpm
```

- Add these lines to the *$HOME/.bashrc* file:

```
PATH="$PATH:/usr/java/j2sdk1.5.0_01/bin/:."
JAVA_HOME=/usr/java/j2sdk1.5.0_01/

export PATH
export JAVA_HOME
```

- In case you want java to be used by any user you can add the lines to the */etc/bashrc* file.

3.1.1.4. Installing Axis 1.1

These are simple libraries that just need to be uncompressed and set in the CLASSPATH

Download the package:

```
# wget http://apache.autinity.de/ws/axis/1_1/axis-1_1.tar.gz
# tar zxvvf axis-1_1.tar.gz –C /usr/local/lib/
```

Add the CLASSPATH in the .bashrc file:

```
AXIS_HOME=/usr/local/lib/axis-1_1/
export CLASSPATH=$AXIS_HOME/lib/axis.jar/:$AXIS_HOME/lib/jaxrpc.jar/:$AXIS_HOME/lib/saaj.jar/:$AXI
logging.jar/:$AXIS_HOME/lib/commons-discovery.jar/:$AXIS_HOME/lib/wsdl4j.jar
```

3.1.1.5. Installing Xerces 2.6.2

These are simple libraries that just need to be uncompressed and set in the CLASSPATH

Download the package:

wget http://archive.apache.org/dist/xml/xerces-j/Xerces-J-bin.2.6.2.tar.gz

tar zxvvf Xerces-J-bin.2.6.2.tar.gz –C /usr/local/lib/

Add the libraries to the CLASSPATH in the .bashrc file:

```
XERCES_HOME=/usr/local/lib/xerces-2_6_2/
export CLASSPATH=$CLASSPATH:$XERCES_HOME….
```

3.1.1.6. Configure network:

Install wirelesstools

```
# urpmi wirelesstools
```

Configure wireless interface:

```
# iwconfig eth1 essid Calimotxo mode Ad-hoc
```

Edit */etc/sysconfig/*network

```
HOSTNAME=ksat93
NETWORKING=yes
GATEWAY=192.168.11.1
```

```
GATEWAYDEV=eth1

NETWORKING_IPV6=yes
IPV6_AUTOCONF=yes
```

Edit */etc/sysconfig/network-scripts/ifcfg-ethX*; where X is the interface number of your network card. In this case was eth0

```
DEVICE=eth0
BOOTPROTO=none
IPADDR=192.168.11.100
NETMASK=255.255.255.0
ONBOOT=yes

IPV6INIT="yes"
```

For some reason adding the wireless extension configurations to these files didn't seem to work for us with Mandrake but you can try it by adding the following lines:

```
WIRELESS_ESSID=Calimotxo
WIRELESS_MODE=Ad-hoc
```

For mobile IPv6 we need to edit */usr/local/etc/mip6d.conf*

```
# This is an example of mip6d Mobile Node configuration file
NodeConfig = MN;


## If set to > 0, will not detach from tty
DebugLevel = 10;


## Support route optimization with other MNs
DoRouteOptimizationCN = enabled;


## Use route optimization with CNs
DoRouteOptimizationMN = enabled;


UseMnHaIPsec = disabled;


UseCnBuAck = enabled;


MnHomeLink = {
      LinkName = "eth1";
```

```
        HomeAddress =      2001:638:202:100::202/64;
        HomeAgentAddress = 2001:638:202:100::201;

        MnRoPolicy = {
                Proto = ICMP;
                DoRouteOptimization = enabled;
        };
};
```

Restart the network:

```
# service network restart
```

Start MIPL:

```
# mipd6d
```

Caveat: It seems that mipd6d gives problems if you don't start it first in the MN and then in the AR.

- DNS parameters:

/etc/resolve.conf

```
search akogrimo6.rus.uni-stuttgart.de
nameserver 2001:638:202:100::200
```

3.2. Windows XP MN

Name	Ksat96
OS	Windows XP SP1 (MIPv6 is not supported in SP2)
Java	Java 5.0
MIPv6	MS MIPv6 Tech Prev
SOAP	Axis1.1
.NET	.NET Framework 1.1
XML Parser	Xerces 2.6.2

3.2.1. Installation

3.2.1.1. Java

http://java.sun.com/j2se/1.5.0/download.jsp

3.2.1.2. Axis1.1

Download axis http://apache.autinity.de/ws/axis/1_1/axis-1_1.zip

Decompress and add their location in the CLASSPATH variable.

Control Panel->System->Advanced->Environment Variables

3.2.1.3. Xerces 2.6.2

Download Xerces 2.6.2 http://archive.apache.org/dist/xml/xerces-j/Xerces-J-bin.2.6.1.zip

Decompress and add their location in the CLASSPATH variable.

Control Panel->System->Advanced->Environment Variables

3.2.1.4. MIPv6

The installation of the MS MIPv6 Tech Prev is under a NDA with Microsoft.

WS-Gateway

Name	Ksat94
OS	Mandrake 10.0 Linux
Kernel	Linux kernel 2.6.8.1
	Java 5.0
Mobile IPv6[1]	MIPL-2.0
DNS[2]	bind-9.2.3, bind-utils-9.2.3

[1] MIP support was installed in this machine in order to test it as a Correspondent Node (CN) which is totally independent of its function as a WS-Gateway. You can use any other machine in order to use it as a CN.

[2] A DNS server was installed in this machine however is this is totally independent of its functionality as a WS Gateway.

Tomcat	jakarta-tomcat-5.5.7
SOAP	Axis1.1
XML Parser	Xerces 2.6.2
Wireless	Wirelesstools-26

3.4.1. Installation procedure

3.4.1.1. Install kernel with mipl support.

Kernel compilation and so forth is exactly the same as the MN. We can actually copy the kernel that we had compiled for the MN in to the HA's */usr/src* directory. Then just do:

```
# make modules_install && make install && lilo –v
```

3.4.1.2. Install Java

Install Java as in the MN.

3.4.1.3. Install Tomcat

Download tomcat-5.5 and install:

```
# wget http://apache.engram.de/apache.org/jakarta/tomcat-5/v5.5.7/bin/jakarta-tomcat-5.5.7.tar.gz
# tar zxvvf jakarta-tomcat-5.5.7.tar.gz
# cp –R jakarta-tomcat-5.5.7 /usr/local/
# cd /usr/local; ln –s jakarta-tomcat-5.5.7 tomcat
# ./usr/local/tomcat/bin/startup.sh
```

After this we can confirm that tomcat is running by opening a browser and browsing: http://localhost:8080

(A common problem is to have a web server already running on that port)

3.4.1.4. Configure network:

Install wirelesstools

```
# urpmi wirelesstools
```

Configure wireless interface:

```
# iwconfig wlan0 essid Calimotxo mode Ad-hoc
```

In this node we used a different driver than in the MN that's why the interface is called wlanX instead of ethX. Look annexes for further description.

- Edit */etc/sysconfig/*network

```
NETWORKING=yes
HOSTNAME=ksat94
NETWORKING_IPV6=yes
IPV6FORWARDING=yes

IPV6_AUTOCONF=no
IPV6_AUTOTUNNEL=no

IPV6_DEFAULTGW="2001:638:202:100::100%eth0"

FORWARD_IPV4=true
GATEWAY=192.168.1.1

GATEWAYDEV=eth0
```

- Edit */etc/sysconfig/network-scripts/ifcfg-wlan0*. wlan0 is the networkcard that looks to the MN's network.

```
DEVICE=wlan0
BOOTPROTO=none
ONBOOT=yes

IPV6INIT="yes"
IPV6ADDR="2001:638:202:60::100/64"
IPV6_ROUTER=yes
IPV6_AUTOCONF=no
TYPE=Ethernet
USERCTL=no
PEERDNS=no

IPADDR=192.168.11.1
NETMASK=255.255.255.0
```

- Edit */etc/sysconfig/network-scripts/ifcfg-eth0*. Eth0 is the interface looking towards the back-bone.

```
DEVICE=eth0
BOOTPROTO=none
ONBOOT=yes
IPADDR=192.168.1.100
NETMASK=255.255.255.0

IPV6INIT="yes" # Enable IPv6 initialization of this interface
IPV6ADDR="2001:638:202:100::200/64"

IPV6TO4INIT="no" # Disable 6to4 initialization of this interface [default]


IPV6_ROUTER=yes # IPv6 autoconfiguration: interface is for routing
[default, if forwarding is on]


IPV6_AUTOCONF=no # disable IPv6 autoconfiguration [default, if forwarding
is on]
```

- Install radvd and edit */etc/radvd.conf*. This sends Router Advertisements to the network so that clients can forge an IPv6 address with the correspondent network prefix. Even though the MN doesn't need as it is manually configured we may need it afterwards or in order to easily attach new computers. (I configured the MN manually because that way the address is easier to remember.)

```
interface wlan0
{
        AdvSendAdvert on;
        AdvReachableTime 5000;
        AdvIntervalOpt on;
        MinRtrAdvInterval 1;
        MaxRtrAdvInterval 5;
        AdvSourceLLAddress on;

#
# new EUI-64 prefixes
#
        prefix 2001:0638:0202:0060::/64
      {
                AdvOnLink on;
                AdvAutonomous on;
#           AdvRouterAddr on;           #Just for MIPv6
        };


};
```

- In the case your Router to internet is not using IPv6, we will also need to send Router Advertisements to the back-bone network, so that the windows machine is able to forge its IPv6 address. (This is not the case in our USTUTT test-bed, since the Router Advertisements are sent by the back-bone router.)

```
interface eth0
{
        AdvSendAdvert on;
        AdvReachableTime 5000;
        AdvIntervalOpt on;
        MinRtrAdvInterval 1;
        MaxRtrAdvInterval 5;
        AdvSourceLLAddress on;

#
# new EUI-64 prefixes
#
        prefix 2001:0638:0202:0100::/64
      {
                AdvOnLink on;
                AdvAutonomous on;
#           AdvRouterAddr on;        # Just for MIPv6
```

```
        };

};
```

- In order to provide the MN with IPv4 internet access you can do it by using MASQUERADE.

```
# iptables –A POSTROUTING –t nat –o eth0 –j MASQUERADE
```

To make this automatic everytime the computer boots add the previous line in the file:

/etc/rc.d/rc.local

Regarding MIPL edit */usr/local/etc/mip6d.conf*. As I said before this function is totally independent of the work developed by the node as WS-Gateway and could be implemented in any other computer.

```
NodeConfig = CN;
DoRouteOptimizationCN = enabled;
UseMnHaIPSec = disabled;
```

- Restart the network:

```
# service network restart
```

- Restart the radvd.conf

```
# service radvd restart
```

- Start MIPL

```
# mip6d
```

4) DNS Configuration.

We are using the DNS to translate names into IPv6 addresses. As you will see there is no mapping of the names into IPv4 addresses in order to avoid using IPv4 without realizing.

- Install bind, bind-utils.

- Edit named.conf.

```
options {
        directory "/var/named";
        pid-file "/var/run/named/named.pid";
        listen-on-v6 {
```

```
                any;
                };
        forwarders {
        2001:638:202:1:a00:20ff:fe88:fcb1;
        2001:720:410:1001::140;
        };
};
controls {
        inet 127.0.0.1 allow { localhost; } keys { key; };
};

zone "0.0.1.0.2.0.2.0.8.3.6.0.1.0.0.2.ip6.arpa" {
        type master;
        file "reverse-2001-638-202-100_64.IP6.ARPA";
};
key "key" {
        algorithm       hmac-md5;
        secret
"c3Ryb25nIGVub3VnaCBmb3IgYSBtYW4gYnV0IG1hZGUgZm9yIGEgd29tYW4K";
};
zone "akogrimo6.rus.uni-stuttgart.de" in {
        type master;
        file "akogrimo6.rus.uni-stuttgart.de.hosts";
        };
logging {
        category resolver {
                default_syslog;
                default_debug;
                default_stderr;
                null;
                };
        category queries {
                default_syslog;
                default_debug;
                default_stderr;
```

```
        null;
        };
    category general {
        default_syslog;
        default_debug;
        default_stderr;
        null;
        };
    };
```

- Then we have to edit the zone and reverse look up files which go in the */var/named/* folder.

/var/named/akogrimo6.rus.uni-stuttgart.de.hosts

```
$TTL 86400
akogrimo6.rus.uni-stuttgart.de. IN      SOA     ksat94. patrick\.mandic.rus.uni-stuttgart.de. (
                8
                10800
                900
                604800
                3600)

    IN      NS      ksat94.

ksat96.akogrimo6.rus.uni-stuttgart.de.  IN      AAAA    2001:638:202:100::203
ksat97.akogrimo6.rus.uni-stuttgart.de.  IN      AAAA    2001:638:202:100::201
ksat94.akogrimo6.rus.uni-stuttgart.de.  IN      AAAA    2001:638:202:100::200
ksat93.akogrimo6.rus.uni-stuttgart.de.  IN      AAAA    2001:638:202:100::202
```

/var/named/reverse-2001-638-202-100_64.IP6.ARPA. I created this file making use of a web tool (http://www.fpsn.net/index.cgi?pg=tools&tool=ipv6-inaddr). If you do it that way you will avoid some typing mistakes. The backslash (\) means a broken line and shouldn't appear in the real file.

```
; http://tools.fpsn.net/ipv6-inaddr
;
$TTL 3d ; Default TTL (bind 8 needs this, bind 9 ignores it)
0.0.1.0.2.0.2.0.8.3.6.0.1.0.0.2.ip6.arpa.       IN      SOA     0.0.1.0.2.0.2.0.8.3.6.0.1.0.0.2.ip6.arpa.\
```

```
patrick.mandic.rus.uni-stuttgart.de. (
                200501267
                24h
                30m
                2d
                3d)

                ; Name server entries
                IN     NS     ksat94.
; IPv6 PTR entries

; Subnet #1
$ORIGIN 0.0.1.0.2.0.2.0.8.3.6.0.1.0.0.2.ip6.arpa.

0.0.2.0.0.0.0.0.0.0.0.0.0.0.0.0        IN      PTR     ksat94.akogrimo6.rus.uni-stuttgart.de.


;
; End of zone file.
; Thank you for using the fpsn.net IPv6 Reverse DNS zone builder

; Additionally you will need to add the following AAAA record entries
; to their respective zone files:
; ksat94.akogrimo6.uni-stuttgart.de.        IN      AAAA    2001:638:202:100::200
; ksat93.akogrimo6.uni-stuttgart.de.        IN      AAAA    2001:638:202:100::201

3.0.2.0.0.0.0.0.0.0.0.0.0.0.0.0.1.0.2.0.2.0.8.3.6.0.1.0.0.2.ip6.arpa.          IN          PTR  \
ksat96.akogrimo6.rus.uni-stuttgart.de.
1.0.2.0.0.0.0.0.0.0.0.0.0.0.0.0.1.0.2.0.2.0.8.3.6.0.1.0.0.2.ip6.arpa.          IN          PTR  \
ksat97.akogrimo6.rus.uni-stuttgart.de.
2.0.2.0.0.0.0.0.0.0.0.0.0.0.0.0.1.0.2.0.2.0.8.3.6.0.1.0.0.2.ip6.arpa.          IN          PTR  \
ksat93.akogrimo6.rus.uni-stuttgart.de.
```

/etc/resolve.conf

```
search akogrimo6.rus.uni-stuttgart.de
nameserver ::1
```

- (Re-)Start named.

```
# /etc/init.d/named restart
```

3.5. Simple WS

Name	Ksat96
OS	Windows Server 2003
Java	Java 5.0
Internet Information Services IIS 5.1	
.NET Framework 1.1	

3.5.1. Installation procedure

3.5.1.1. Install Windows Server 2003

- Install Windows Server 2003 with Internet Information Services IIS or add IIS to the existing installation via the control panel "Add/Remove Windows Components"
    - When you have selected the Application Server->IIS item click on details and make sure "FrontPage 2002 Server Extension" is selected.
- Check the version of IIS:
    - Double click C:\WINDOWS\SYSTEM32\INETSRV\IIS.msc
    - From the "?" menu select "Info…"
- Install the .NET Framework 1.1 redistributable, e.g. by starting Windows Update and selecting the 1.1 version from the optional components.

3.5.1.2. Install Java:

http://java.sun.com/j2se/1.5.0/download.jsp

3.5.1.3. Network configuration

Refer to Annex B for a list of useful Windows network commands.

1) Install IPv6

- Open a console

```
/> netsh interface ipv6 install
/> netsh interface ipv6 set address "Local Area Connection" 2001:638:202:100::203
```

2) Configure IPv6 DNS

- Open a console

```
/> netsh interface ipv6 add dns "Local Area Connection" 2001:638:202:100::200
```

In order for the IPv6 DNS to work the regular TCP/IP support has to be activated. (Windows must be reusing some modules or something…). In addition, we will actually configure the suffixes for IPv6 in the IPv4 TCP/IP settings.

- Local Area Connection -> Properties -> TCP/IP properties->"advanced" button-> DNS tab -> Append these DNS suffixes: *akogrimo6.rus.uni-stuttgart*

3.6. MIP Home Agent (HA)

Name	Ksat96
OS	Mandrake 10.0
Kernel	2.6.8.1
Java	Java 1.4.2_06
Openssl	Libopenssl0.9.7-devel-0.9.7c Openssl-0.9.7
Mobile IPv6	MIPL-2.0

3.6.1. Configuration

3.6.1.1. Install kernel with mipl support.

Kernel compilation and so forth is exactly the same as the MN. We can actually copy the kernel that we had compiled for the MN in to the HA's */usr/src* directory. Then just do:

```
# make modules_install && make install && lilo –v
```

3.6.1.2. Network configuration

Edit */etc/sysconfig/*network

```
HOSTNAME=ksat97
NETWORKING=yes
GATEWAY=192.168.1.1

NETWORKING_IPV6=yes
IPV6_AUTOCONF=no
IPV6_DEFAULTGW="2001:638:202:100::100%eth0"
IPV6FORWARDING=yes
```

Edit */etc/sysconfig/network-scripts/ifcfg-ethX*; where X is the interface number of your network card. In this case was eth1

```
DEVICE=eth0
BOOTPROTO=none
IPADDR=192.168.1.102
NETMASK=255.255.255.0
NETWORK=192.168.1.0
BROADCAST=192.168.1.255
ONBOOT=yes

IPV6INIT="yes"
IPV6ADDR="2001:638:202:100::201"
IPV6_AUTOCONF="no"
```

In the case we don't have access to the default router in order to add the 2001:638:202:60::/64 network to the routes, we will need to add the route in the HA.

```
# route –A inet6 add 2001:638:202:60::/64 gw 2001:638:202:100::200 dev eth0
```

In order to make this permanent:

```
# echo "eth0 2001:638:202:60::/64 2001:638:202:100::200" >> /etc/sysconfig/static-routes-ipv6
```

Edit /usr/local/etc/mip6d.conf:

```
NodeConfig = HA;
HomeAgentInterfaceList = ("eth0);
UseMNHaIPSec = disabled;
```

Restart the network:

```
# service network restart
```

Start MIPL:

```
# mipd6d
```

/etc/resolve.conf

```
search akogrimo6.rus.uni-stuttgart.de
nameserver 2001:638:202:100::200
```

4. Tests

4.1. Suggestions

- For management reasons and installing software from internet all the nodes support IPv4 and IPv6 simultaneously. Once the all the applications are installed and the tests are being run, the use of IPv4 should be avoided in the machines that are not using IPv4 for testing purposes – that is, all but the WS-Gateway, which communicates with google using IPv4. It is quite common in machines with dual stack that we think we are communicating using IPv6 when in fact IPv4 is the protocol handling the communication instead. Sometimes is quite useful to use a network sniffer like tcpdump or ethereal in order to find out what is really going on in the network. Regarding the Window Server 2003 and Windows XP SP1, the IPv4 interface needs to be activated for the DNS queries to work. Windows Server 2003 will use IPv6 to do the queries, however Windows XP can only use IPv4 to do them.
- In order to run tests and applications that support IPv6 a very good think to know the format for literal IPv6 addresses in URL's which is explained in RFC 2732. In order to use a literal IPv6 address in an URL it should be enclosed in the "[" and "]" characters.

  For example:

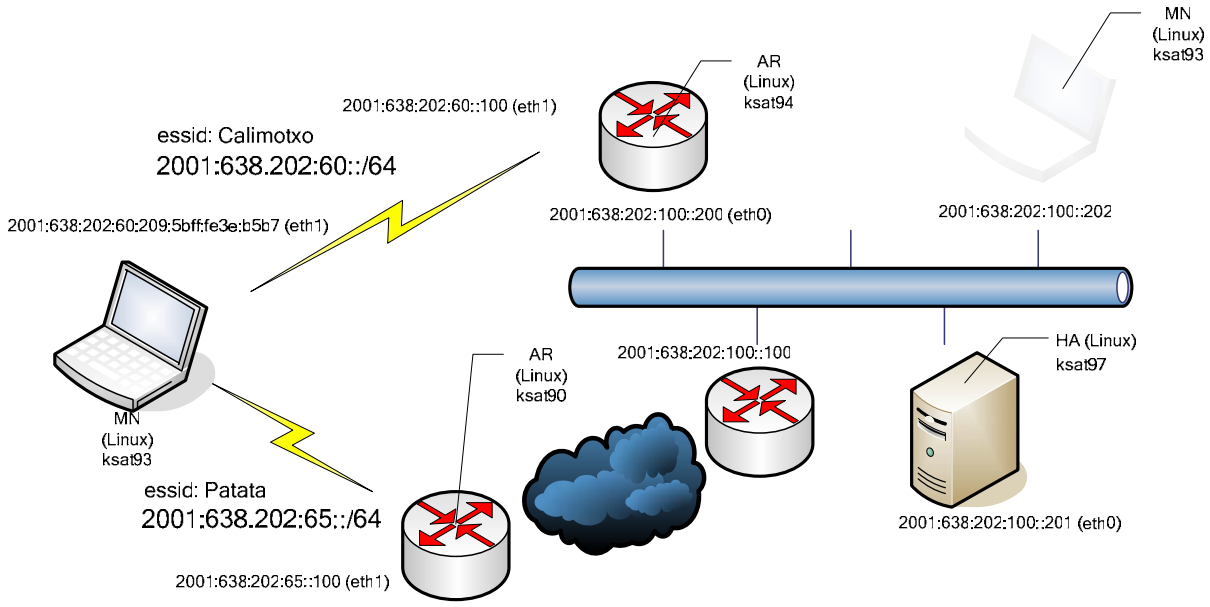
  `FEDC:BA98:7654:3210:FEDC:BA98:7654:3210`

  `Would be represented as follows:`

  `Fehler! Hyperlink-Referenz ungültig.`

4.2. MIPv6 test:

The functionality of MIPv6 has been tested. Hand-overs and communications can be carried out without any problems, however the route optimization procedure doesn't seem to work. The following architecture has been used to perform the tests.



- A handover is performed by simply changing the name of the essid to which the MN is connected (iwconfig INTERFACE essid NETWORK_NAME in linux and selecting wireless network in windows). The handover was successful using Windows XP, Windows CE and Linux as the MN.
- Regarding the Route Optimization procefure (RO) by which the route between a MN and a CN is optimized, it works fine with linux although the following happened to us a few times:

  Using ksat94 as a CN and paying attention to the mip6d logs (running mip6d –d 10), it can be seen how BUs are (irregularly) sent to the CN and a BA is returned, the MN´s mip6d breaking down with a segmentation fault afterwards.

  RO was successfully performed by Windows XP and Windows CE MNs.
- The use of a Windows Server 2003 as a CN was also tested. Even though the Windows Server 2003 family doesn't fully implement IPv6 Mobility, it does implement the CN functionality. However this implementation is based on the mobility for IPv6 draft 14, and didn't work at all. The CN would complain about not understanding mobility headers, since in draft 14 they were still not specified.
- IPSec was successfully tested to secure BUs and BAs in Linux.
- It is also important to remark that DNS doenst work over IPv6 as transport protocol for Windows XP and Pocket PC 2003. Instead, IPv4 needs to be used to resolve IPv6 addresses, which may limit somewhat mobility since a MN needs to always have IPv4 and (M)IPv6 connectivity in order to be completely functional (Unless the users are able to remember by heart 16-digit long hexadecimal numbers)

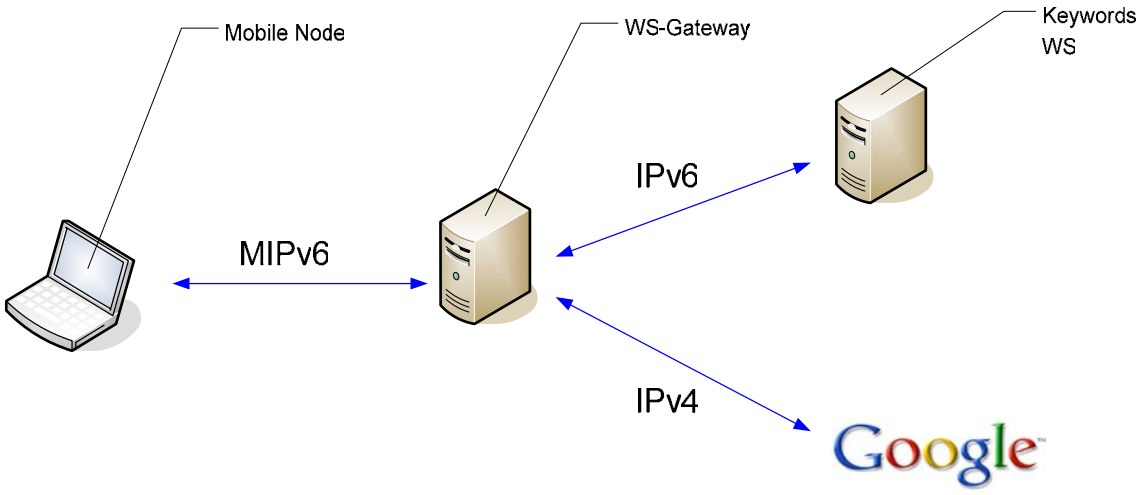
4.3. WS Tests

This chapter describes the development of the Web services and Web service clients that are used in this Howto testbed. The development is realized with Java5, Axis1.1 and the XML

Parser Xerces2.6.2 and .NET Framework 1.1, however it could have been build using any other kind of Web Services toolkit. The testbed is organized around the public Web Service of Google. The whole testbed is outlined in Figure 4 where a Web service client (Mobile Node) requests the TermsCounts method offered by the Gateway Web service (WS-Gateway). Then, the WS-Gateway asks the Keyword Web service (Keywords WS) for the keywords to search in google. With these keywords the WS-Gateway queries the Google Web service obtaining some results that are sent back to the Mobile Node Client. In the following chapters a description of the software used to perform this test is given.

Regarding the Operative system used we used Windows Server 2003 for the Keywords WS, Linux (Mandrake 10.0) for the Gateway, and Linux (Mandrake 10.0), Windows XP and Pocket PC 2003 for the Mobile Node Client. The WS technologies used in each node are: .NET and IIS for the Keywords WS, Google WS, java, Tomcat, axis and xerces parser for the WS-Gateway, and on the MN java, axis and the xerces parser when using Linux and Windows XP and .NET Compact Framework 2.0 (Beta) – we didn't try this one yet - when using Pocket PC 2003.

The code of the applications developed to carry out the test can be downloaded from the bscw server http://www.hlrs.de/bscw/bscw.cgi/0/53267.



Keywords WS

To enable the IPv6 support of the.NET Framework, edit the machine.config file found in:

C:\WINDOWS\Microsoft.NET\Framework\v1.1.4322\CONFIG\

Edit the file and replace "<!-- <ipv6 enabled="false"/> -->" by "<ipv6 enabled="true"/>"

4.3.1.1. Installation

Download the file Testbed_Windows_KeywordsService.zip from http://www.hlrs.de/bscw/bscw.cgi/0/53291 . Use the enclosed KeywordsSetup.msi installer package to install the Keywords Web-service on the Windows 2003 machine. You may either use

the Release or the Debug version. Optionally you can copy the file Keywords.txt to C:\ so the keywords are read from file.

Start IIS 6.0.

```
[WebMethod]
public string GetKeyword()
{
      return "Akogrimo";
}

[WebMethod]
public string[] GetKeywords()
{
      return GetKeywordsFromFile( "C:\\Keywords.txt" );
}
```

Table 1 - C# Code

The Web-service implements two methods. GetKeyword() returns a string, and GetKeywords() returns a string-array.

4.3.2. Google WS

In order to use the Google Web-Service you need to register here and generate a *License Key*. This key will have a limit on the number of requests a day that you can make. The default limit is 1000 queries per day. The next step is to use the WSDL file in order to generate the stubs classes for the client side. Therefore you can use the link to Google WSDL (http://api.google.com/googleSearch.html) or download the Google developer's kit with the WSDL. After that you'll find the AXIS WSDL2Java tool in "org.apache.axis.wsdl.WSDL2Java". The basic invocation form looks like this:

% java org.apache.axis.wsdl.WSDL2Java (WSDL-file-or-URL)

For this discussion, assume we executed the following:

% java org.apache.axis.wsdl.WSDL2Java http://api.google.com/GoogleSearch.wsdl

The generated files will reside in the directory "GoogleSearch". They are put there because that is the target namespace from the WSDL and namespaces map to Java packages. These generated files we'll use in the Gateway service but if somebody is interested in writing a Google command line client here is an example source code:

```
import GoogleSearch.*;


public class GoogleClient
{
  public static void main(String [] args)
  {
    try
    {
      GoogleSearchService dienst = new GoogleSearchServiceLocator();
      GoogleSearchPort port = dienst.getGoogleSearchPort();
      GoogleSearchResult result = port.doGoogleSearch
("License_Key_String","Akogrimo",0,5,true,"",true,"","","");
```

```
      System.out.println("Searchquery: " + result.getSearchQuery());
      System.out.println("Search number: " +
result.getEstimatedTotalResultsCount());
      System.out.println("Startindex: " + result.getStartIndex());
      System.out.println("Endindex: " + result.getEndIndex());

      ResultElement[] res = result.getResultElements();

      for (int i = 0; i < result.getEndIndex(); i++ )
      {
        System.out.println("URL: " + res[i].getURL());
        System.out.println("Title: " + res[i].getTitle());
        System.out.println("Snippet: " + res[i].getSnippet());
        System.out.println("Directory Category: " +
res[i].getDirectoryCategory());
        System.out.println("Directory Title: " +
res[i].getDirectoryTitle());
        System.out.println("Summary: " + res[i].getSummary());
        System.out.println("Cached Size: " + res[i].getCachedSize());
        System.out.println("Related Information Present: " +
res[i].isRelatedInformationPresent());
        System.out.println("Host Name: " + res[i].getHostName());
      }
    }
    catch (Exception e)
    {
       System.err.println(e.toString());
    }
  }

}
```

Only what you have to do in this client example is writing your license key in the port.doGoogleSearch method and compile the source code.

4.3.3. Gateway WS

For developing the Gateway service the steps are following:

1. Generate client classes with the WSDL from the Keywords Web service.

The Keywords Web service is hopefully up and running if not please sees above. If yes, use the WSDL2Java tool again like for the Google client:

% java org.apache.axis.wsdl.WSDL2Java (WSDL-file-or-URL)

In this case it is:

% java org.apache.axis.wsdl.WSDL2Java http://ksat96/Testbed/Keywords.asmx?WSDL

The generated files will reside in the directory " org\akogrimo\www\Testbed".

2. Generate client classes with the WSDL from the Google Web service.

Just use the generated Google client classes again from chapter 4.3.3 Google WS.

3. Write the Gateway Web service

In the testbed the Gateway WS is a simple class with one method "TermsCounts" that gives back the Google search results. This method uses the generated Keywords client classes for requesting the search keys and after that it uses the Google client classes for requesting the search results regarding the search keys. The source code for this Gateway WS is:

```
/* Web Service, response search keys and the number of Google search
results*/
```

```
import org.apache.axis.client.Call;
import org.apache.axis.client.Service;
import GoogleSearch.*;
import org.akogrimo.www.Testbed.*;

public class TermsCounts_ksat_WS
{
  public String[][] TermsCounts() throws javax.xml.rpc.ServiceException,
                        java.net.MalformedURLException,
java.rmi.RemoteException
  {
    Service  service = new Service();
    Call     call    = (Call) service.createCall();
    String termsCounts[][] = new String[3][];

    // Deliver search keys – using Keywords Web Service

    KeywordsLocator dienst1 = new KeywordsLocator();
    KeywordsSoap port1 = dienst1.getKeywordsSoap(new
java.net.URL("http://ksat96/Testbed/Keywords.asmx"));
    String[] terms = port1.getKeywords().getString();

    // deliver Google results - using Google WebService
    GoogleSearchService dienst = new GoogleSearchServiceLocator();
    GoogleSearchPort port = dienst.getGoogleSearchPort();
    String[] circa = new String[terms.length];
    String[] counts = new String[terms.length];
    for (int i=0; i<terms.length; i++)
    {
      GoogleSearchResult suche = port.doGoogleSearch("Google_License_
key",terms[i],0,5,true,"",true,"","","");
      circa[i]=circa[i].valueOf(suche.getEstimatedTotalResultsCount());
      counts[i]=counts[i].valueOf(suche.getEndIndex());
    }
    termsCounts[0]=terms; termsCounts[1]=circa; termsCounts[2]=counts;
    return termsCounts;
  }
}
```

Don't forget to write your license key in the port.doGoogleSearch method before compiling the source code.

4. Copy all necessary classes and libraries to the Tomcat servlet engine and start Tomcat.

First, you have to create the following directory structure in the Tomcat "webapps" directory:

- webapps/axis/WEB-INF/classes
- webapps/axis/WEB-INF/lib

After this copy all class files to the Tomcat classes directory and all necessary libraries (axis and Xerces)to the Tomcat "lib" directory and start the Tomcat

5. Deploy the Gateway Web service.

At least the Gateway Web service has to be deployed to the Tomcat with the Web Service Deployment Descriptor. For the Gateway WS the deployment file (with name deploy.wsdd) is:

```
<deployment xmlns="http://xml.apache.org/axis/wsdd/"
            xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">
 <service name="TermsCounts_ksat_WS" provider="java:RPC">
  <parameter name="className" value="TermsCounts_ksat_WS"/>
  <parameter name="allowedMethods" value="*"/>
 </service>
</deployment>
```

Once you have this file, you need to send it to an Axis server in order to actually deploy the described service. You can do this with the AdminClient, or the "org.apache.axis.client.AdminClient" class. If you have deployed Axis on a server other than Tomcat, you may need to use the -p <*port*> argument. The default port is 8080. A typical invocation of the AdminClient looks like this:

% java org.apache.axis.client.AdminClient deploy.wsdd

<Admin>Done processing</Admin>

After doing this, you should see now in an Internet Browser the running Gateway WS at the following URL with the method TermsCounts:

http://ksat94:8080/axis/services/TermsCounts_ksat_WS

with the wsdl:

http://ksat94:8080/axis/services/TermsCounts_ksat_WS?wsdl

Of course it's possible to request the TermsCounts method with:

http://ksat94:8080/axis/services/TermsCounts_ksat_WS?method=TermsCounts

4.3.4. MN Windows XP WSClient

Now that the Gateway WS is running it's time to develop the Gateway Web service client. In the chapters before we use the WSDL2Java tool but it's possible to develop a Web service client by using the axis classes. Here is an example of a command line client by using the axis classes:

```
import org.apache.axis.client.Call;
import org.apache.axis.client.Service;
import org.apache.axis.encoding.XMLType;
import org.apache.axis.utils.Options;

import javax.xml.rpc.ParameterMode;

public class TermsCounts_ksat_WC
{
  public static void main(String [] args) throws Exception
  {
    try
    {
      String endpoint =
"http://ksat94:8080/axis/services/TermsCounts_ksat_WS";
      Service service = new Service();
      Call    call    = (Call) service.createCall();
      System.out.println("create Call...");
      call.setTargetEndpointAddress( new java.net.URL(endpoint) );
      call.setOperationName( "TermsCounts" );
      call.setReturnType( XMLType.SOAP_ARRAY );
      System.out.println("set Endpoint, operation name, return type ...");
      String[][] ret = (String[][]) call.invoke( new Object [] {} );
      for (int i=0; i<ret[0].length; i++)
        System.out.println("Search key: "    +ret[0][i]+"\t"+
                           "Google result: "+ret[1][i]+"\t"+
                           "Result return: "  +ret[2][i]);
    }
    catch (Exception e)
      {
         System.err.println("Clientfehler: ");
       System.err.println(e.toString());
      }
  }
}
```

This Gateway WS client requests the TermsCounts method and gets back the search key, the amount of the Google search result and the amount of the result return.

4.3.5. MN Linux WSClient

The MN Linux WS client and the MN Windows WS client are both the same, so please see chapter 4.3.5.

4.3.6. MN Pocket PC 2003 WSClient

In order to develop WS for the Pocket PC 2003 there are several alternatives. One of them would be to use Java, however the java versions supported (java 1.2) is quite old and don't support IPv6.

On the other hand the Microsoft .NET Framework can be used however, the current stable version for Pocket PC (.NET Compact Framework 1.1) doesn't support IPv6. .NET Compact Framework 2.0 (beta version) is supposedly available for MSDN subscriptors and supports IPv6. We didn't try .NET Compact Framework 2.0 in the Pocket PC 2003 yet.

5. Conclusions

By doing the exercise of building a small testbed some issues have arisen that wouldn't have been taken into account otherwise. The following table summarizes the required dependencies that need to be solved in order to build a WS environment over IPv4, IPv6 and MIPv6 combining the Operating systems Windows Server 2003, Windows XP SP1, Power PC 2003 and Linux.

IPv6 dependencies

	Windows Server 2003	Windows XP SP1	Power PC 2003	Linux
MIPv6	No. It only implements CN functionality of draft-13 => not usable	Yes. With MS MIPv6 Tech Preview based on draft-24. (Just for SP1)	Yes. With MS MIPv6 Tech Preview based on draft-24.	Yes. With MIPL (RFC 3775).
DNS over IPv6	Yes.	No. (IPv6 addressed are requested over IPv4)	No. (IPv6 addressed are requested over IPv4)	Yes.
Java 1.4 IPv6 support	Supposedly yes, however **it didn't work**.	Supposedly yes, however **it didn't work**.	Java 1.4 not supported. Earlier versions don't support IPv6.	Yes.
Java 5.0 IPv6 support	Yes.	Yes.	Java 5.0 not supported.	Yes.
IPSec (IPv6)	ESP encryption not supported. IKE not supported.	ESP only supported with MS MIPv6 Tech Preview. IKE not supported.	ESP only supported with MS MIPv6 Tech Preview. IKE not supported.	Yes.
Axis standalone server IPv6 support	No.	No.	No.	No.
Axis with Tomcat IPv6 support	Yes.	Yes.	Yes.	Yes.
.NET IPv6	Yes. .NET Framework >=1.1	Yes. .NET Framework >=1.1	Yes. .NET Compact Framework	MONO >= 0.26 Yes.

	=>2.0 (Beta) (not tested yet)	
	(not tested yet)	

Non IPv6 dependencies

	Java 1.4	Java 5.0
Tomcat 5.0	Yes	No
Tomcat 5.5	Yes. With additional packate (compact)	Yes. By default.

Annex A. IPSec for MIPv6

1) Install ipsec-tools-0.2.5
2) Create a file with the configuration of the security associations in both, the MN and the HA:

```
# touch /etc/sa.conf
# chmod 600 /etc/sa.conf
```

Edit the file

sa.conf

```
#First MN -> HA SA for BUs
add  2001:638:202:100::202 2001:638:202:100::201 esp 2000
        -m transport
        -E des-cbc "Whateverkey"
        -A hmac-sha1 "this is a test key" ;
#Then HA -> MN SA for BAs
add  2001:638:202:100::201 2001:638:202:100::202 esp 2001
        -m transport
        -E des-cbc "Whateverkey"
        -A hmac-sha1 "this is a test key" ;
```

If we want to use more robust random keys they can be built this way:

```
# dd if=/dev/random count=16 bs=1 | xxd -ps
```

3) Load the configuration:

```
# setkey –f /etc/sa.conf
```

Annex B. Wireless drivers

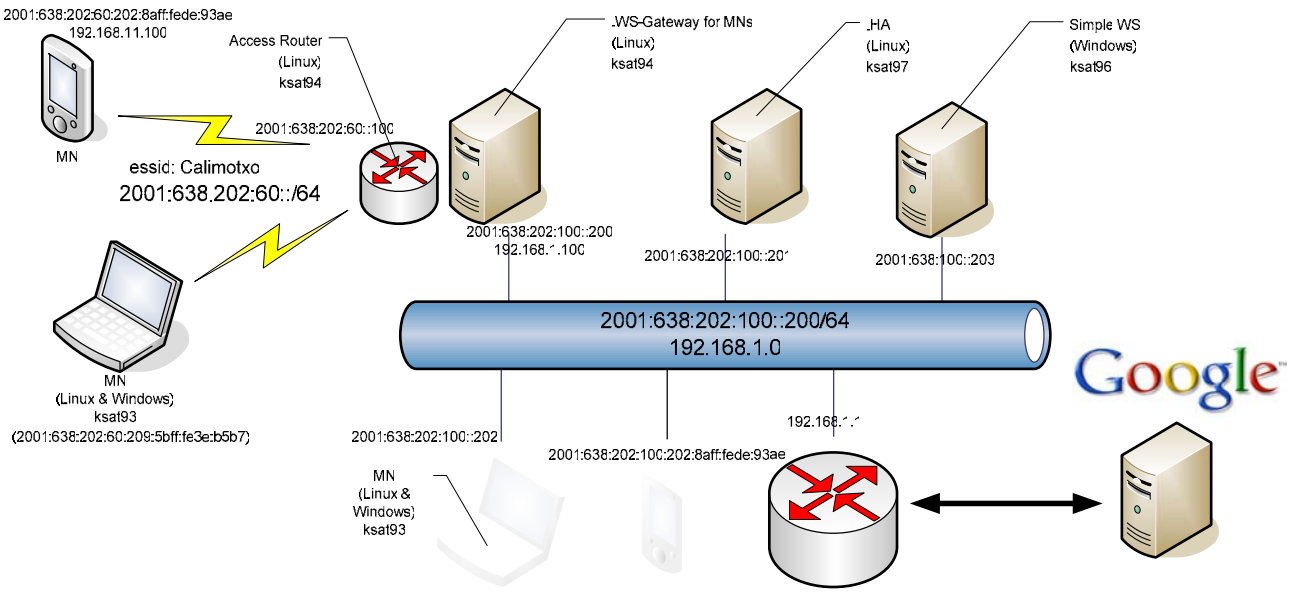
The drivers needed for the wireless cards to work depend on the hardware that you have. In our case we are using Netgear MA401 cards which use the chipset prims2. The "orinoco_cs" driver shipped with the standard kernel is enough to support this chipset but it lacks of some functionality like the possibility of monitoring the network. In our case we installed the Orinoco modules in the MN and a more versatile driver in the AR called "hostap". The latest version of this driver can be downloaded from http://hostap.epitest.fi

For more information about the different drivers, please refer to: http://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/

Annex C. Windows Network Commands

Install IPv6	netsh interface ipv6 install
List addresses	ipconfig /all
Add address to interface	netsh interface ipv6 add address INTERFACENAME ADDRESS Ex: netsh interface ipv6 add address "Local Area Connection" 3ffe:ffff::1
Remove address from interface	netsh interface ipv6 delete address INTERFACENAME ADDRESS Ex: netsh interface ipv6 delete address "Local Area Connection" 3ffe:ffff::1
Add dns	netsh interface ipv6 add dns INTERFACENAME ADDRESS Ex: netsh interface ipv6 add dns "Local Area Connection" 3ffe:ffff::2
Add route	netsh interface ipv6 add route INTERFACENAME ADDRESS Ex: netsh interface ipv6 add dns "Local Area Connection" 3ffe:ffff:3
Ping	ping ADDRESS Ex: ping 2001:0DB88::4
Ping forcing IPv6 address resolution	ping -6 NODE_NAME Ex: ping -6 ksat94
Traceroute	tracert ADDRESS Ex: tracert 3ffe:ffff::3
Display network latency and loss	pathping ADDRESS Ex: pathping 3ffe:ffff::2
Display network connections	netstat –s

Annex D.


2001:638:202:60:202:8aff:fede:93ae
192.168.11.100
Access Router
(Linux)
ksat94
WS-Gateway for MNs
(Linux)
ksat94
HA
(Linux)
ksat97
Simple WS
(Windows)
ksat96
MN
essid: Calimotxo
2001:638.202:60::/64
2001:638:202:100::200
2001:638:202:100::203
2001:638:202:100::200/64
192.168.1.0
MN
(Linux & Windows)
ksat93
(2001:638:202:60:209:5bff:fe3e:b5b7)
MN
(Linux &
Windows)
ksat93
Google