Lab 4 Mobile IP

April 14, 2011

Contents

1 Introduction

This document describes the software and procedures to set up and use mobile IPv6 for Linux.

1.1 What is Mobile IP?

Each mobile node is always identified by its home address, regardless of its current point of attachment to the Internet. While situated away from its home, a mobile node is also associated with a care-of address, which provides information about the mobile node's current location. IPv6 packets addressed to a mobile node's home address are transparently routed to its care-of address. The protocol enables IPv6 nodes to cache the binding of a mobile node's home address with its care-of address, and to then send any packets destined for the mobile node directly to it at this care-of address.

1.2 Why Mobile IP?

Without specific support for mobility in IPv6, packets destined to a mobile node would not be able to reach it while the mobile node is away from its home link. In order to continue communication in spite of its movement, a mobile node could change its IP address each time it moves to a new link, but the mobile node would then not be able to maintain transport and higher-layer connections when it changes location. Mobility support in IPv6 is particularly important, as mobile computers are likely to account for a majority or at least a substantial fraction of the population of the Internet during the lifetime of IPv6.

1.3 How does it work?

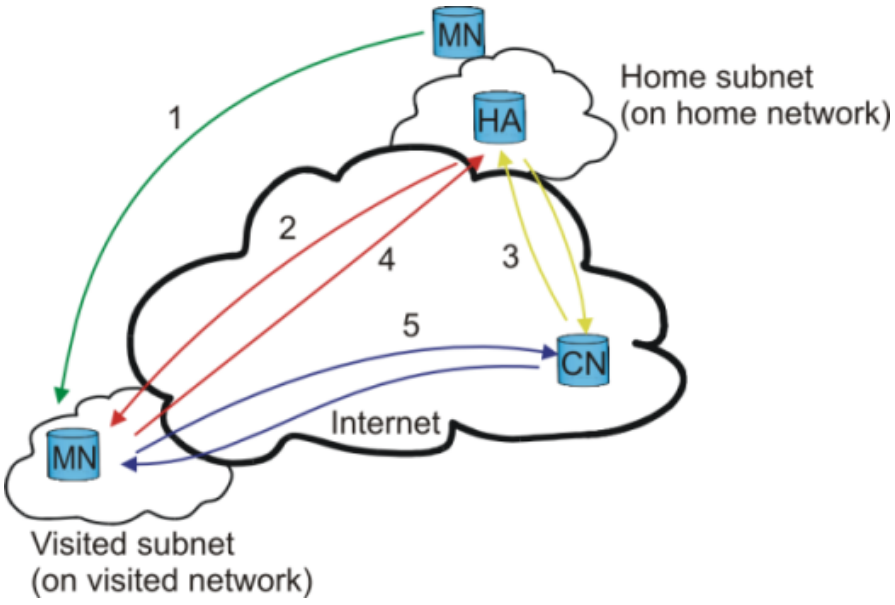

Figure 1: Mobile Ip

As figure 1:

1. The Mobile Node(MN) travels to a foreign network and gets a new care-of-address.
2. The MN performs a binding update to its Home Agent (HA) (the new care-of-address gets registered at HA). HA sends a binding acknowledgement to MN.
3. A Corresponding Node(CN) wants to contact the MN. The HA intercepts packets destined to the MN.
4. The HA then tunnels all packets to the MN from the CN using MN's care-of-address.
5. When the MN answers the CN, it may use its current care-of-address(and perform a binding to the CN) and communicate with the CN directly (route optimization) or it can tunnel all its packets through the HA.

1.4 IPv6

IP version 6 (IPv6) is a new version of the Internet Protocol, designed as the successor to IP version 4 (IPv4). The changes from IPv4 to IPv6 fall primarily into the following categories:

1. Expanded addressing capabilities
2. Header format simplification
3. Improved support for extensions and options
4. Flow labeling capability
5. Authentication and privacy capabilities

For more information on IPv6 in general, visit the IETF's IPv6 Working Group.

2 Test bed

Now you should have a working MIPv6 patched kernel and installed userlevel tools. If anything went wrong, go through the above sections carefully.

2.1 Testcase

The addresses we are using in our test-bed are global. Our test-bed consist of four nodes, see figure 2.

1. HA - Home Agent: The HA is located at the home network. It has one wireless interface with address `200X:5c0:8d03:1::1` (**Note:** `X` is your testbed Group ID, we have 3 group there in the labroom. And MN, AR, CN should obey this rule too.) and a wired interface with address `200X:5c0:8d03:2::1`.
2. MN - Mobile Node: When MN is on the "home network", it has address `200X:5c0:8d03:1::2`. When MN travels to another network, it generates a new "care-of" address.

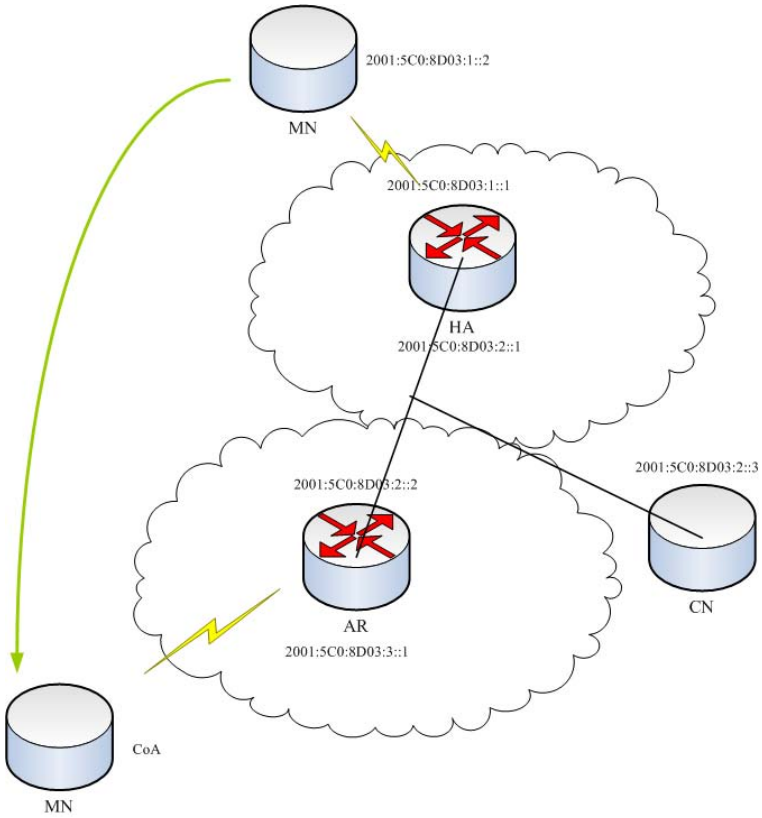

Figure 2: Mobile IPv6 testbed

3. AR - Access Router: The link between AR and R is our "internet" (may be a cross-cable). The AR has two interfaces; the wired interface to R has address `200X:5c0:8d03:2::2`, the wireless has address `200X:5c0:8d03:3::1`.

4. CN - Corresponding Node: sending messages to MN, with a wired address `200X:5c0:8d03:2::3`.

2.2 Setting up a fully functional IPv6 network

Note: all the nodes should be able to ping each other, and the different wireless networks have different ESSIDs.

For each of the desktops, open a terminal and type the following command to become root(super administrator).

```
sudo su
```

the password is: sjtu

Then, we configure each desktop as different nodes.

1. MN: The Mobile Node has one wireless interface. Forwarding should be turned off, but should accept autoconf and ra's.

```
iwconfig wlan0 mode ad-hoc essid homenet_groupX enc off
ifconfig wlan0 inet6 add 200X:5c0:8d03:1::2/64
iwconfig wlan0 channel 1
echo "0" > /proc/sys/net/ipv6/conf/wlan0/forwarding
```

```
echo "1" > /proc/sys/net/ipv6/conf/wlan0/autoconf
echo "1" > /proc/sys/net/ipv6/conf/wlan0/accept_ra
echo "1" > /proc/sys/net/ipv6/conf/wlan0/accept_redirects
echo "0" > /proc/sys/net/ipv6/conf/all/forwarding
echo "1" > /proc/sys/net/ipv6/conf/all/autoconf
echo "1" > /proc/sys/net/ipv6/conf/all/accept_ra
echo "1" > /proc/sys/net/ipv6/conf/all/accept_redirects
```

2. HA: It should have forwarding turned on because it uses normal routing to deliver packets captured from a physical interface to the virtual tunnel interface.

```
ifconfig eth1 inet6 add 200X:5c0:8d03:2::1/64
iwconfig wlan0 mode ad-hoc essid homenet_groupX enc off
ifconfig wlan0 inet6 add 200X:5c0:8d03:1::1/64
iwconfig wlan0 channel 1
echo 1 > /proc/sys/net/ipv6/conf/wlan0/forwarding
echo 0 > /proc/sys/net/ipv6/conf/wlan0/autoconf
echo 0 > /proc/sys/net/ipv6/conf/wlan0/accept_ra
echo 0 > /proc/sys/net/ipv6/conf/wlan0/accept_redirects
echo 1 > /proc/sys/net/ipv6/conf/all/forwarding
echo 0 > /proc/sys/net/ipv6/conf/all/autoconf
echo 0 > /proc/sys/net/ipv6/conf/all/accept_ra
echo 0 > /proc/sys/net/ipv6/conf/all/accept_redirects
ip route add 200X:5c0:8d03:3::/64 via 200X:5c0:8d03:2::2
```

3. AR: The Access Router (on a foreign network) also has two interfaces; one wireless and one line. Forwarding must be turned on.

```
ifconfig eth1 inet6 add 200X:5c0:8d03:2::2/64
iwconfig wlan0 mode ad-hoc essid visitnet_groupX enc off
ifconfig wlan0 inet6 add 200X:5c0:8d03:3::1/64
iwconfig wlan0 channel 3
echo 1 > /proc/sys/net/ipv6/conf/all/forwarding
echo 0 > /proc/sys/net/ipv6/conf/all/autoconf
echo 0 > /proc/sys/net/ipv6/conf/all/accept_ra
echo 0 > /proc/sys/net/ipv6/conf/all/accept_redirects
ip route add 200X:5c0:8d03:1::/64 via 200X:5c0:8d03:2::1
```

4. CN: The source of the video stream sending to the MN.

```
ifconfig eth1 inet6 add 200X:5c0:8d03:2::3/64
ip route add 200X:5c0:8d03:1::/64 via 200X:5c0:8d03:2::1
```

If you want to test if two nodes are connected, use the following command:

```
ping6 <IPv6 address>
```

for example,

```
ping6 200X:5c0:8d03:3::1
```

2.3 Configuring Mobile IPv6

When you issue make install, neither init script nor configuration file is installed. Example configuration files are found under the extra/ directory under the source code.

The configure file would be /etc/mip6d.conf. As you should make sure that this file is exist and properly configured.

HA: The HA config file should contain these settings:

```
# Mobile IPv6 configuration file: Home Agent
# filename: /etc/mip6d.conf
NodeConfig HA;
# If set to > 0, will not detach from tty
DebugLevel 10;
# List of interfaces where we serve as Home Agent
Interface "wlan0";
UseMnHaIPsec disabled;
```

MN: The MN config file should look like this:

```
# Mobile IPv6 configuration file: MN
# filename: /etc/mip6d.conf
NodeConfig MN;
# If set to > 0, will not detach from tty
DebugLevel 10;
MnDiscardHaParamProb enabled;
Interface "wlan0";
MnRouterProbes 1;
MnHomeLink "wlan0" {
HomeAgentAddress 200X:5c0:8d03:1::1;
HomeAddress 200X:5c0:8d03:1::2/64;
}
UseMnHaIPsec disabled;
```

2.4 Configuring radvd on AR

When MN comes to a new network, it does a link-local address configuration, going to the next phase if that succeeds.

The next phase of autoconfiguration involves obtaining a Router Advertisement or determining that no routers are present. If routers are present, they will send Router Advertisements that specify what sort of autoconfiguration a host should do. If no routers are present, stateful autoconfiguration should be invoked.

We'll configure RADVD on AR's wireless interface.

Note: Check if the wireless card is up before start the radvd.

```
ifconfig wlan0 up
```

The radvd.conf file should contain this:

```
interface wlan0
{
    AdvSendAdvert on;
    AdvIntervalOpt on;

    MinRtrAdvInterval 3;
    MaxRtrAdvInterval 10;
    AdvHomeAgentFlag off;

    prefix 200X:5c0:8d03:3::/64
```

```
    {
        AdvOnLink on;
        AdvAutonomous on;
        AdvRouterAddr on;
    };
};
```

We then start it:

```
cp radvd.conf /etc/
cp radvd.conf /usr/local/etc/
radvd start
```

You should now be able to use radvdump to see that the radvd messages really are being sent periodically:

```
radvdump
```

Sample radvd messages:

```
Note: {Min,Max}RtrAdvInterval cannot be obtained with radvdump
        AdvCurHopLimit: 64
        AdvManagedFlag: off
        AdvOtherConfigFlag: off
        AdvHomeAgentFlag: off
        AdvReachableTime: 0
        AdvRetransTimer: 0
        AdvDefaultPreference: medium
        Prefix 200X:5c0:8d03:3::/64
        AdvValidLifetime: 2592000
        AdvPreferredLifetime: 604800
        AdvOnLink: on
        AdvAutonomous: on
        AdvRouterAddr: on
        AdvSourceLLAddress: 00 16 6F C7 AF 25
        AdvIntervalOpt:
        AdvInterval: 10
}
```

2.5 Configuring radvd on HA

To enable the MN to know when it's home, HA should also be sending out RAs. We should therefore enable RADVD on the HA as well. The radvd.conf file should contain:

```
interface wlan0
{
    AdvSendAdvert on;
    MaxRtrAdvInterval 3;
    MinRtrAdvInterval 1;
    AdvIntervalOpt off;
    AdvHomeAgentFlag on;
    HomeAgentLifetime 10000;
    HomeAgentPreference 20;
    AdvHomeAgentInfo on;
    prefix 200X:5c0:8d03:1::1/64
    {
```

```
        AdvRouterAddr on;
        AdvOnLink on;
        AdvAutonomous on;
        AdvPreferredLifetime 10000;
        AdvValidLifetime 12000;
    };
};
```

We then start it:

```
cp radvd.conf /etc/
cp radvd.conf /usr/local/etc/
radvd start
```

Also do a radvdump on HA to check whether radvd messages are beeing sent. You'll notice that MN (when home) generates a new address based on RADVD messages.

3 Starting MIPv6

Before starting IPV6, you should make sure the wireless card wlan0 is up.

Start Mobile IPv6 on HA.

```
ifconfig  wlan0 up
mip6d -c /etc/mip6d.conf
```

Then the message displayed in the screen is the following:

```
{
mip6d[4528]: MIPL Mobile IPv6 for Linux v2.0.2 started (Home Agent)
main: MIPL Mobile IPv6 for Linux started in debug mode, not detaching from
          terminal
conf_show:  config_file  = /etc/mip6d.conf
conf_show: mip6_entity = 2
conf_show: debug_level = 10
conf_show: PolicyModulePath = [internal]
conf_show: DefaultBindingAclPolicy = 0
conf_show: NonVolatileBindingCache = disabled
conf_show: KeyMngMobCapability = disabled
conf_show: UseMnHaIPsec = enabled
conf_show: MnMaxHaBindingLife = 262140
conf_show: MnMaxCnBindingLife = 420
conf_show: MnRouterProbes = 0
conf_show: MnRouterProbeTimeout = 0.000000
conf_show: InitialBindackTimeoutFirstReg = 1.500000
conf_show: InitialBindackTimeoutReReg = 1.000000
conf_show: UseCnBuAck = disabled
conf_show: DoRouteOptimizationMN = enabled
conf_show: MnUseAllInterfaces = disabled
conf_show: MnDiscardHaParamProb = disabled
conf_show: SendMobPfxSols = enabled
conf_show: OptimisticHandoff = disabled
conf_show: SendMobPfxAdvs = enabled
conf_show: SendUnsolMobPfxAdvs = enabled
conf_show: MaxMobPfxAdvInterval = 86400
conf_show: MinMobPfxAdvInterval = 600
conf_show: HaMaxBindingLife = 262140
```

```
conf_show: DoRouteOptimizationCN = enabled
xfrm_cn_init: Adding policies and states for CN
xfrm_ha_init: Adding policies and states for HA
ha_if_addr_setup: Joined anycast group 200X:5c0:8d03:1:fdff: ffff : ffff : fffe on iface
        5
}
```

Next start Mobile IPv6 on MN.

```
ifconfig wlan0 up
ifconfig ip6tnl0 up
ifconfig sit0 up
mip6d -c /etc/mip6d.conf
```

the message displayed in the screen is the following:

```
mip6d[5430]: MIPL Mobile IPv6 for Linux v2.0.2 started (Mobile Node)
main: MIPL Mobile IPv6 for Linux started in debug mode, not detaching from
        terminal
conf_show: config_file = /etc/mip6d.conf
conf_show: mip6_entity = 1
conf_show: debug_level = 10
conf_show: PolicyModulePath = [internal]
conf_show: DefaultBindingAclPolicy = 0
conf_show: NonVolatileBindingCache = disabled
conf_show: KeyMngMobCapability = disabled
conf_show: UseMnHaIPsec = enabled
conf_show: MnMaxHaBindingLife = 262140
conf_show: MnMaxCnBindingLife = 420
conf_show: MnRouterProbes = 0
conf_show: MnRouterProbeTimeout = 0.000000
conf_show: InitialBindackTimeoutFirstReg = 1.500000
conf_show: InitialBindackTimeoutReReg = 1.000000
conf_show: UseCnBuAck = disabled
conf_show: DoRouteOptimizationMN = enabled
conf_show: MnUseAllInterfaces = disabled
conf_show: MnDiscardHaParamProb = enabled
conf_show: SendMobPfxSols = enabled
conf_show: OptimisticHandoff = disabled
conf_show: SendMobPfxAdvs = enabled
conf_show: SendUnsolMobPfxAdvs = enabled
conf_show: MaxMobPfxAdvInterval = 86400
conf_show: MinMobPfxAdvInterval = 600
conf_show: HaMaxBindingLife = 262140
conf_show: DoRouteOptimizationCN = enabled
xfrm_cn_init: Adding policies and states for CN
xfrm_mn_init: Adding policies and states for MN
conf_home_addr_info: HoA address 200X:5c0:8d03:1:0:0:0:2
conf_home_addr_info: HA address 200X:5c0:8d03:1:0:0:0:1
__tunnel_add: created tunnel ip6tnl1 (6) from 200X:5c0:8d03:1:0:0:0:2 to 200X:5c0:8
        d03:1:0:0:0:1 user count 1
conf_home_addr_info: Home address 200X:5c0:8d03:1:0:0:0:2
flag_hoa: set HoA 200X:5c0:8d03:1:0:0:0:2/128 iif 6 flags 10 preferred_time
        4294967295 valid_time 4294967295
conf_home_addr_info: Added new home_addr_info successfully
__md_discover_router: discover link on iface eth1 (5)
md_change_default_router: add new router fe80:0:0:0:216:6 fff :fec7:5ba1 on interface
        eth1 (5)
```

```
mn_addr_do_dad: DAD succeeded!
mn_move: 1535
mn_move: in home net
}
```

Also when issue the ifconfig, you'll notice the tunnel (ip6tnl1) is up.

4 Doing some tests

We could try to use vlc to stream videos through mobile ipv6 protocol.

4.1 Movement detection

Generic movement detection uses Neighbor Unreachability Detection to detect when the default router is no longer bi-directionally reachable, in which case the mobile node must discover a new default router(usually on a new link).

To easily see whats going on, you should have one xterm window for each of these commands (on MN):

```
watch ifconfig wlan0
watch route -A inet6
tcpdump -i wlan0 -vv ip6 or proto ipv6
```

To "travel" to another net, you can issue the command on MN:

```
iwconfig wlan0 essid visitnet_groupX channel 3
```

The MN is then on the other wireless network, and since it is sending out "router solicitation" (multicast), our AR will respond with it's prefix. MN will then configure itself with at new IPv6 address with the received prefix and it's own MAC address. If you type ifconfig wlan0 you will see the new IPv6 address.

4.2 Real stream testing - smooth handover

To really get the feel on how mobile IP works, fire up vlc. Then do a "travel" and you can see an almost smooth handover.

Vlc on the CN act as the server Follow the figures 3 4 5 to setup the vlc.

Vlc on the MN act as the client The setup of client is simple, just as figure 6.

Switch between homenet and visitnet

Using the following command to change the network.

```
iwconfig wlan0 essid visitnet_groupX channel 3
iwconfig wlan0 essid homenet_groupX channel 1
```

You'll see that the video continue to display in the vlc at MN. That means the mobile ip is **o**k!

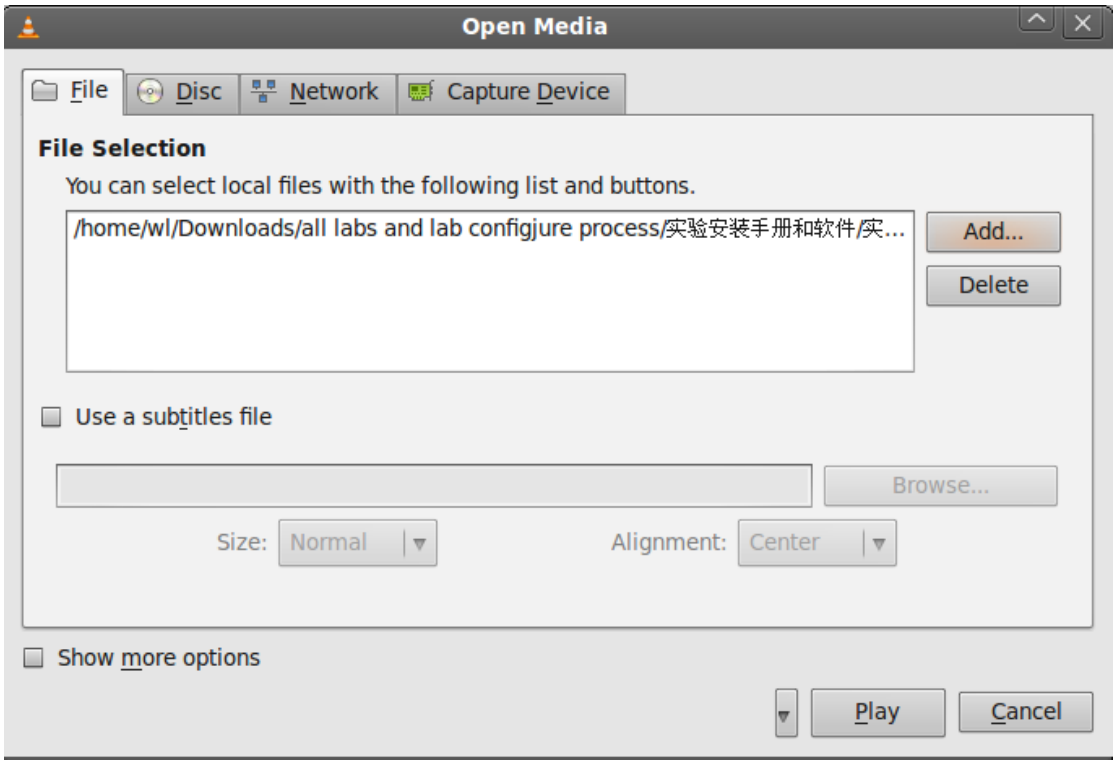

Figure 3: Vlc config on CN - Open the file

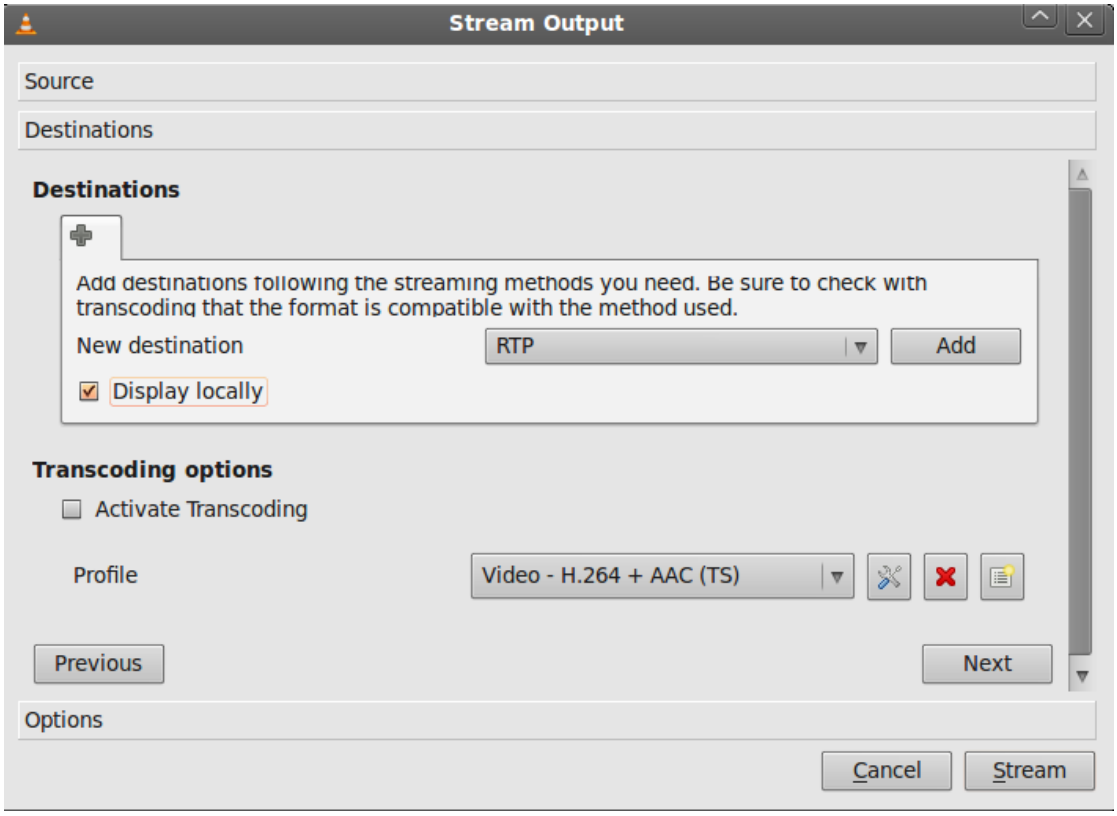

Figure 4: Vlc config on CN - Choose the package protocol

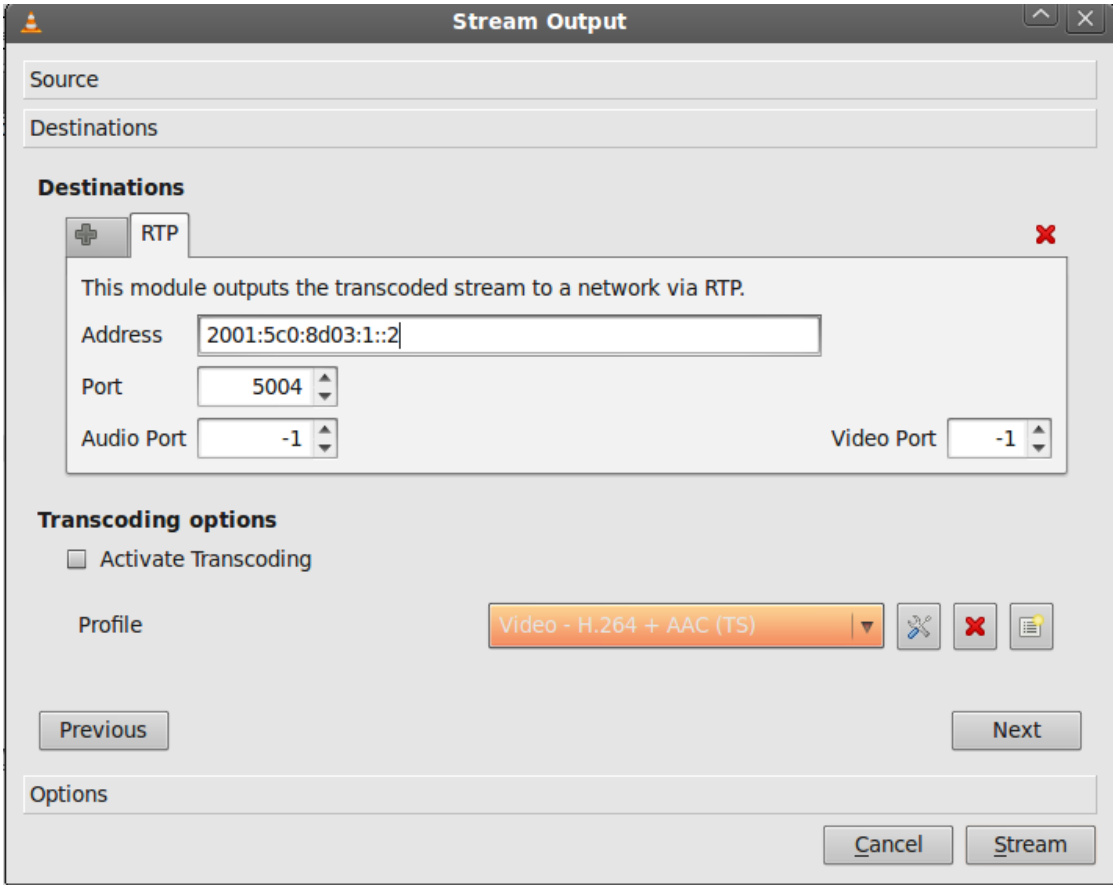

Figure 5: Vlc config on CN - Input the streaming ip address

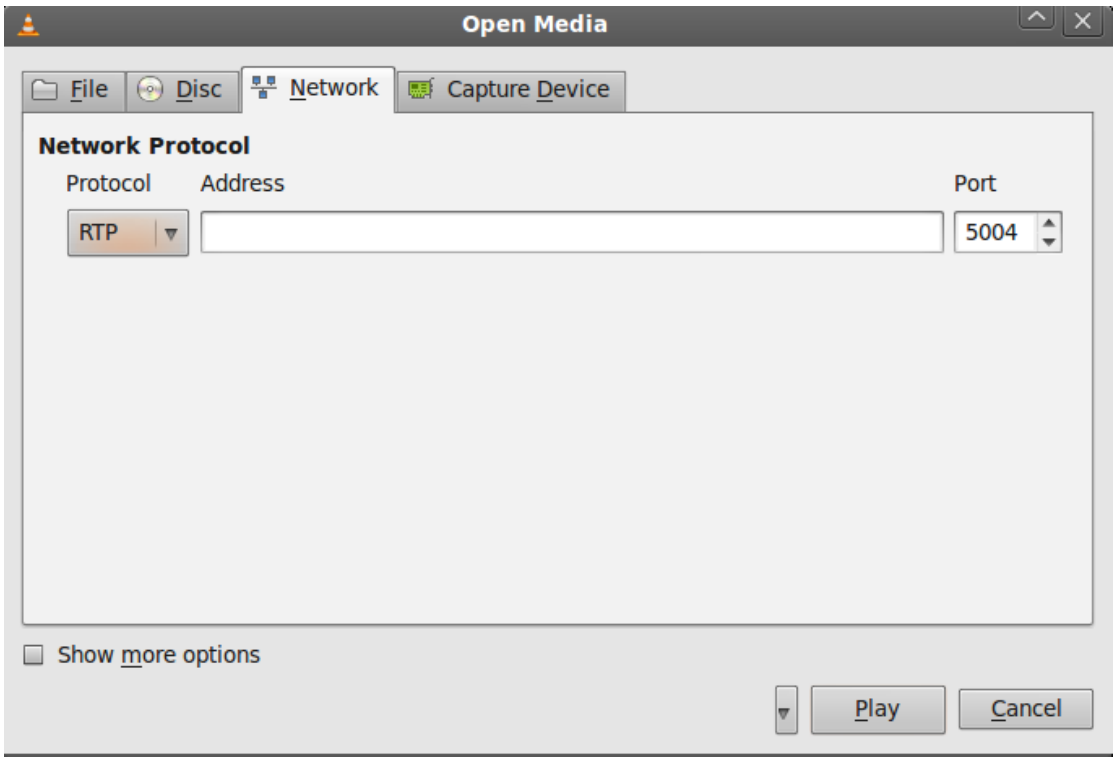

Figure 6: Vlc config on MN - Open the streaming video

5 Questions

1. What are the handoff and its process? Please show the handoff process with signaling diagram?
2. What is the difinition of handoff delay? What is the rough value of the handoff delay in our experiment?
3. What is the IPV6 address of MN after roaming to visitnet?
4. How to set a default gateway in IPv6?
5. If MN has travelled through several visited LAN, and then returning home. Does the interface still have all the autogenerated IPv6 addresses from all the visited networks?

References

[1] `http://tldp.org/HOWTO/Mobile-IPv6-HOWTO/index.html`

[2] `http://umip.org/docs/index.html`

A Compile a kernel for Mobile Ipv6

In order to complete this lab, you need a mobility-ready kernel as well as an userland daemon. The kernel will be retrieved from the official linux kernel repository, and recompiled with the needed options.

The following help you to setup the testbed for lab4 if you want to complete lab4 on your own desktops.

You need to compile a kernel for your system. Hopefully, the IPv6 mobility support has been integrated into the kernel sources since version 2.6.26, so no patches are needed anymore on recent kernel versions.

Download the kernel sources (version >= 2.6.26) from `http://www.kernel.org/`. For example with a 2.6.35 kernel:

```
$ cd /usr/src/
$ wget http://www.kernel.org/pub/linux/kernel/v2.6/linux-2.6.35.tar.gz
```

Uncompress the sources, and create a symbolic link to the kernel sources:

```
$ tar zxf linux-2.6.35.tar.gz
$ rm linux-2.6.35.tar.gz*
$ ln -s /usr/src/linux-2.6.35/include /usr/src/linux
$ cd linux-2.6.35/
```

Edit the kernel configuration file to turn on the mobility options.

```
$ make menuconfig
```

To enable the mobility features, you need to turn on the following options in the kernel:

```
General setup
--> Prompt for development and/or incomplete code/drivers [
        CONFIG_EXPERIMENTAL]
--> System V IPC [CONFIG_SYSVIPC]

Networking support [CONFIG_NET]
--> Networking options
    --> Transformation user configuration interface [CONFIG_XFRM_USER]
    --> Transformation sub policy support [CONFIG_XFRM_SUB_POLICY]
    --> Transformation migrate database [CONFIG_XFRM_MIGRATE]
    --> PF_KEY sockets [CONFIG_NET_KEY]
    --> PF_KEY MIGRATE [CONFIG_NET_KEY_MIGRATE]
    --> TCP/IP networking [CONFIG_INET]
    --> The IPv6 protocol [CONFIG_IPV6]
        --> IPv6: AH transformation [CONFIG_INET6_AH]
        --> IPv6: ESP transformation [CONFIG_INET6_ESP]
        --> IPv6: IPComp transformation [CONFIG_INET6_IPCOMP]
        --> IPv6: Mobility [CONFIG_IPV6_MIP6]
        --> IPv6: IPsec transport mode [
         CONFIG_INET6_XFRM_MODE_TRANSPORT]
        --> IPv6: IPsec tunnel mode [CONFIG_INET6_XFRM_MODE_TUNNEL]
        --> IPv6: MIPv6 route optimization mode [
         CONFIG_INET6_XFRM_MODE_ROUTEOPTIMIZATION]
        --> IPv6: IPv6-in-IPv6 tunnel [CONFIG_IPV6_TUNNEL]
        --> IPv6: Multiple Routing Tables [CONFIG_IPV6_MULTIPLE_TABLES]
        --> IPv6: source address based routing [CONFIG_IPV6_SUBTREES]
```

```
File systems
--> Pseudo filesystems
    --> /proc file system support [CONFIG_PROC_FS]
```

Now proceed to the kernel compilation and installation:

```
$ make
$ make modules
$ make modules_install
$ make install
```

To ease the kernel patching process and for future use, we create the deb packages for the newly compiled kernel.

```
$ make-kpkg --initrd --append-to-version=-custom kernel_image
        kernel_headers
```

Verify your bootloader configuration and reboot on your new kernel.

```
$ sudo update-grub
```

You could see your current kernel by typing:

```
$ uname -r
```

B Building the UMIP Userland from the Sources

UMIP.org aims at gathering resources on the Mobile IPv6 and NEMO Basic Support stack UMIP, and proposes new UMIP releases. UMIP would be the best userland daemon for our lab.

First, we need additional packages to be able to compile and use UMIP. Be sure to have the following packages: autoconf, automake, bison, flex, libssl-dev, indent, ipsec-tools, radvd on your system.

```
$ apt-get install autoconf automake bison flex libssl-dev indent ipsec-tools radvd
```

Then, retrieve the source code from our repository:

```
$ cd /usr/src/
$ git clone http://www.umip.org/git/umip.git
$ cd umip/
```

Provided that you have built your kernel as explained in the previous section, you can compile and install UMIP with:

```
$ autoreconf -i
$ CPPFLAGS='-isystem /usr/src/linux/include/' ./configure --enable-vt
$ make
$ make install
```

The –enable-vt option enables a virtual terminal, which can be useful to retrieve the binding cache or binding update list information on the Home Agent or the Mobile Node.

C Vlc Setup

Download the latest vlc package vlc-1.1.0.tar.bz2 and extract it.

Get to the path:

```
$ cd Downloads/vlc-1.1.0/
```

Configure and install:

```
$ ./configure
```

Any error appears, it may tell you to add command to avoid it. For example:

```
configure: error: Could not find lua. Lua is needed for some interfaces (rc, telnet
          , http) as well as many other custom scripts. Use --disable-lua to ignore
          this error.
```

So, you simply add the command:

```
$ ./configure --disable-lua
```

After configuration complete, you could install it by:

```
$ make
$ make install
```