WIDE Technical-Report in 2006

The E-Bicycle Demonstration Setup on Tour de France 2006

wide-tr-nautilus6-ebicycle-tour-de-france-00.pdf

WIDE Project : http://www.wide.ad.jp/

If you have any comments on this document, please contact to ad@wide.ad.jp

The E-Bicycle Demonstration Setup on Tour de France 2006

Romain KUNTZ

Nautilus6 Working Group
E-mail: kuntz@sfc.wide.ad.jp
September 25th 2006

Abstract. In this memo, we summarize the E-bicycle demonstration performed by the Nautilus6 Working Group during the Tour de France bicycle event in July 2006. We first overview the scenario and equipments involved in the demonstration, and then explain how to setup the whole platform from scratch.

1 Introduction

The Tour de France [1] is a 3-weeks long professional bicycle race that takes place all over around France in July. This is the third most important sport event in the world after the Olympic Games and the football worldcup (from a coverage point of view).

The Nautilus6 project [2] has organized an E-bicycle [3] demonstration on Friday, July 21st, Saturday, July 22nd and Sunday, July 23rd, near the last step of the Tour de France race. This event has been setup in collaboration between people from WIDE Project [4] (Japan), ENST Rennes [5] (France), ULP Strasbourg [6] (France), and INRIA Rocquencourt [7] (France).

This demonstration was a great opportunity to demonstrate IPv6 Mobility to the public and all Nautilus6 partners in France and Japan. Anyone interested in this demonstration was able to follow it in Live or remotely on the Internet.

This memo explains the technical details of the demonstration, for people willing to setup the same kind of events. We will overview in the next section the demonstration scenario and equipments, then explain how to configure each entity involved in the event.

2 The Demonstration Scenario

The E-bicycle is a bicycle that embeds an IPv6 Personal Area Network (PAN). This PAN is made of several IPv6 devices and is connected to the Internet through several access technologies.

Thanks to IPv6 and mobility protocols such as NEMO Basic Support [8], any people in the world is able to follow the whereabouts of the E-Bicycle near the Tour de France event, while the E-Bicycle is on the move. This includes pictures of the surroundings, sound, video, etc.

For this demonstration, the E-Bicycle is composed of (Fig.1):

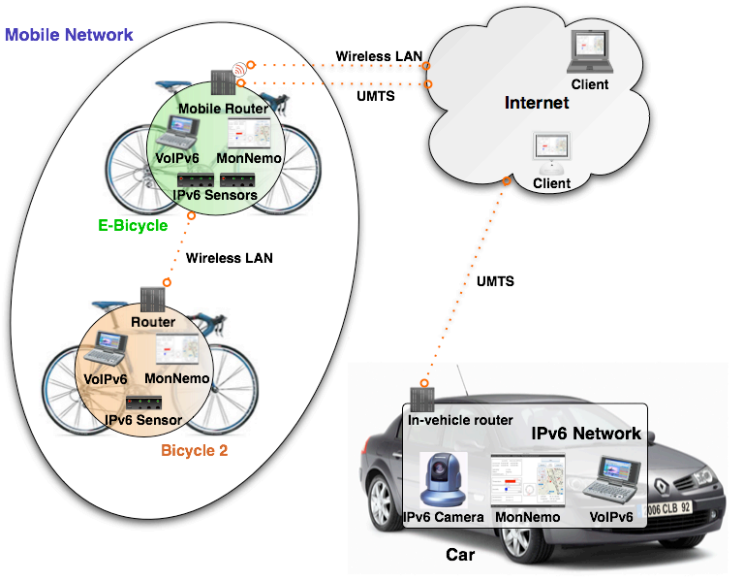

Fig. 1. The Demonstration Scenario

- A multihomed Linux Mobile Router using NEPL [9] (NEMO Platform for Linux) and Multiple Care-of Addresses Registration [10] (MCoA). The Mobile Router is connected to the Internet through a cellular technology (UMTS) and Wireless LAN. MCoA allows to use both access technologies at the same time or to divert the traffic to one of them in case of failure of the other one,
- Several IPv6 sensors (temperature and humidity sensor, direction sensor, GPS sensor) which can be remotely queried using SNMPv1 [11].

Another bicycle takes benefit from the E-Bicycle's Internet access, this bicycle being connected to the E-Bicycle's PAN using wireless LAN.

A vehicle is following both bicycles. This vehicle is also connected to the Internet using cellular technology (UMTS). An IPv6 Camera is located inside the car, its flow being sent to the Internet using XCAST6 [12]. An uninterrupted audio session using Voice-over-IPv6 is maintained between the bicycles and the vehicle. A laptop displays the bicycles' sensors values and some pictures from the IPv6 camera using the MonNemo software [13], that allows the public to easily follow remotely the whereabouts of the bicycles.

People in France and Japan could attend the demonstration remotely with:

- The MonNemo application (Fig.2), that allowed to see the E-Bicycle and the vehicle position on a map, some pictures of the surrounding from the IPv6 camera, and the environment conditions by displaying the IPv6 sensors values. A chat also allowed to keep in touch between the vehicle and the remote attendees.

– The IPv6 camera video flow (Fig.3), that was sent to several client over the Internet using XCAST6 (in cooperation with the WIDE XCAST Working Group [12]). XCAST6 is a new and flexible scheme for multicast, suitable for many to many group communication for relatively small groups (it is not designed for a large number of members).
– Various statistics about the number of attendees, the reachability of the system, etc. (Fig.4)

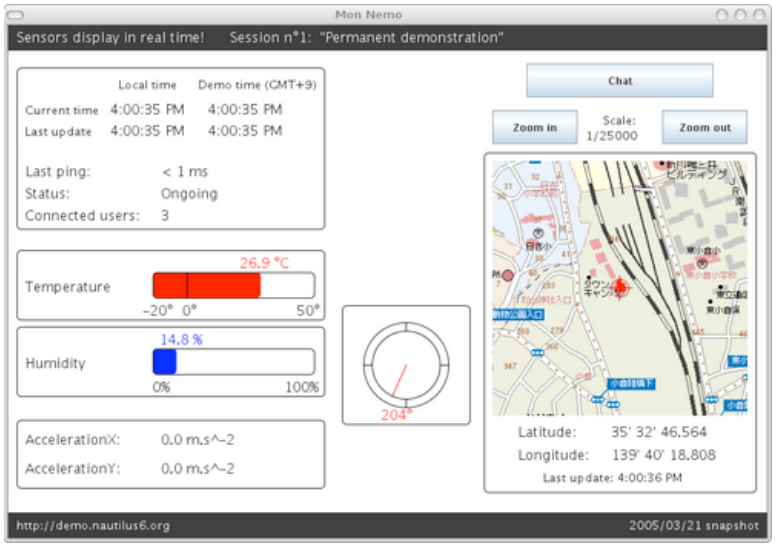

Fig. 2. The MonNemo application

3 The Demonstration Setup

3.1 Network Topology

The figure 5 is a sample network topology that can be used to setup the services described in section 3.2.

3.2 Servers and Home Agent

Several services need to be installed in your network:

– A Home Agent (HA), where your Mobile Routers can register,
– A L2TP [14] server, in order to be able to create IPv6-over-IPv4 tunnels. This service is very useful, for example when using cellular cards on your Mobile router, as cellular providers only provide IPv4 access,

Fig. 3. The XCAST flow is sent using VIC

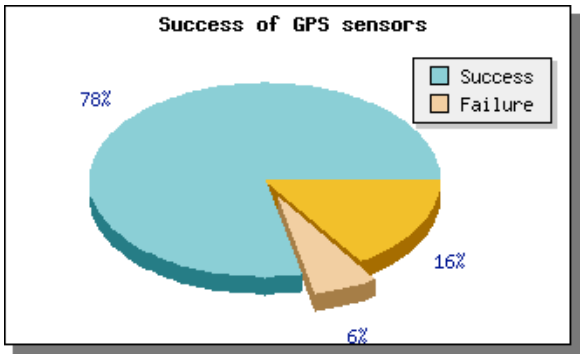

Fig. 4. The GPS sensor reachability

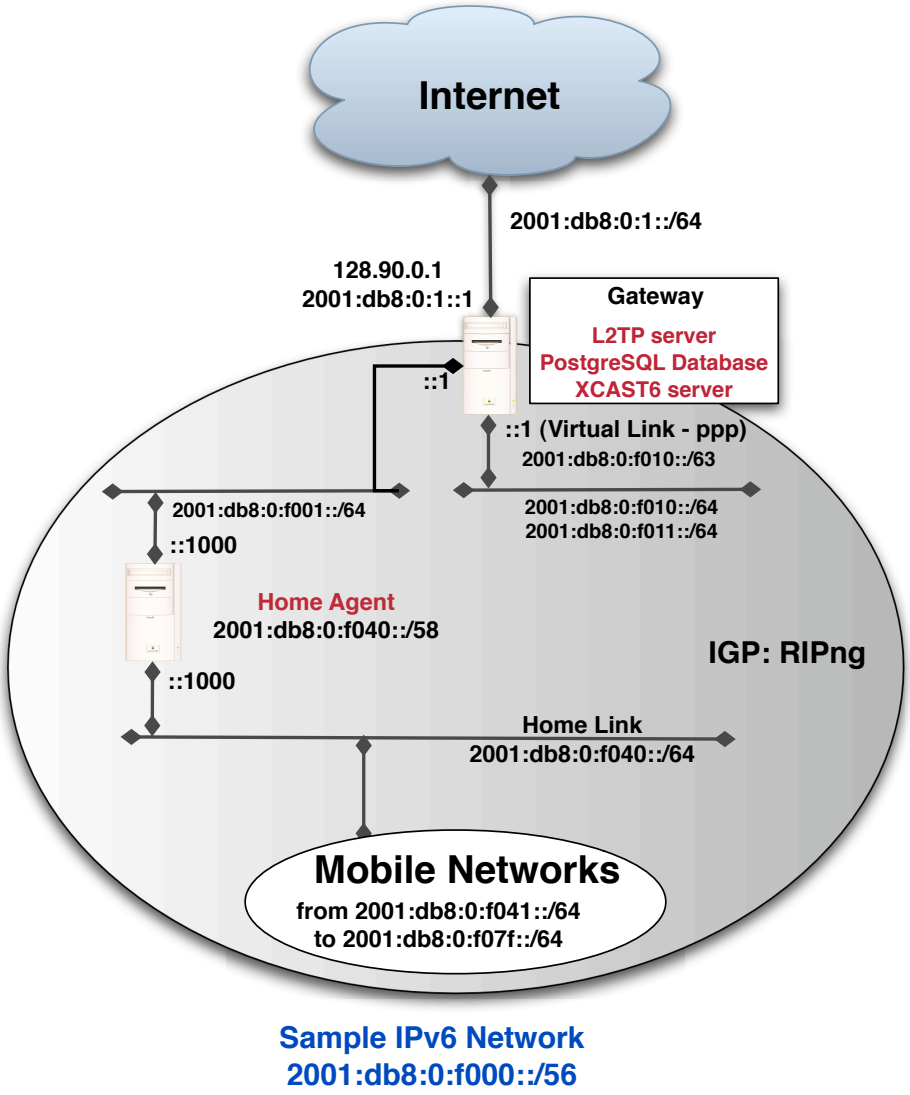

Fig. 5. The Network Topology

– A database server (PostgreSQL [15]) that will be useful for the MonNemo monitoring software (see section 3.6). This database will be used to store various informations about your mobile networks,
– An XCAST group management server, where XCAST clients will be able to register to receive the XCAST flow.

Some of those services can be setup on the same machine. But we advice to separate the Home Agent from all other services. L2TP server, XCAST group management server, and the database server can be installed on the same machine.

Home Agent Setup The Home Agent is using the NEPL implementation [9] with Multiple Care-of Addresses registration support [10]. Those implementations are freely available. The NEPL Howto [16] explains in detail how to setup a Home Agent with Multiple Care-of Addresses registration support (especially section 6).

L2TP server The L2TP server uses the Roaring Penguin implementation [17]. Sample configuration files to setup a L2TP server are provided in Appendix A. Be sure to have the **ppp_synctty** and **n_hdlc** kernel modules compiled and loaded before starting your L2TP server.

Database server The database server is a **postgreSQL** [15] server. It supports IPv6, which is mandatory when we use the MonNemo application. PostgreSQL is available as a software package for most of the Linux distributions, and thus should be easy to install and setup. For easy management of the database via a web interface, **phppgadmin** can also be installed.

The database structure used for the MonNemo software evolves from time to time, according to the features we add to the software. If you would like to setup a database server, please contact us (see section 4) to get a dump of the database structure.

XCAST6 group management server The XCAST6 group management server allows XCAST6 users to register their IPv6 addresses to the server and retrieve all participants' addresses for an XCAST6 session. This server is mandatory in the XCAST deployment.

The WIDE XCAST working group provides all the necessary information to setup an group management server. XCAST6 for Linux[1] is shipped with a kernel patch (for kernels 2.6.10, 2.6.12, and 2.6.16) and some userland libraries and softwares for the Fedora Core (from 2 to 5) Operating System. A README[2] explains how to setup the system.

XCAST6 is also available for *BSD operating system. You can get more information on the WIDE XCAST Working Group webpage [12].

Nautilus6 also provides Debian packages for XCAST6. You can download them from Nautilus6's software webpage [18].

[1] XCAST6 for Linux: http://xcast-linux.sourceforge.net/0.2-rc1/
[2] http://xcast-linux.sourceforge.net/0.2-rc1/README.txt

3.3 The E-Bicycle

The E-Bicycle embeds a Mobile Router to get Internet Access while on the move. This Mobile Router is multihomed, with a a cellular card and a WirelessLAN device. L2TP is used in order to get IPv6 connectivity on the cellular interface.

The Mobile Router The Mobile Router (MR) runs on a 2.6.15 GNU/Linux kernel with Mobility and Multiple Care-of Addresses (MCoA) registration support. The mobility management is handled by NEPL (NEMO Platform for Linux) [9], patched with the MCoA support provided by Nautilus6 [10]. The NEPL Howto [16] explains in detail how to setup and configure a Mobile Router from scratch with MCoA.

The Mobile Router is a SOEKRIS net4521 [19] board. The operating system is stored on a compact flash memory card. The Voyage Linux [20] distribution is especially designed for the soekris board, and allows to have a functional Debian-based operating system on the Soekris board without much effort. The CF image can be easily customed for our needs, by replacing the current kernel with a mobility-enabled kernel, as explained on their website[3]

UMTS Connectivity setup If you intend to get internet connectivity with a 3G cellular card, the following links may help you:

- The **Nozomi driver**[4] for Linux may support your 3G card,
- The **comgt**[5] utility software will allow you, among others, to set the PIN code of the SIM card,
- Once the card is properly initialized, the **wvdial** software [21] allows to establish the connection with the provider. Samples configuration files for **wvdial** are provided in Appendix B.

L2TP Client setup Using L2TP on the Mobile Router allows to get IPv6 connectivity over the IPv4-only cellular network.

The L2TP client on the Mobile Router uses the Roaring Penguin implementation [17]. Sample configuration files to setup a L2TP client are provided in Appendix C. Be sure to have the **ppp_synctty** and **n_hdlc** kernel modules compiled and loaded before starting your L2TP client.

The Access Point In order to provide IPv6 connectivity to the companion bicycle, an access point is configured on the E-Bicycle. The soekris embeds an **Atheros Mini-PCI SL-5354MP ARIES** card, which can be used as an access point thanks to the **MadWifi drivers** [22]. The Mobile Router must be configured to advertise a different Mobile Network Prefix via the Access Point, thus it is necessary to configure the MR and the HA to allow the registration of several Mobile Network Prefixes on the MR.

[3] http://wiki.voyage.hk/dokuwiki/doku.php?id=voyage_kernel
[4] http://www.pharscape.org/3G/nozomi
[5] http://www.pharscape.org/3G/comgt/

3.4 The Companion Bicycle

This bicycle takes advantage from the connectivity of the E-Bicycle's mobile network. It embeds a router and a small network which is actually a sub-network of the E-bicycle's network.

The Router This router is an usual Linux computer configured as a router, that connects to the E-Bicycle's Access Point using WirelessLAN. The prefix advertised by this router in its sub-network is one of the Mobile Network Prefix delegated by the HA to the MR.

3.5 The In-vehicle Network

The in-vehicle network embeds a Mobile Router for mobility management, an access point to provides Wireless IPv6 access to the E-Bicycle, an IPv6 Camera and an XCAST client.

The Mobile Router The in-vehicle Mobile Router connects to the Internet with a cellular card, getting IPv6 access thanks to a L2TP tunnel, and running NEPL for the mobility management. All the information to setup such a Mobile Router can be found in the E-Bicycle section (3.3).

The Access Point In order to provide a wireless IPv6 access to the E-Bicycle, thus allowing it to have several connectivity to the Internet, an access point is configured in the in-vehicle network.

The IPv6 Camera Panasonic[6] sells IPv6 camera that can be remotely accessed with a web browser. Any people connected to the IPv6 Internet in the world is thus able to watch the surroundings of the vehicle. The video stream from the camera is also sent using XCAST6 as explained in the next section.

The XCAST6 client A computer in the vehicle network is configured with XCAST6. It must be configured the same way as the group management server (see section 3.2), but without the need to setup an apache server and the **xcgroupsrv.cgi** script file.

Once the client has subscribed to a group on the XCAST6 group management server (thanks to the **xcgroup** command), the video from the IPv6 camera can be sent to that group using XCAST6. The **VIC** software (available from the XCAST6 packages) allows to send the top left corner of the screen as a video flow to the group. We thus put a web browser with the camera image in that corner, and use VIC to send this area as an XCAST6 video to the group.

[6] http://www.panasonic.com

3.6 Applications

In this section, we present the monitoring and Voice-over-IP (VoIP) softwares used during the demonstration.

MonNemo The MonNemo [13] (Monitoring NEMO) software allows any people connected to the Internet to remotely monitor a Mobile Network. MonNemo displays real-time information from different kind of sensors, a map with the curent location of the NEMO (Fig.2), pictures from an IPv6 camera. It also integrates a chat system (Fig.6).

Used as a server, MonNemo retrieves all the information from the sensors and store them in a postgreSQL database (see section 3.2).

As a client, the MonNemo software can be used either as a standalone Java application, or be accessible as a Java applet via a web browser. In both case it connects to the postgreSQL database to retrieve all the information to display in the application.

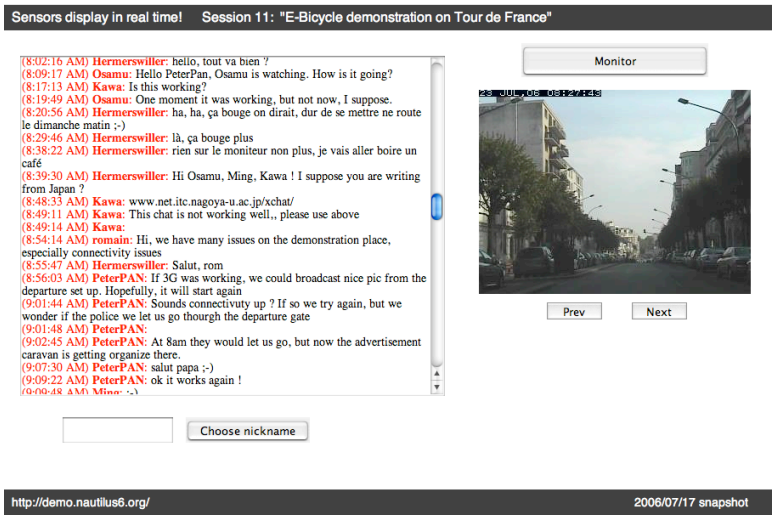

Fig. 6. MonNemo: The chat system and pictures from the IPv6 camera

Ekiga Ekiga [23] (formely known as GnomeMeeting) is an open source VoIP and video conferencing application for Linux. Ekiga is used on the E-Bicycle, the companion bicycle and in the vehicle in order to have an uninterrupted audio session between all peers.

In order to have three different participants to the same audio session, an openMCU server [24] must be installed, for example on a computer in the vehicle's network.

4 Support

Nautilus6 provides to public mailing lists two inform and help the users of our technologies. Details can be found on Nautilus6 website[7].

Acknowledgments

The demonstration has been prepared from July 5th to 20th 2006, at INRIA in Rocquencourt (France). This demonstration is the result of a collaboration between people from several organizations:

- The University of Tokyo (Japan) and Keio University (Japan), that took care of the IPv6 infrastructure and the E-Bicycle setup,
- ENST Rennes (France), that took care of the second bicycle and the in-vehicle network,
- ULP Strasbourg (France), that improved the MonNemo application,
- INRIA Rocquencourt (France), that provided the equipments and the office to setup the demonstration.

References

1. Le Tour de France, a professional bicycle race all around France. Web page, as of August 2006. http://www.letour.fr/indexus.html.
2. Nautilus6 Working Group, WIDE Project. Web page, as of August 2006. http://www.nautilus6.org.
3. Thierry Ernst, Romain Kuntz, and Francois Leiber. A Live Light-Weight IPv6 Demonstration Platform for ITS Usages. In *5th International Conference on ITS Telecommunications (ITST)*, Brest, France, June 2005.
4. WIDE Project (Widely Integrated Distributed Environment). Web page, as of August 2006. http://www.wide.ad.jp.
5. ENST (cole Nationale Suprieure des Tlcommunications) Bretagne. Web page, as of August 2006. http://international.enst-bretagne.fr/welcome/.
6. ULP (Louis Pasteur University), Network and Protocol Team. Web page, as of August 2006. http://www-r2.u-strasbg.fr/english/index_en.php.
7. INRIA (Institut National de Recherche en Informatique et en Automatique), IMARA team. Web page, as of August 2006. http://www.inria.fr/recherche/equipes/imara.en.html.
8. Vijay Devarapalli, Ryuji Wakikawa, Alexandru Petrescu, and Pascal Thubert. Network Mobility (NEMO) Basic Support Protocol. Request For Comments 3963, IETF, January 2005.
9. Go-Core in co-operation with the Nautilus6/WIDE project. NEPL, NEMO Platform for Linux. Web page, as of August 2006. http://www.mobile-ipv6.org.
10. Romain Kuntz and Jean Lorchat. Multiple Care-of Addresses Registration for NEPL. Web page, as of August 2006. http://software.nautilus6.org/MCoA/.
11. Jeffrey D. Case, Mark Fedor, Martin Lee Schoffstall, and James R. Davin. A Simple Network Management Protocol (SNMP). Request For Comments 1157, IETF, May 1990.
12. The XCAST Working Group. Web page, as of August 2006. http://www.xcast.jp.
13. MonNemo, a NEMO Monitoring application. Web page, as of August 2006. http://software.nautilus6.org/MonNemo/.

[7] http://www.nautilus6.org/ml.php

14. Gurdeep Singh Pall, Bill Palter, Allan Rubens, W. Mark Townsley, Andrew J. Valencia, and Glen Zorn. Layer Two Tunneling Protocol (L2TP). Request For Comments 2661, IETF, August 1999.
15. PostgreSQL, an open source database. Web page, as of August 2006. http://www.postgresql.org.
16. NEPL (NEMO Platform for Linux) HOWTO. Web page, as of August 2006. http://www.nautilus6.org/doc/nepl-howto/.
17. RP-L2TP, a user-space implementation of L2TP for Linux. Web page, as of August 2006. http://sourceforge.net/projects/rp-l2tp/.
18. Nautilus6 softwares repository. Web page, as of August 2006. http://software.nautilus6.org.
19. Soekris Engineering. Web page, as of August 2006. http://www.soekris.com.
20. Voyage Linux, a Debian-based distribution for x86-based embedded platform. Web page, as of August 2006. http://www.voyage.hk/software/voyage.html.
21. WvDial makes it easy to connect your Linux workstation to the Internet. Web page, as of August 2006. http://open.nit.ca/wiki/?WvDial.
22. MadWifi, a Linux kernel device driver for Wireless LAN chipsets from Atheros. Web page, as of August 2006. http://madwifi.org.
23. Ekiga, an open source VoIP and video conferencing application. Web page, as of August 2006. http://www.ekiga.org.
24. The OpenH323 project, an Open Source implementation of the ITU-T H.323 teleconferencing protocol. Web page, as of August 2006. http://www.openh323.org.

APPENDIX

A L2TP server configuration files

The L2TP configuration file is **/etc/l2tp/l2tp.conf**. Read the comments within the files to configure properly your server.

```
# /etc/l2tp/l2tp.conf
# Global section
global

# Load handlers
load-handler "sync-pppd.so"
load-handler "cmd.so"

# Bind address
listen-port 1701

# Sync-pppd handler configuration.
section sync-pppd
lns-pppd-opts "file /etc/ppp/options.l2tp"

# Peer section
section peer
hostname l2tp
peer 0.0.0.0
mask 0
port 1701
lns-handler sync-pppd
hide-avps no
retain-tunnel 1

# Configure the cmd handler. You MUST have a "section cmd"
# line even if you don't set any options.
section cmd
# EOF
```

The **/etc/ppp/options.l2tp** file configures various L2TP options for ppp.

```
# /etc/ppp/options.l2tp
noauth
show-password
noccp
noaccomp
nobsdcomp
nopcomp
```

```
noip
-defaultroute
ipv6 ::1,::2
ipparam l2tp
# EOF
```

The **/etc/ppp/ipv6-up** script must also be updated to send Router Advertisements over the L2TP tunnels. This will allow to trigger the movement event for the mobility stack on the Mobile Router. You need to reserve a few /64 IPv6 prefixes for this operation. Be sure to replace the addresses in the file with yours, and that the prefixes you use are routable to your L2TP server.

```
#!/bin/sh
# /etc/ppp/ipv6-up

# Extract calling arguments
PPP_IFACE="$1"
PPP_TTY="$2"
PPP_SPEED="$3"
PPP_LOCAL="$4"
PPP_REMOTE="$5"
PPP_IPPARAM="$6"
export PPP_IFACE PPP_TTY PPP_SPEED PPP_LOCAL PPP_REMOTE \
       PPP_IPPARAM

# radvd daemon
PATH=/sbin:/bin:/usr/sbin:/usr/bin
DAEMON=/usr/sbin/radvd
# Be sure to have a radvd.conf file for
# each of your PPP interface.
CFGFILE=/etc/radvd.conf-${PPP_IFACE}
PIDFILE=/var/run/radvd-${PPP_IFACE}.pid
DEBUGLEVEL=4


# Duplicate the entries according to the number of
# host you wish to support. Replace the IPv6 prefixes
# and addresses with your address space.
case "${PPP_IFACE}" in
        ppp0)
        NEW_ROUTE="2001:db8:0:f010::/64"
        V6ADDR="2001:db8:0:f010::1/64"
        ;;
        ppp1)
        NEW_ROUTE="2001:db8:0:f011::/64"
        V6ADDR="2001:db8:0:f011::1/64"
```

```
        ;;
        *)
        logger -t ipv6-up "${PPP_IFACE} not matched"
        ;;
esac

if [ ! -x $DAEMON ]; then
  logger -t ipv6-up "Daemon radvd not found. exiting."
  exit 1
fi

# IPv6 forwarding must be enabled for radvd to work
if test $(cat /proc/sys/net/ipv6/conf/all/forwarding) -eq 0;
then
  logger -t ipv6-up "Activating IPv6 forwarding for radvd."
  echo 1 > /proc/sys/net/ipv6/conf/all/forwarding
fi

if [ ! -e $CFGFILE ]; then
  logger -t ipv6-up "${CFGFILE} does NOT exist."
  exit 1
fi

if start-stop-daemon --oknodo --start --pidfile ${PIDFILE} \
   --background --exec ${DAEMON} -- \
   -d ${DEBUGLEVEL} -C ${CFGFILE} -p ${PIDFILE}; then
   logger -t ipv6-up "radvd started on ${PPP_IFACE}."
else
   logger -t ipv6-up "radvd failed starting on ${PPP_IFACE}."
   exit 1
fi

if /sbin/ifconfig ${PPP_IFACE} add ${V6ADDR} &>/dev/null;
then
  logger -t ipv6-up "Added address ${V6ADDR} to ${PPP_IFACE}."
else
  logger -t ipv6-up "Adding ${V6ADDR} to ${PPP_IFACE} failed."
fi
# EOF
```

The **/etc/ppp/ipv6-down** script must also be updated to stop sending Router Advertisements when the PPP interface becomes down.

```
#!/bin/sh
# /etc/ppp/ipv6-down
```

```
# Extract calling arguments
PPP_IFACE="$1"
PPP_TTY="$2"
PPP_SPEED="$3"
PPP_LOCAL="$4"
PPP_REMOTE="$5"
PPP_IPPARAM="$6"
export PPP_IFACE PPP_TTY PPP_SPEED PPP_LOCAL PPP_REMOTE \
       PPP_IPPARAM

# radvd daemon
PATH=/sbin:/bin:/usr/sbin:/usr/bin
DAEMON=/usr/sbin/radvd
PIDFILE=/var/run/radvd-${PPP_IFACE}.pid

start-stop-daemon --stop --pidfile ${PIDFILE} \
                  --exec ${DAEMON}
rm -f ${PIDFILE}
logger -t ipv6-down "radvd on ${PPP_IFACE} stopped"
# EOF
```

Each PPP interface must have an **radvd.conf** configuration file. The configuration file for ppp0 will be **/etc/radvd.conf-ppp0**. A sample configuration would be (change the IPv6 prefix with the same one as in **/etc/ppp/ipv6-up**):

```
# /etc/radvd.conf-ppp0
interface ppp0
{
        AdvSendAdvert on;
        MinRtrAdvInterval 100;
        MaxRtrAdvInterval 300;
        AdvIntervalOpt on;
        prefix 2001:db8:0:f010::/64
        {
                AdvOnLink on;
                AdvAutonomous on;
                AdvRouterAddr on;
        };
};
# EOF
```

B wvdial configuration files

The **wvdial.conf** file is used to configure the dialing parameters. Here is a sample configuration file. Check the **wvdial.conf** manpage to tune the file according to your configuration.

```
# /etc/wvdial.conf
[Dialer Defaults]
Baud = 115200
Modem = /dev/noz0
Dial Command = ATDT
Carrier Check = yes
Stupid Mode = yes

[Dialer 3G]
Init = ATZ
Init1 = AT+CMEE=1
Init2 = AT+CGDCONT=1,"IP","username"
New PPPD = yes
Phone = *99#
Username = "username"
Password = "password"
# EOF
```

Some options must also be set for **ppp** in the **/etc/ppp/peers/wvdial** file:

```
noauth
name wvdial
usepeerdns
```

wvdial can then be executed with:

```
# wvdial 3G
```

C L2TP client configuration files

The **/etc/l2tp/l2tp.conf** allows to configure the L2TP client. Here is a sample configuration file. Replace the peer address with your L2TP server's address.

```
# /etc/l2tp/l2tp.conf
# Global section (by default, we start in global mode)
global

# Load handlers
load-handler "sync-pppd.so"
load-handler "cmd.so"

# Bind address
listen-port 1701

# Sync-pppd handler configuration
section sync-pppd
```

```
lac-pppd-opts "call l2tp"

# Peer section
# Replace the peer address with the L2TP server's address
section peer
peer 128.90.0.1
port 1701
lac-handler sync-pppd
lns-handler sync-pppd
hide-avps no

# Configure the cmd handler. You MUST have a
# "section cmd" line even if you don't set any options.
section cmd
# EOF
```

The **/etc/l2tp/l2tp-secrets** file includes login and password when using authentication methods between peers. We do not use such feature in our example, thus the file should be as follow:

```
# /etc/l2tp/l2tp-secrets
""       *       ""       *
# EOF
```

Some options must also be set for **ppp** in the **/etc/ppp/peers/l2tp** file:

```
# /etc/ppp/peers/l2tp
noauth
unit 1
+ipv6
ipv6cp-accept-local 1
noip
ipparam "l2tp"
lcp-echo-interval 60
lcp-echo-failure 4

debug
kdebug 1
# EOF
```

We can now setup the L2TP tunnel on the client. We first execute **l2tpd** in foreground (**-f**) with debug messages (**-d 65535**):

```
# l2tpd -d 65535 -f &
```

Then we setup a l2tp tunnel (replace the IPv4 address with your L2TP server's address):

```
# l2tp-control "start-session 128.90.0.1"
```