Title:	Document Version:
Deliverable D4.7 **Head End Router, Home Gateway and Set Top Box delivery for 2nd year trials**	1.4

Project Number:	Project Acronym:	Project Title:
IST-2001-37613	6POWER	IPv6, QoS & Power Line Integration

Contractual Delivery Date:	Actual Delivery Date:	Deliverable Type* - Security**:
30/11/2004	07/11/2004	R – PU

* Type:	P - Prototype, R - Report, D - Demonstrator, O - Other
** Security Class:	PU- Public, PP – Restricted to other programme participants (including the Commission), RE – Restricted to a group defined by the consortium (including the Commission), CO – Confidential, only for members of the consortium (including the Commission)

Responsible and Editor/Author:	Organization:	Contributing WP:
Jean-Mickaël Guérin	6WIND	WP4

Authors (organizations):

Jordi Palet (Consulintel), Alan Delaney (PACE).

Abstract:

This deliverable presents the prototypes delivered to the trials during the second year and the extension of the project.

Keywords:

Autoconfiguration, HE, IPv6, QoS, Stack, STB.

Revision History

The following table describes the main changes done in the document since created.

Revision	Date	Description	Author (Organization)
v1.0	20/02/2004	Document creation	Jean-Mickael Guerin (6WIND)
v1.1	26/03/2004	Update DHCPv6 section	Jean-Mickael Guerin (6WIND)
v1.2	05/11/2004	Add new section about new Head Prototype	Jean-Mickael Guerin (6WIND)
v1.3	05/11/2004	Adds section about Pace set top box IPv6 stack	Alan Delaney (Pace)
v1.4	07/11/2004	Final PSC review	Jordi Palet (Consulintel)

Executive Summary

IPv6 and QoS software design and implementation has been developed to provide in one hand IPv6 advanced features useful in PLC network, and on the other hand to make IPv6 aware the project Set Top Boxes.

This deliverable presents how to configure the equipments to take benefit of the technology developed during the project, including autoconfiguration with DHCPv6 and QoS.

The new Head End prototype is also described, gathering in one box IPv6 software and new PLC technology.

Table of Contents

1. ***Introduction*** ***6***

2. ***IPv6 Stack on Set Top Box*** ***7***

3. ***IPv6 Autoconfiguration: Prefix Delegation*** ***8***

3.1 **General commands** **8**

3.2 **Radius parameters** **8**

4. ***New QoS Implementation*** ***9***

4.1 **Tfc context** **9**

4.2 **Pre-defined conditioners** **10**

4.3 **Queuing context** **11**

4.4 **Pre-defined classes** **11**

5. ***Final Project Head End Prototype*** ***13***

6. ***Summary and Conclusions*** ***15***

7. ***References*** ***16***

Table of Figures

Figure 5-1: ***Reference Design to be used in IST2004*** **...... 13**
Figure 5-2: ***200 Mbps PLC Card*** **...... 13**
Figure 5-3: ***External Look of the Integrated PLC HE*** **...... 14**

1. INTRODUCTION

During the second year and the project extension, the main work was focus on IPv6 implementation in the Set Top Box, and a new implementation of QoS in the Head End.

Autoconfiguration with DHCPv6 (Prefix Delegation) has been improved as well, providing a CLI.

Near the end of the project, a new prototype for the Head End has been provided, including IPv6 features developed during the project and the new 200 Mbps PLC technologies.

2. IPv6 Stack on Set Top Box

The Set Top Box (STB) runs Linux using a 2.4.19 kernel. Many of the components in the software stack are not affected by the version of IP that is being used.

However, components that accept host names or literal IP addresses and use that information solely to connect to the remote host required minimal modification. They have been altered to use the *getaddrinfo* library call (or *dnsd_getaddrinfo* for non-blocking lookups to the DNS daemon) as this returns the pre-built structures for passing to the socket system.

Some components use UNIX domain sockets by default, but can be configured to use INET domain sockets over the IPv4 loopback interface. These do not require modification for IPv6 support. Some components use INET domain sockets to communicate with other processes on the same machine. These should be updated to use UNIX domain sockets by default.

Any components using multicast need to create INET6 protocol sockets and switch the *ioctl* calls to use IPv6 group membership adding and leaving in place of the IPv4 equivalents. The multicast transform (IGMP_ATI_UDP) has been updated to make it IPv6 compliant.

Pace components that are now IPv6-ready include:

- Engineering menu.
- Settings manager.
- Display manager.
- DNS daemon.
- NCMail (POP3 and SMTP components).
- MCC core.
- Pmsg.
- Parseurl.
- SendMessage.
- IGMP_ATI_UDP.
- Root filesystem (sets up the interfaces and routes).
- Stream__ATI_stream.

Additional components that are believed to be IPv6-ready but are not fully tested yet, include:

- Net-snmp.
- OpenSSH (but nothing uses OpenSSH currently anyway).
- Linux kernel (latest version (2.4.23) only; earlier versions have faulty support for MLD).
- Browser (tested to be fully IPv6-compatible except for SSL).

3. IPV6 AUTOCONFIGURATION: PREFIX DELEGATION

During the project, DHCPv6 base specification have been standardized and referred to RFC3315. DHCPv6 prefix option is in last stage and is about to be standardized.

3.1 General commands

DHCPv6 server configuration is done in 'dhcpv6server' context.

First enable DHCPv6 service:

6OS{myconfig-dhcpv6server} dhcpv6server enable

To set IPv6 DNS addresses that will be sent to CPE, use following command:

6OS{myconfig-dhcpv6server} dns-server X::X

3.2 Radius parameters

To fit the 6POWER scenario we needed to fix Radius parameters. There are three parameters to configure:

- IPv6 address of radius server.
- Port on which the radius server is listen to (default is standard 1812).
- Secret shared with the radius server.

Example:

6OS{myconfig-dhcpv6server} radius-server-address X::X

6OS{myconfig-dhcpv6server} radius-server-port 1812

6OS{myconfig-dhcpv6server} radius-server-secret *yoursecret*

4. NEW QOS IMPLEMENTATION

A new implementation has been developed, more adapted to the Head End.

DiffServ configuration is done by mean of two contexts:

- Traffic conditioning at ingress interface: The tfc context.
- Queuing at egress interface: The queuing context

4.1 Tfc context

Traffic conditioning configuration is done in 'tfc' context. Three kinds of conditioners can be defined:

- A token bucket meter, using command tb-meter.
- A two rate three-color maker, using command trtc-marker.
- A simple marker, using command marker.

DSCP of flows are marked at ingress interface using command marker. The commands filter and filter6 allow assigning IPv4 and IPv6 flows to a marker. DSCP will be used to assign the flows to traffic classes.

Defining a token bucket is done with the following command:

6OS{myconfig-tfc} tb-meter [name] [interface] rate [rate-value] depth [depth-value]

where:

- Name is the name of the token bucket, ef is a pre-defined name.
- Interface is the interface name e.g eth1_0.
- Rate-value is the rate of the token bucket expressed either in Kb/s or Mb/s (e.g. 512K 10M).
- Depth-value is the depth of the token bucket expressed either in Kb or Mb (e.g. 64K 1M).

Defining a two rate three color marker is done with the following command:

6OS{myconfig-tfc}trtc-marker [name] [interface] rate [rate-value] depth [depth-value] peak-rate [pr-value] peak-depth [pd-value]

where:

- Name is the name of the token bucket, ef is a pre-defined name.
- Interface is the interface name e.g eth1_0.
- Rate-value is the rate of the three colors marker, expressed either in Kb/s or Mb/s (e.g. 512K 10M).
- Depth-value is the depth of the three colors marker, expressed either in Kb or Mb (e.g. 64K 1M).
- Pr-value is the peak-rate of the three colors marker, expressed either in Kb/s or Mb/s (e.g. 512K 10M).

- Pd-value is the peak-depth of the three colors marker, expressed either in Kb or Mb (e.g. 64K 1M).

Defining a simple marker is done with the following command:

6OS{myconfig-tfc} marker [name] [interface] mark

where:

- Name is the name of the marker (e.g. foo).
- Interface is the interface name (e.g eth1_0).

Defining a flow filter is done with the filter or filter6 command:

6OS{myconfig-tfc}filter [name] [interface] [conditioner] [src_prefix] [src_port] [dst_prefix] [dst_port] [proto]

where:

- Name is the name of the filter (e.g. foo).
- Interface is the interface name (e.g eth1_0).
- Conditioner is the name of the conditioner associated to this filter. It must be of of the (tb-meter/trtc-marker/marker) conditioners define in the context.
- Src_prefix is the prefix where the address source must belong.
- A.B.C.D/M in case of IPv4 prefix (command filter).
- X:X::X:X/M in case of IPv6 prefix (command filter6).
- Src_port is the source port (UDP/TCP), 0 means any port.
- Dst_prefix is the prefix where the address source must belong.
- A.B.C.D/M in case of IPv4 prefix (command filter).
- X:X::X:X/M in case of IPv6 prefix (command filter6).
- Dst_port is the destination port (UDP/TCP), 0 means any port.
- Proto is the protocol to match: udp or tcp, 0 means all protocol.

4.2 Pre-defined conditioners

Pre-defined EF conditioner named ‘ef’ is based on a token bucket meter, marking DSCP with EF DSCP code point (101110) in profile packets, dropping out profile packets. Parameters to provide are the rate and the peak rate allowed. Then define EF flows to go into ‘ef’ conditioner.

Example:

eth1_0 is ingress interface

6OS{myconfig-tfc}tb-meter ef eth1_0 rate 6M depth 64K

6OS{myconfig-tfc}filter myEFtraffic ef eth1_0 1.1.1.1/24 0 2.2.2.2/24 80 tcp

Four pre-defined AF conditioners named ‘af-1’, ‘af-2’, ‘af-3’, ‘af-4’ are based on a two rate three color marker, classifying packets in committed profile, out of committed profile but in the peak profile, or out of peak profile. Drop precedence is marked into DSCP code points according to profile.

	Class AF1	Class AF2	Class AF3	Class AF4
Low Drop Precedence	001010	010010	011010	100010
Medium Drop Precedence	001100	010100	011100	100100
High Drop Precedence	001110	010110	011110	100110

Example:

6OS{myconfig-tfc}trtc-marker af-1 eth1_0 rate 3M depth 32K peak-rate 10M peak-depth 64K

6OS{myconfig-tfc}filter6 myAF1traffic af-1 eth1_0 ::/0 0 2001::/64 80 tcp

4.3 Queuing context

In this context one can define classes of services. The scheduling mechanism is based on Class Based Queuing that provides link sharing which permits to redistribute excess bandwidth to improve link utilization.

Link characterization is done using the link-bandwidth command.

Configuration of classes is done using ‘class’ command. Pre-defined classes are implemented for diffserv mechanisms – see below. You may set the maximum rate that the queue discipline will allow on egress interface using ‘interface’ command.

Defining a link is done with the following command:

6OS{myconfig-queuing}link-bandwidth [interface] [bw-value]

where:

- Interface is the interface name (e.g eth1_0).
- Bw-value is the link bandwidth, expressed either in Kb or Mb (e.g. 200K, 10M).

Defining a class is done with the following command:

6OS{myconfig-queuing}class [name] [interface] bandwidth [pbw-value]

where:

- Name is the name of the class, ef af-1, … are a pre-defined names.
- Interface is the interface name (e.g eth1_0).
- Pbw-value is the link bandwidth, expressed either in % of the total (e.g. 20%).

4.4 Pre-defined classes

Following figure shows how bandwidth is shared between EF and AF classes when using pre-defined classes:

Pre-defined EF class named ‘ef’ can be used to specify percentage of the total bandwidth for EF traffic. Any flows whose DSCP is EF codepoint will go into EF class. Other classes are not allowed to borrow bandwidth from EF class.

Pre-defined AF class named ‘af-1’, ‘af-2’, ‘af-3’, ‘af-4’ are used to configure AF classes by setting percentage of the total bandwidth. Any flows whose DSCP is one of AF codepoints will go into the right AF class. AF classes can borrow unused bandwidth from each other.

Other flows whose DSCP is neither EF nor AF will go into default class.

Example:

eth0_0 is egress interface – 14 Mbps to EF, 20 Mbps for AF-1

```
6OS{myconfig-queueing}link-bandwidth eth0_0 100M

6OS{myconfig-queueing}class ef eth0_0 bandwidth 14%

6OS{myconfig-queueing}class af-1 eth0_0 bandwidth 20%
```

5. FINAL PROJECT HEAD END PROTOTYPE

Two boxes composed the first prototype for the Head End: An IPv6 router and an external PLC modem. PLC modem was acting as an Ethernet bridge.

In parallel of the project, 6WIND software technology has been improved, being the major change the support of multiple Operating Systems such as BSD and Linux, through 6WIND product DevSuite.

Although Linux support is in alpha stage, this version has been used near the end of the project.

The key advantage of this early DevSuite is the ability to embed the PLC driver developed by DS2, for both PLC technologies, 45 Mbps and 200 Mbps.

The features are limited to IPv6 advanced autoconfiguration, that is, DHCPv6 Prefix delegation, with Radius access for Authentication and Authorization. The whole Advanced set of features are only available on the first prototype, running BSD on the router.

The following figures show the reference design used for the IST2004 trial, and a picture of the 200 Mbps PLC card that is plugged-in this unit.

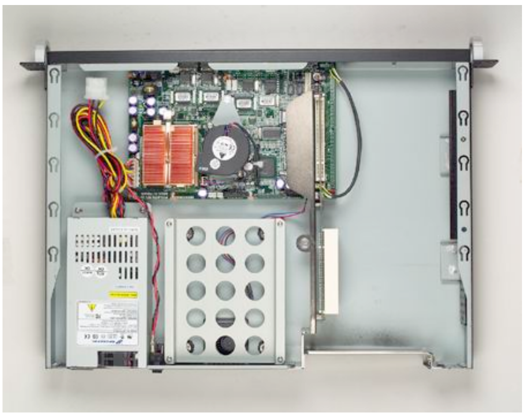

Figure 5-1: Reference Design to be used in IST2004

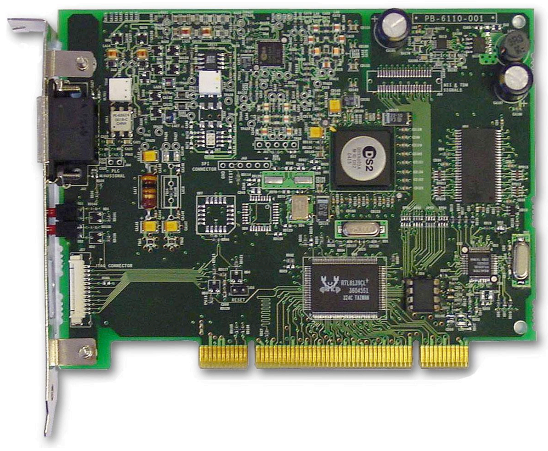

Figure 5-2: 200 Mbps PLC Card

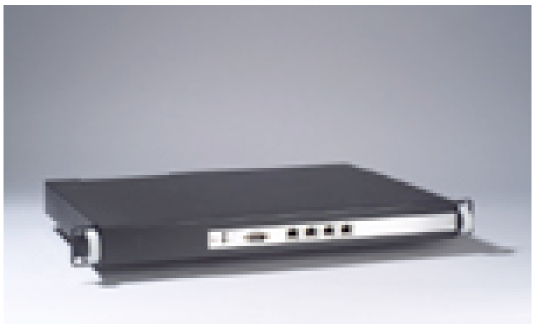

Figure 5-3: External Look of the Integrated PLC HE

This IPv6/PLC router prototype is based on ADvantech FWA-3680, with 200 Mbps PLC card, running 6WIND Devsuite software.

6. SUMMARY AND CONCLUSIONS

This document provides the details of the final prototypes of the devices developed in the 6POWER project.

Those devices include the digital Set-Top-Box with integrated IPv6 stack, and the IPv6/PLC Head End.

Both devices are to be used in the final project demonstration at IST2004 event.

7. REFERENCES

RFC 2474	Definition of the Differentiated Services Field (DS Field)
RFC 2475	An Architecture for Differentiated Services
RFC 2597	Assured Forwarding PHB Group
RFC 2598	An Expedited Forwarding PHB Group
RFC 2698	A Two Rate Three Color Marker
RFC 3315	Dynamic Host Configuration Protocol for IPv6 (DHCPv6)
RFC 3346	DNS Configuration Options for DHCPv6
RFC 3633	IPv6 Prefix Options for DHCPv6