Telecommun Syst (2016) 61:221–233
DOI 10.1007/s11235-015-9996-6

The IPv6 QoS system implementation in virtual infrastructure

Halina Tarasiuk · Sławomir Hanczewski · Adam Kaliszan · Robert Szuman · Łukasz Ogrodowczyk · Iwo Olszewski · Michał Giertych · Piotr Wiśniewski

Published online: 18 March 2015

Abstract The paper provides a novel solution to implement the IPv6 QoS system as one of Parallel Internets. This approach exploits both an IP protocol and a virtualization technology. Therefore, the proposed IPv6 QoS system is an evolutionary approach for Future Internet. Also, an important advantage of the proposal is that virtual nodes of the IPv6 QoS network operate on separated control and data planes. Such a way of the implementation allows to develop the system with flexible routing and virtual network management.

Keywords IPv6 protocol · Next Generation Networks · Virtualization technology · Software router

H. Tarasiuk (✉) · P. Wiśniewski
Institute of Telecommunications, Warsaw University of Technology, Nowowiejska 15/19, 00-665 Warsaw, Poland
e-mail: halina@tele.pw.edu.pl

P. Wiśniewski
e-mail: p.wisniewski@tele.pw.edu.pl

S. Hanczewski · A. Kaliszan
Chair of Communication and Computer Networks, Poznan University of Technology, Polanka 3, 60-965 Poznań, Poland
e-mail: slawomir@hanczewski.pl

A. Kaliszan
e-mail: adam.kaliszan@put.poznan.pl

R. Szuman · Ł. Ogrodowczyk · I. Olszewski · M. Giertych
Poznań Supercomputing and Networking Center, Noskowskiego 10, 61-704 Poznań, Poland
e-mail: rszuman@man.poznan.pl

Ł. Ogrodowczyk
e-mail: lukaszog@man.poznan.pl

I. Olszewski
e-mail: ict@iwolszewski.pl

M. Giertych
e-mail: mikus@man.poznan.pl

1 Introduction

Nowadays, one of the most important research topics of the information and communication technologies (ICT) area is Future Internet (Europe) named also as Internet 3.0 (US) or New Generation Networks (Japan). The main challenge of Future Internet is to solve well recognized limitations of current internet. These limitations refer, among others, to quality of service (QoS) guarantees for the following groups of applications: content delivery, high resolution video (3D), Internet of Things. Virtualization is a main technology which has been considered for Future Internet [1]. This technology allows to deploy IP-based and post-IP protocols in the same physical network infrastructure for efficient handling of the above-mentioned groups of applications as well as future applications. An application of the virtualization technology and post-IP protocols to a network requires a new architecture. In this spirit, the Future Internet Engineering project [2] designed the IIP architecture, which consists of 4 main levels (Level 1– Level 4) [2]. Level 1: Physical infrastructure, Level 2: virtualization (creation of virtual nodes and links), Level 3: Parallel Internets, Level 4: virtual networks (VN). Based on logical separation of routing tables, VN can be created for specific Parallel Internet (Level 4). As a consequence the IIP architecture and the IIP system provides two virtualization levels, Level 2 and Level 4. Finally, the project developed three Parallel Internets (Level 3) above phisical network infrastructure, which enables virtualization. One Parallel Internet is IP-based and two Parallel Internets are post-IP.

In this paper we focus on the IPv6 QoS Parallel Internet and the IPv6 QoS system. In particular, we describe the following features of the IPv6 QoS system: (1) data and control planes of the IPv6 QoS virtual node, and (2) Level 4 routing dedicated for IPv6 QoS virtual networks. The prototype of

the IPv6 QoS virtual node has been developed for the NP-3 EZchip programmable network processor (EZappliance) [3]. Exemplary test applications of the IPv6 QoS Parallel Internet are Internet of Things applications as e-health (eDiab, eAsthma, SmartFit,), public security (MobiWatch), HomeNetEnergy, and e-learning.

The IPv6 QoS system applies main functional blocks of NGN and DiffServ architectures [4–6] defined by ITU-T and IETF, respectively. However, comparing with proposals described in the literature e.g. [7–11] the system is a prototype implementation of NGN and DiffServ over a virtual infrastructure. Such a system requires to develop and implement: (1) new interfaces between Level 2 and Level 3 of the IIP architecture, and (2) QoS mechanisms at packet and call levels [12]. It also exploits such feature of the IPv6 protocol as IPv6 in IPv6 tunnels to deploy routing in VN (Level 4).

Network virtualization was one of the topics of the 4WARD project [13]. 4WARD proposes VNet architecture with a network virtualization concept. Comparing to our approach the 4WARD concept includes only network nodes operating on IPv4 protocol. It is dedicated to best effort network only. Our approach, as we mentioned before, focuses on the IPv6-based network ("IPv6 in IPv6 tunnels") with QoS guarantees for selected flows. For this purpose appropriate QoS mechanisms for QoS guarantees are deployed in the network [12,14,15].

The IPv6 QoS virtual node separates data and control planes. The most popular open platform, which also exploits such a concept is Openflow [16]. For this purpose Openflow defines a set of interfaces. However, based on our best knowledge these interfaces are dedicated for the control plane only and they does not support any QoS mechanisms. Therefore, we propose a new set of interfaces between the data and control planes dedicated to the IPv6 QoS virtual node.

The investigated solution for the IPv6 QoS control plane differs from the solutions used so far. The most common one is a full virtualization of both, control and data plane. Such an approach can only be used on devices supporting a full virtualization. It requires significant reserves of CPU time, discs and memory. A separation between data and control planes enables independent implementation of control plane entities from data plane hardware platforms. Our way of virtualization of routing functions makes a considerable difference comparing to the existing solutions. The routing functionalities are performed independently within one process. This simplifies a management and decreases the hardware requirements.

The paper organization is the following. Chapter 2 provides more details about the IPv6 QoS system. Chapter 3 describes a concept of routing and VN for this system. Implementation aspects of the IPv6 QoS node are discussed in Chapter 4. Chapter 5 shows exemplary results of control plane performances. Finally, Chapter 6 concludes the paper.

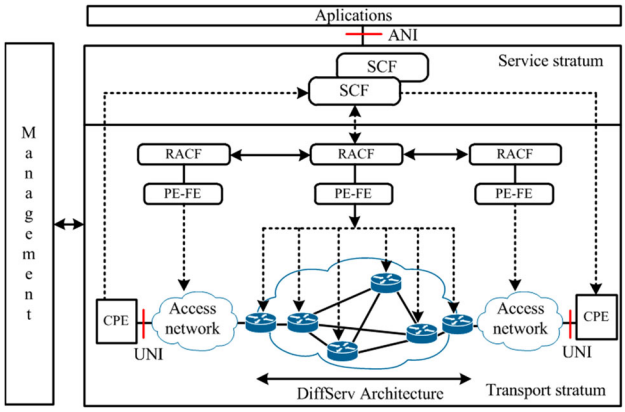

Fig. 1 IPv6 QoS architecture

2 IPv6 QoS system

This section describes principles of the IPv6 QoS Parallel Internet. Specifically, we introduce general architecture of the IPv6 QoS system and we present realization of the IPv6 QoS node in virtualization ecosystem provided by the IIP system.

2.1 IPv6 QoS architecture

Figure 1 shows the proposed architecture of the IPv6 QoS system that, as mentioned before, follows ITU-T and IETF recommendations [4,5]. It is composed of transport stratum and service stratum that are linked with management and applications/services layers. Moreover, since the IPv6 QoS system is deployed as one of the Parallel Internets (Level 2 of the IIP architecture), it utilizes interface from virtualization level (Level 2) and provides interface for virtual networks Level 4) of the IIP system. We assume that the resource and admission control functions (RACF) are performed for core IP domain and for each access network independently, as depicted in Fig. 1). Decided policy rules are then enforced in the underlaying network nodes by the technology-dependent PE-FE elements (Policy Enforcement Functional Entity).

2.2 Network node

Main functional elements of a prototype node of the IIP system are depicted in Fig. 2. The IIP system enables coexistence of different virtual nodes on the same physical node. Each virtual node, comprised of data plane, control plane and management modules, operates autonomously. Moreover, the IIP system provides performance isolation between nodes of Parallel Internets [2]. As a consequence virtual nodes may be based on different protocol stacks. The IPv6 QoS Parallel Internet is based on the TCP/IP protocol stack, whereas two other Parallel Internets, Data Stream Switching (DSS)

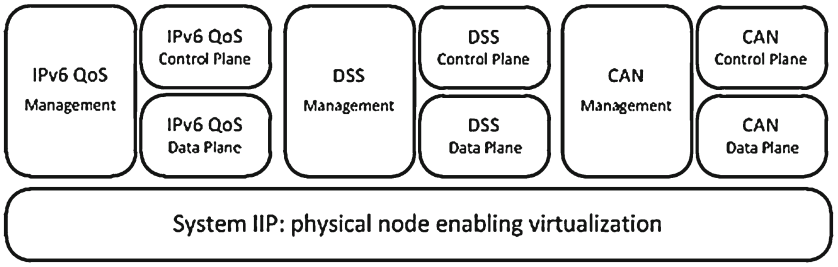

Fig. 2 Functional diagram of the system IIP node

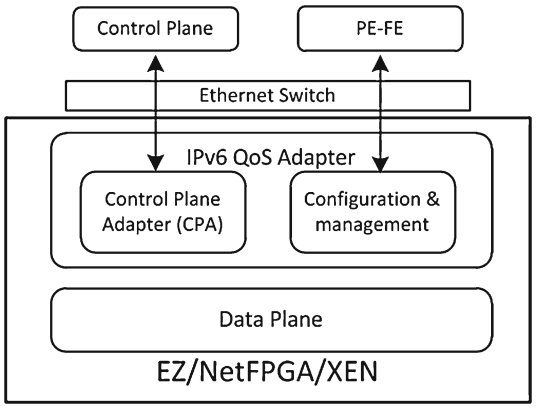

Fig. 3 Prototype model of the IPv6 QoS network node

and Content Aware Networks (CAN), deploy different customized protocol stacks.

The model of the IPv6 QoS Parallel Internet virtual node is presented in Fig. 3. It is composed of: (1) data and control plane modules, and (2) IPv6 QoS adapter that constitutes of control plane adapter and configuration and management element. The PE-FE block depicted in Fig. 3 is a part of the signalling system, that provides communication facility between network elements. The control plane module may be deployed in a separate network device (as depicted in Fig. 3) or in the same device, as data plane module. Such a flexibility allows to aggregate separate control plane modules. The aggregation of the control plane enables for significant simplification of the management process. The abstract model of the IPv6 QoS virtual node applies to all network platforms exploited in the IIP system [17], EZappliance [3], NetFPGA [18], and XEN [19]. Nevertheless, in the following chapters we present in details only the implementation of EZappliance-based IPv6 QoS virtual node.

3 Routing and virtual networks

3.1 Routing

We assume that IPv6 QoS network users do not operate directly on network resources of Level 3 of the IIP architecture, but they operate on dedicated virtual networks' (Level 4) resouces. This solution poses specific requirements on a control plane module.

The first requirement is the need for topology abstraction for VN, that operate on Level 4. The IPv6 QoS Parallel Internet's network consists of core routers (CRs) and edge routers (ERs). The VNs are not aware of the Level 3 structure. The only common element of the VNs and the IPv6 QoS Parallel Internet are the ERs, which are connected to VN. Moreover, from the viewpoint of the VN, nodes (ERs) are connected to common link (star topology—Fig. 4). The CRs, that operate on Level 3, are invisible on the Level 4.

The second requirement is the routing table for the Level 3 network. ER and CR nodes have a similar functionality range to such nodes in current IP networks. Level 3 routing table is required in order to exchange packets between ERs. VN on Level 4 are operable, after building topology tree on routing table for IPv6 QoS Level 3 network. Stable path determination between ERs is essential for right communication between those nodes. A difference between curent network and the IPv6 QoS Parallel Internet is in functionalities of edge nodes. ERs fulfill the routing functions on Level 3 and at the same time act as the points, that connect the VN from Level 4 with the network of Level 3.

The third requirement is ensuring communication between hosts in Level 4 VN. Each of the VNs is attached to different client networks with different addressing scheme. In consequence, each virtual node has its own unique link state data base. In order to ensure communication between given parts of VN, an appropriate number of dedicated routing processes need to be launched in ERs. Each process should be isolated. In the node, each separated routing proces owns virtual routing table.

The fourth requirement is virtual networks' data separation. The data separation technique is described in the Sect. 4.2.

We considered three solutions for routing in the IPv6 QoS network [20–25]:

- static routing,
- OSPFv3 routing protocol,
- special, dedicated routing protcol.

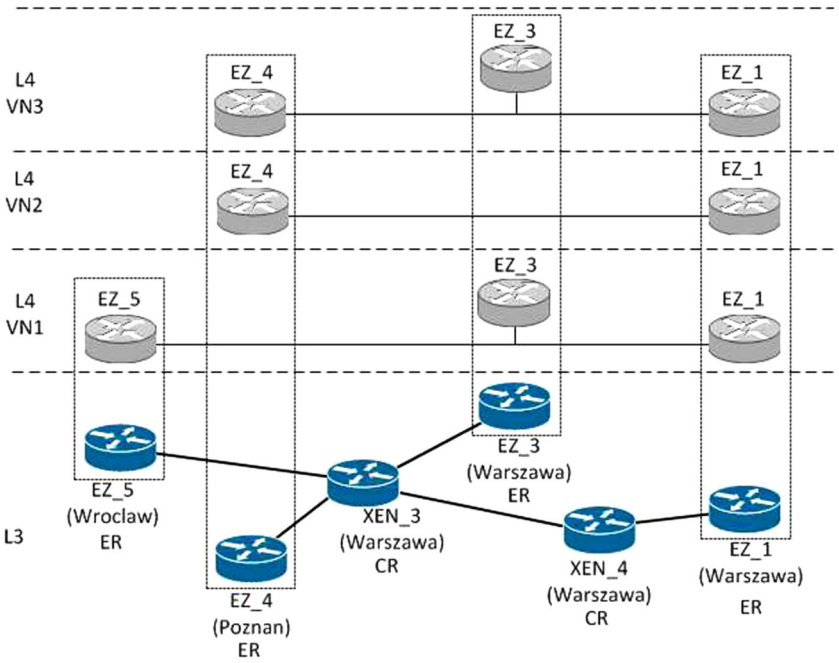

Fig. 4 Topology of the IPv6 QoS Parallel Internet laboratory network (L3), and exemplary Virtual Networks (L4 VN1, L4 VN2 and L4 VN3)

Finally OSPFv3 protocol was chosen. The main reasons to choose OSPFv3 protocol were: protocol stability, ready open source code implementations and IPv6 protocol support. Static routing does not guarantee the network stability on both levels (three and four), when the topology or link state is changing. Using static routing is not scalable and does not fulfill the IIP project assumptions.

Unlike the OSPFv2, the OSPFv3 protocol was designed to be independent of the protocol operating in the third (network) layer of the OSI model. The OSPFv2 was only intended for IPv4, so it became necessary to develop a next version of the OSPF protocol to support IPv6. The OSPFv3 is also fit for routing path calculation in networks with the new version of protocol operating in the network layer. The OSPF protocol is characterized by stability and scalability. The OSPFv3 specification is available for free [26]. There are also available open source implementations of OSPFv3 protocol, e.g. Quagga [27] or Xorp [28].

Chapter 4.1 describes implementation details of the control plane for IPv6 QoS network.

3.2 Virtual networks

The virtual networks are formed using IPv6 QoS Parallel Internet resources on Level 4 of the IIP architecture. All VNs are formed by virtual network operator (VNO) for long time period i.e. days, months, relying on service provider (SP) demand for applications group e.g. e-health. Over one core network domain two or more sites can be connected by VN. The service/applications requirements, passed on from the SP to the VNO, influence the VN forming process. The VN creation, modification and removal are done by the VNO via the management functionality offered by the IPv6 QoS system. The appropriate set of nodes and links is provided by parallel internet operator (PIO) to SP relying on the aforementioned service/applications requirements. Various virtual private network (VPN) services [29] can be used to create VN. A VPN can be considered as a network service with a set of tunnels between network end-points. This kind of network offers traffic isolation and security depending on the chosen technique. Different VPN services used on provider edge nodes were considered. Finally, Level 3 VPN has been chosen as a technique for the VN creation in the IPv6 QoS Parallel Internet. The reason was that clients do not have to use their own routing protocols because they are offered by VN Provider. Level 3 VPN functions as an IP subnet which interconnects user IP networks situated in different locations. Each Level 3 VPN is using a separate virtual routing table (VRF) on a provider edge router so the IP addresses between VPNs do not have to be unique. There are many different ways to build an IPv6 VPN, e.g. as BGP/MPLS VPNs [30] or through various tunnels techniques acting on top of IP protocol. Unfortunately, due to a lack of complete IPv6 implementation for LSP creation the first approach needs a dual-stack IPv4/IPv6. Therefore, we recommended to use VNs made of tunnels, because we do not consider to use IPv4 protocol at all, in our solution. Different protocols for packet tunnelling: IPsec, IPv6 in IPv6 or generic routing encapsulation (GRE) can be used to create the tunnels. However, it is important that packets encapsulated by them can be visible as IPv6 packets in the core network.

To fulfil requirements, we have selected IPv6 in IPv6 encapsulation mechanism [31] without any additional packet encryption (IPsec), as it is simpler solution for

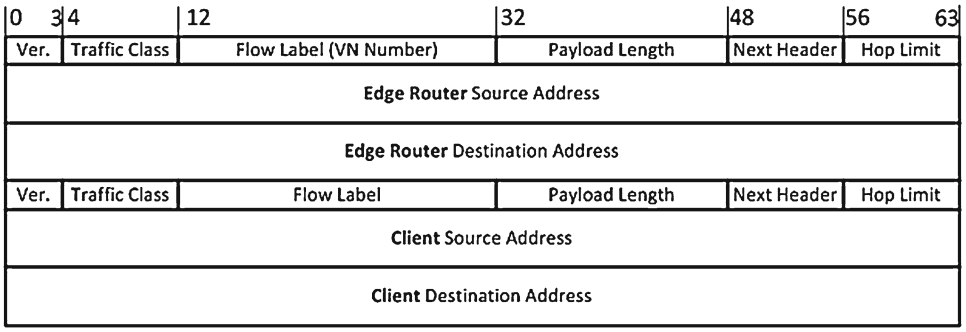

Fig. 5 IPv6 in IPv6 tunnel header example

implementation than GRE packets. Since this approach follows the Generic Packet Tunnelling in IPv6 Specification [32] and it only exploits standard features of IPv6 protocol, there are no other requirements for using additional protocols or packet types. Each packet incoming from access networks is encapsulated with additional IPv6 header that contains source and destination addresses of tunnel end-points. The traffic class field is copied from the headers of incoming IPv6 packet to encapulation IPv6 headers. The advantage of this method is the fact that packets' payload does not have to be analysed in the core network nodes reducing packet processing time. Moreover, "Flow Label" field in the outer packet header contains a numeric identifier of VN to distinguish traffic and QoS parameters between different VNs. System management is responsible for assigning unique value of the VN identifier during the VN creation process.

In the proposed solution we assume that every client's IP packet is encapsulated in the packet with a new IPv6 outer header (as it is depicted in Fig. 5). In this new header for each packet the Source Address field contains an IP address of ingress edge router - tunnel entry point node while the destination address field contains IP address of egress edge router - tunnel exit point node. The traffic class field value is copied from original (client) packet header to the new header without any modification enabling CR to treat such a packet properly without inspecting inner IP header. Virtual network identifier is stored in the Flow Label field of the outer header. Entire field (20 bits) is reserved for this number. The value of Hop Limit field is initially set to 32 by ingress edge router (tunnel entry point) and it is decreasing by 1 on each core network router on the path passing this packet. The remaining fields: version, payload length and next header will fulfil IPv6 specification [31]. In the inner IP header the Hop Limit field is decremented only by ER so the core network infrastructure is not visible from client hosts. Other fields in original header remain unchanged.

To manage properly such VN a dedicated management architecture has been designed. The Level 4 of this management system architecture is shown in Fig. 6. It consists of three functional modules which are responsible for VN management. virtual network acceptance (VNA) component running on the Level 3 management creates separate instance of the Level 4 sub-system to manage particular VN. Single Level 4 management instance representing single VN consists of instances of: routing management (RM), resource and topology management (RTM), and virtual network status management (VNSM).

RTM is the module on the Level 4 management subsystem mainly responsible for maintaining the logical topology, other related resources and information about VN. Single instance of the RTM is related to the single instance of the virtual network and all the operations are limited to this particular VN. It interacts with other Level 4 management modules—RM and VNSM—as a data provider. RTM provides the VN topology including such parameters like: list of ER belonging to the specified VN, IP addresses of Level 4-ER connecting client networks and links between Level 4-ER with information about their capabilities.

RM is a module responsible of routing maintenance on VN level. Similary to RTM single instance of the RM is related to the single instance of the virtual network and all operations are limited to appropriate VN. It also interacts with RTM and VNSM Level 4 management modules and also L3 management modules.

The next module of the Level 4 management system is VNSM. Single instance of VNSM is related only to the single instance of the virtual network and all operations are limited to selected VN. It cooperates with neither other running VNSMs nor different VNs. For the proper working VNSM requires connectivity with virtual network (for performing active monitoring) and access to its ERs (for analyzing nodes counters and active tests commissioning). The component functionality is focused on the monitoring of the actual state of the entire VN, i.e. its elements including virtual nodes, links, connectivity to the client networks. However, the implementation of this module is expected in the extended version of our system.

Access to the Level 4 management system is available through the dedicated web interface for VN users and it allows to perform all VN operations. In particular it allows to monitor network topology and routing maintenance.

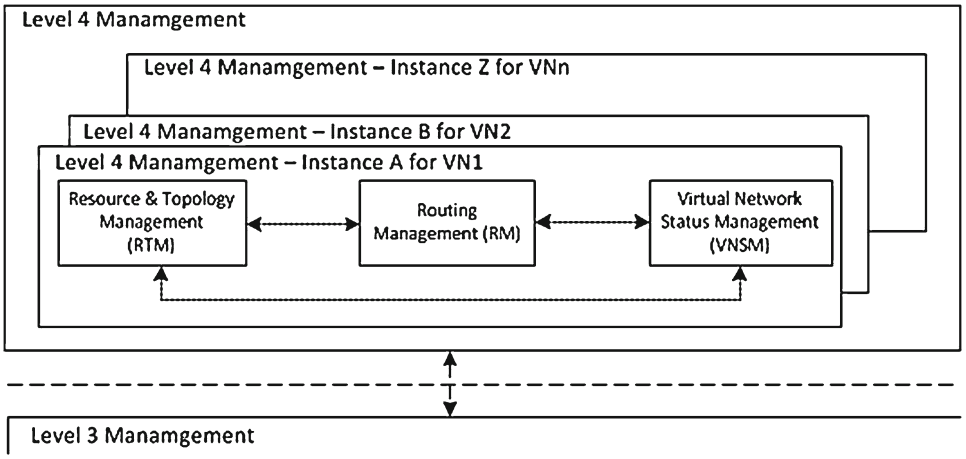

Fig. 6 VN management modules and interfaces

4 Implementation aspects of IPv6 QoS edge node

4.1 Control plane

The control plane (CP) implementation for the IPv6 QoS network node was divided into several steps. In the first step, the base (reference) implementation was chosen. As it was mentioned in chapter 3.1, CP functions are handled by software router. Such a router has to fulfill the following requirements:

- stability,
- OSPFv3 protocol support,
- possibility of its modification in order to distribute its modules that operate on separate machines.

The initial idea involved the use of one of the software routers: XORP [28] or Quagga [27]. However, having considered the router stability and the OSPFv3 implementation issues Quagga was chosen. Another reason for this decision is the fact that Quagga is the most common routing protocol software distribution employed by software router producers, and it is written entirely in C. The CP built for the IPv6 QoS network was based on the Quagga in version 0.99.18 (releaseded on 23.03.2011).

The next step was modification of Quagga implementation, that let the router working according to our specification. The implementation includes the following modifications:

- Quagga modification, that enables possibility of launching Quagga modules on remote machine (virtual or physical).
  The remote machine with CP is outside data plane (DP). The Quagga software architecture is module oriented. CP functionality is implemented in modules that handle routing protocols, e.g. ospf6d module is responsible for OSPFv3 protocol. Zebra module intermediates between DP (Linux Kernel) and CP modules. The Zebra module was called control plane adapter monitor (CPAM) in the IIP project. Zebra protocol [33] provides DP abstraction, that is DP's architecture independent. Zebra protocol messages may be exchanged via local sockets or TCP sockets. In order to enable possibility of CP module distribution, Zebra server rules modification was required. This modification lets the server to accept connection from remote machine. Previously only local connections were accepted. CP module modification was also required. Each CP modules has a direct access to DP interfaces. By using Berkley socket API, the modules were sending and receiving routing protocol messages. Therefore, it was necessary to add new control plane adapter transceiver (CPAT) module, that is responsible for providing access for CP module to DP interfaces on remote machine. The messages between CP and CPAT are exchanged using control plane adapter transceiver protocol (CPA-TP). Figure 7 shows modified Quagga architecture.
- Implementation of multiple routing protocol instances within one CP (ospf6d) process. First routing instance is dedicated for IPv6 QoS network that operates on the Level 3. Other routing instances are dedicated for VN on Level 4. Each network has its own Virtual Routing Table.
- CPA-MP [34] and CPA-TP protocols modification that enables the support of VN. Each message has a field, that specifies the network number. Value 0 is for IPv6 QoS Level 3 network and the rest is for Level 4 VN.

Moving CP to remote machine was very important. It guarantees a full abstraction between DP and CP that is hardware independent. There are three different DP devices (NetFPGA, EZappliance, XEN). The DP devices are treated by CP in the same way. It is possible, because a communication between the software router and DP device is implemented through additional CPAM and CPAT module (Fig. 7). To provide communication between DP and CP, two TCP connections are established. The first connection operates on 2600

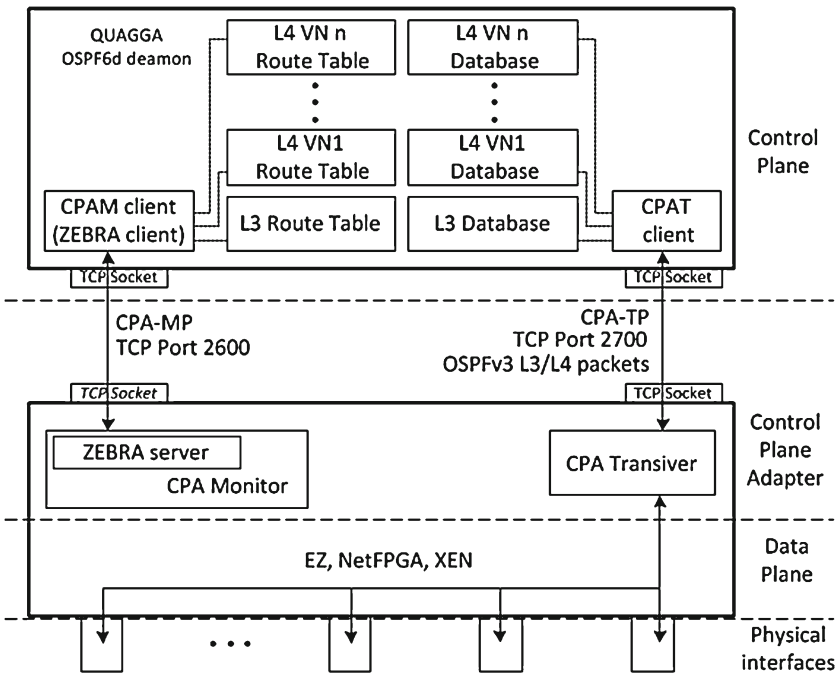

Fig. 7 Control plane and data plane communication

port and forwards CPAM (modified Zebra protocol) messages. The second connection performs on 2700 port and intermediates in exchanging routing protocol massages via CPA-TP protocol. Such an approach allows the scenario that the CPs of all nodes are on the same machine. The described implementation does not prohibit launching of CP on the DP machine.

4.2 Data plane

The IPv6 QoS adapter (adapter) acts as a middleware between CPA module in control plane, and EZproxy (HostCPU) and NP3 network processor in data plane (see Fig. 3). CPA supports modified zebra protocol (CPA-MP) to share information about EZappliance data plane interfaces, routerID parameter and static routing entries for both Level 3 and Level 4 instances. This module acts also as an OSPFv3 packets modifier and transmitter that: (1) writes appropriate values in TrafficClass and FlowLabel fields in the header, and (2) handles OSPFv3 packet tunneling for Level 4. The Adapter also contatins the Configuration and Management module. It provides command line interface (CLI) towards PE-FE module from transport stratum of NGN [35] architecture. CLI is used for the node configuration. Whole Adapter's configuration of EZappliace is maintained in configuration file in JavaScript object notation (JSON) format and commited to network processor on demand. API provided by NP-3 processor is used for configuration. Search structures of NP-3 are adopted to be used as L3 and L4 (VRF) routing tables. They also play roles of classifiers and configuration tables. Adapter is implemented on the external Linux server. The common object request broker architecture (Corba) [36] technology is used for inter-modules communication in the distributed edge node architecture. Network processor is responsible for all data plane operations assigned to Edge Router i.e. classification, two-level routing, policing, shaping, scheduling and IP in IP tunnel termination. Shaping and scheduling are performed by built-in configurable traffic manager while other operations are performed by programmable task optimized processors (TOPs). Only OSPF packets are transmitted via adapter to proper OSPF instance.

The assumed virtualization technology impacts on the implementation of the network processor part in the IPv6 edge node. The IIP system provides a virtualization layer which is the base for higher level implementations. The result of this solution is a separation from physical layer. Therefore, the implementation of the IPv6 QoS virtual node is unaware of physical ports and links. Instead, virtualized resources are presented as independently numbered virtual ports. Additional processing before and after IPv6 router code is a result of this abstraction.

EZappliance NP-3 network processor consists of 5 processing stages (TOPs): parse, search I, resolve, search II and modify. The initial part of virtualization processing is performed on first stage (TOPparse). This allows IPv6 router code to start processing in TOPparse already having information about ingress virtual port for each packet. Because of NP3 architecture the second part of virtualization processing has to begin in third stage (TOPresolve). For this reason the number of virtual port for egress needs to be specified by IPv6 code in TOPresolve. All decisions related to packet classification and routing must be taken in TOPresolve. The

limitation of processing stages available for crucial parts of IPv6 router implementation affects its efficiency and hence performance of the entire solution. Availability of only one search stage (i.e. TOPsearch I) for IPv6 QoS processing results in high logical parallelization of search queries. Some of performed search operations frequently appear to be unnecessary because of the results of other searches performed in parallel. Virtualization also has its advantages. Virtual link can transfer IP packets directly allowing IPv6 code to be unaware of link layer technology. This makes the IP layer completely independent from lower layers. Therefore, there is no need to keep IP-MAC mapping tables, hence neighbour discovery protocol (NDP) is also used in a much lesser extent.

5 Exemplary results

The evaluation of the presented solution can be done according to its efficiency and resources requirements. The efficiency depends on the functioning of DP and has a direct influence on the QoS parameters. For the DP evaluation, the QoS parameters are the primary criteria. For the evaluation of the routing implementation the main criteria are resources requirements that include: operating memory, disk memory and processor time (CPU load).

The effective resources management is conducive to an increase in the system flexibility and, consequently, makes it possible to support more virtual machines. The Sect. 5.1–5.4 show the utilized resources with the application of the solution presented in the paper. In addition, the influence of moving the CP to a separate machine is considered.

5.1 CPU resources

The main implementation criterion was efficiency followed by the CPU uniform time distribution between the routing processes of each virtual network. This was made possible by an extensive use of libraries elaborated for the Quagga project. An appropriate library was written specifically for the purposes of the project, which supported multitasking. Technically, multitasking was implemented with the help of co-routines. Co-routines are executed according to the FIFO algorithm and, consequently, the starvation free flow control process is secured (no influence between routing processes). The command interpreter of each Quagga module (including the module supported by the OSPFv3 protocol) contains a command that shows statistics with the co-routines that load the system most. The operating system sees all routing processes as one process, which provides a number of advantages. The operating system executes fewer commands. Task switching at the application level is much faster than switching a task context by the operating system. Another advantage is the possibility to determine how heavily the CPU is loaded

```
PID  USER PR NI VIRT RES  SHR S %CPU %MEM TIME+   COMMAND
2309 root 15 0  2944 1276 748 S 0.0  0.5  0:00.00 lt-ospf6d
```

Fig. 8 Example of the results of the `top` command

by the process supporting CP and how much memory is used by this process. These statistics are available by using the `top` system command (Fig. 8).

During the tests, the CPU load was lower than 1 %. As the load relies on the intensity with which Control Plane messages (OSPFv3 messages) are exchanged between IPv6 QoS nodes, it is a function Eq. (1) of two parameters: K and N, where parameter K is the number of nodes and parameter N is the number of VN:

$$L \approx KN + C. \tag{1}$$

The constant parameter C refers to the operating system itself. A single machine is enough for the presented concept, which eliminates the necessity of setting up separate virtual machines for each virtual network.

Taking into account the fact that $KN << C$, the computational complexity order is equal to $\Theta(C)$ and does not depend on the number of VN. In the solution with the virtual nodes applying to Level 4 (edge router), the load equals

$$L \approx KN + NC, \tag{2}$$

therefore, the computational complexity is equal to $\Theta(NC)$, which means that it is proportional to the number of virtual networks.

5.2 Memory resources

The maximum number of virtual machines is limited by the available system memory resources. The machines cannot share common memory resources nor can they occupy the memory to a higher extent than the whole system has to offer. Consequently, this means that the memory occupied by one virtual network cannot be used by other networks. The memory allocated to each machine should have some amount of headroom to spare, in case memory resources are poorly used. Memory of the machines involved is consumed by the operating system kernel and by the CPU process, among others. The Quagga software router implementation provides a possibility to monitor memory allocation. Memory allocation, in turn, depends on the number of neighbours and VN.

The `show` command is one of the many commands available in the interpreter of a process supporting the CPU. The `show` command is used to display detailed information on the allocations of the memory. A demand on memory via CPU software starts out with a half a megabyte of memory. This demand (as in the case of CPU) is a function of two parameters: N and K. It is, however, the minimum increase,

so the memory resources do not restrict the system. The state of the machine memory on which CPU is running can be also monitored through the `top` command.

In order to estimate memory requirements in relation to the number of nodes involved, numerous measurements were conducted in the study. Each node was composed of two interfaces. One interface was connected to the same link as the remaining nodes. The other supported connections with the client network. The measurements carried out in the study prove that a process of connecting a new node consumes about 4 KB of memory. The procedure does not entail any problems, so one may assume that the system is not restricted by memory resources.

5.3 Disc resources

Most likely, the demand for disc resources will remain stable, not relying on the number of VN. A separate configuration, stored in 1 kB memory, is required for each network. In addition, event logs can be stored for each of the configured virtual networks. The latter option may be disabled, however, when an appropriate configuration is being used.

5.4 Signalling messages delay

Moving the control plane module to another machine results in minimal delays in sending signalling messages. On the assumption that Node A sends an OSPF hello packet to a neighboring Node B, the packet passes through 3 points, traversing the network. It is first transmitted from CP to CPA.

First, it is sent between CP and CPA. Then, CPA sends the packet via the DP interface of Node A to DP interface of Node B. Next, the message is being transferred between CPA and CP in Node B. Such a solution might seem to have only disadvantages when we consider a situation in which sending of messages of a routing protocol is involved. Still, it does offer some advantage in the case of many networks as it enables CP to carry out blocking operations that stop the process from the moment of sending the packet. The CP-CPA link is not loaded and makes fast packet sending (without any delays) possible. The CPA serves as a buffer for these messages.

Sending an OSPF packet may take a while, as the link between the two nodes might be loaded. However, in the proposed solution this feature has no impact on the CP operation. Therefore, moving of the Control Plane module to a dedicated (separate) machine, without a direct access to interfaces of Data Plane device, has no important influence in the performance of a node and on delays in sending messages.

What has resulted from the tests taken in PL-Lab network [37] (see Fig. 4) using ping6 command is that the average response time between CP and CPA machine is 0.23 ms.

More complicated is mesuring of routing protocol message delay between CPA that is transmitting routing message and CPA that is receiving this message. The mean delay in bidirectional transmission between node A and node B we can obtain in a following way. Let's assume, that $T_{A\to B}$ means an average delay of messages send by A to B and a $T_{B\to A}$ means an average delay of messages send by B to A. Under assumption that the number of the messages sent between nodes A and B in both directions (from A to B, and form B to A) is the same, an average delay is equal to:

$$T_{A\leftrightarrow B} = \left(T_{A\to B} + T_{B\to A}\right)/2. \tag{3}$$

Let's now consider the caltulating method of the delay $t_{A\to B}(n)$ in sending n-th packet from A to B. Let's assume the following notation: $t_{\to A}(n)$ is a time of receiving n-th packet in A node's CPT in order to sent it to node B and $t_{B\to}(n)$ is a time of sending by B node's CPA received packet to its CP. Tle local clocks in nodes are not ideally synchronized, as this is impossible. Let $\xi_{A,B}(n)$ be a relative time's synchronization error between clocks in node A and B, that determines clock's A delay in comparison to clock B, when the n-th packet is sent. It can be written:

$$t_{A\to B}(n) = t_{B\to}(n) - t_{\to A}(n) + \xi_{A,B}(n), \tag{4}$$

$$t_{B\to A}(n) = t_{A\to}(n) - t_{\to B}(n) - \xi_{A,B}(n). \tag{5}$$

The mesurements were teken analzing the delay of hello message. Those packets are sent alternately by each node, therefore, we can assume that if A sends to B a packet number n, the node B sends to node A a packet number $n+1$.

$$T_{A\leftrightarrow B} = 1/k \sum_{n=1}^{k} \frac{t_{A\to B}(2n) + t_{B\to A}(2n+1)}{2}. \tag{6}$$

Substituting Eq. (6) into the Eqs. (4) and (5), we obtain:

$$T_{A\leftrightarrow B} = 1/k \sum_{n=1}^{k} \frac{t_{B\to}(2n) - t_{\to A}(2n) + \xi_{A,B}(2n)}{2} + \\ +1/k \sum_{n=1}^{k} \frac{t_{A\to}(2n+1) - t_{\to B}(2n+1) - \xi_{A,B}(2n+1)}{2}. \tag{7}$$

If we assume, that relative clock error is a slow changing function, we can write:

$$\xi_{A,B}(n) = \xi_{A,B}(n+1). \tag{8}$$

Therefore, the Eq. (8) may be transformed in a following way:

$$T_{A\leftrightarrow B} \\ = \sum_{n=1}^{k} \frac{t_{B\to}(2n) - t_{\to A}(2n) + t_{A\to}(2n+1) - t_{\to B}(2n+1)}{2k}. \tag{9}$$

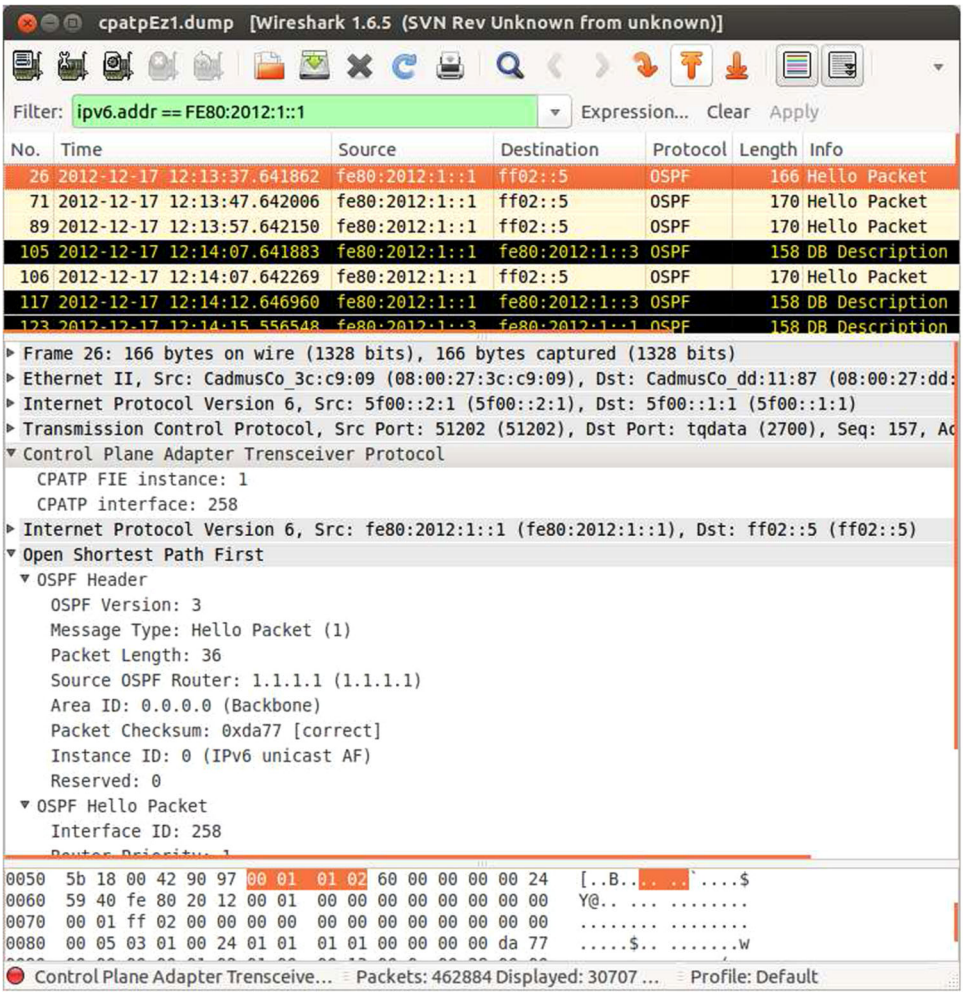

Fig. 9 Wireshark with plugin capable to parse CPA-TP

According to the Eq. (9), while measuring an average delay in sending the packets between A and B in both directions, the influence of an error in synchonization of the clocks has been eliminated.

The undertaken tests have proven that an average delay between any two edge nodes, being a part of the given Virtual Network, oscillates between 0.2 and 0.4 s. Such a delay is related to an early CR's CPA implementation stage. The delay in question refers only to the messages exchanged by routing protocol and is significantly bigger than the data packets delay exchanged by the L4 VN.

The delay does not influence an efficiency of Data Plane functioning. In accordance with the rules of software engineering, the main emphasis in the first stage of implementing works is put on reliability. Along with the next versions the efficiency is being increased so that the final version can offer the full efficiency. The plugin for Wireshark program, written for the tests, has facilitated the conduction of measurements on delay in sending the messages between CPA edge routers. This plugin enables an analysis of the OSPFv3 protocol messages, sent in data field of CPA-TP protocol. Fig. 9 presents program Wireshark monitor with an additional plugin. On the figure a message of CPA-TP protocol is visible The Wireshark program makes it possible to filter the received messages, so we are able to search for OSPF messages with the precised sender and receiver address. Thanks to an appropriate filter configuration, we are able to display the received OSPF Hello messages, sent by A node. This filter is described as follows: *ipv6.addr == FE80:2012:1::A and ipv6.addr == FF02::5*, where FE80:2012:1::A is an IPv6 node's A address and FF02::5 is multicast, destination address. This enables a conduction of simple analysis. It should be mentioned, that OSPFv3 algorithm enables including of delays in the configuration of interfaces.

6 Summary

The paper proposes a novel approach for implementation of the IPv6 QoS system as one of the Parallel Internets in the

virtualized network environment. Such an approach requires to adopt well known IPv6 QoS system to the virtual environment. Moreover, the IPv6 QoS network node is based on the separation of data plane and control.

Therefore, the proposed solution is based on the hardware independent model of virtualized IPv6 QoS network node, which exploits the concept of data and control plane separation. The model is deployed exploiting hardware (EZappliance, NetFPGA) and software (XEN) platforms for data plane. For simlicity of description, we focus only on EZappliance implementation and Linux-based control plane. Thanks to hardware implementation of data plane, it is possible to achieve wire-speed processing of data packets. Moreover, the paper provides the resource and efficiency requirements of the control plane implementation.

The developed control plane software will be publicly available in the near future [38].

Acknowledgments This work has been partially supported by the Polish Ministry of Science and Higher Education under the European Regional Development Fund, Grand "Future Internet Engineering", No. POIG.01.01.02-00-045/09-00. We would like to thank our colleagues for their contriribution to the project and valuable discussions.

References

1. ITU-T Rec. Y.3001 (2011). Future networks: Objectives and design goals. ITU Telecommunication Standardisation Sector.
2. Burakowski, W., Tarasiuk, H., Beben, A., Danilewicz, G. (2012). Virtualized network infrastructure supporting co-existence of Parallel Internet. In in *Proceedings of 13th ACIS international conference on software engineering, artificial intelligence, networking and parallel/distributed computing (SNPD)* (pp. 679–684). Japan: Kyoto.
3. Ezappliance. http://www.ezchip.com
4. ITU-T Rec. Y.2111. (2008). Resource and admission control functions in next generation networks. ITU Telecommunication Standardisation Sector.
5. Blake, S. et al. (1998). An architecture for differentiated service. RFC 2475 (Informational), Internet Engineering Task Force.
6. Mantar, H. A. (2007). A scalable QoS routing model for diffserv over MPLS networks. *Telecommunication Systems*, *34*(3—-4), 107–115. (2007).
7. Burakowski, W., et al. (2011). The IP QoS system. *Journal of Telecommunications and Information Technology*, *3*, 5–11. 2011.
8. Klincewicz, J. G., Schmitt, J. A., & Wong, R. T. (2002). Incorporating QoS into IP enterprise network design. *Telecommunication Systems*, *20*(1–2), 81–106.
9. Wang, S., Xuan, D., Bettati, R., & Zhao, W. (2010). Toward statistical QoS guarantees in a differentiated services network. *Telecommunication Systems*, *43*(3–4), 253–263.
10. Bak, A., et al. (2003). A framework for providing differentiated QoS guarantees in IP-based network. *Computer Communications*, *26*(4), 327–337.
11. Burakowski, W., et al. (2008). Provision of end-to-end QoS in heterogeneous multi-domain networks. *Annales des Telecommunications-Annals of Telecommunications*, *63*(11–12), 559–577.
12. Bernet, Y., et al. (2002). An informal management model for diffserv routers. RFC 3290 (Informational), internet engineering task force.
13. Papadimitriou, P., et al. (2009). Implementing network virtualization for a future internet. In *Proceedings of 20th ITC specialist seminar*. Hoi An, Vietnam.
14. Burakowski, W., & Tarasiuk, H. (2003). Admission control for TCP connections in QoS IP network. In C.-W. Chung, et al. (Eds.), *Proceedings of the Human.Society@Internet, Web and communication technologies and internet-related social issues* (pp. 383–393). Berlin: LNCS 2713, Springer.
15. Braun, T., Diaz, M., Enriquez Gabeiras, J., & Staub, T. (2008). The EuQoS system. In M. Diaz (Ed.), *End-to-end quality of service over heterogeneous networks* (Vol. 6). Berlin: Springer.
16. Mckeown, N., et al. (2008). Openflow: Enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, *38*(2), 69–74.
17. Chydzinski, A., Rawski, M., Wisniewski, P., Adamczyk, B., Olszewski, I., Szotkowski, P., Chrost, L., Tomaszewicz, P., Parniewicz, D. (2012). Virtualization devices for prototyping of future internet. In *Proceedings of 13th ACIS international conference on software engineering, artificial intelligence, networking and parallel/distributed computing (SNPD)* (pp. 672–678). Japan: Kyoto.
18. Gibb, G., et al. (2008). Netfpga - an open platform for teaching how to build gigabit-rate network switches and routers. *IEEE Transactions on Education*, *51*(3), 364–369.
19. Xen project. http://www.xen.org
20. Kim, D., & Tcha, D. (2000). Scalable domain partitioning in internet OSPF routing. *Telecommunication Systems*, *15*(1–2), 113–127.
21. Schmid, A., & Steigner, C. (2002). Avoiding counting to infinity in distance vector routing. *Telecommunication Systems*, *19*(3–4), 497–514.
22. Chakraborty, D., Zabir, S., Chayabejara, A., & Chakraborty, G. (2004). A distributed routing method for dynamic multicasting. *Telecommunication Systems*, *25*(3–4), 299–315.
23. Broström, P., & Holmberg, K. (2009). Design of OSPF networks using subpath consistent routing patterns. *Telecommunication Systems*, *41*(4), 293–309.
24. Rak, J., & Walkowiak, K. (2013). Reliable anycast and unicast routing: Protection against attacks. *Telecommunication Systems*, *52*(2), 889–906.
25. Grajzer, M., Zernicki, T., & Glabowski, M. (2012). ND++—an extended IPv6 neighbor discovery protocol for enhanced stateless address autoconfiguration in MANETS. *International Journal of Communication Systems*, *27*(10), 2269–2288.
26. Coltun, R., Ferguson, D., Moy, J., Lindem, A. (2008). OSPF for IPv6, RFC 5340 (Proposed Standard), internet engineering task force.
27. Quagga project. http://www.quagga.net/
28. Xorp project. http://www.xorp.org/
29. Andersson, L., Madsen, T. (2005). Provider provisioned virtual private network (VPN) terminology. RFC 4026 (Informational), Internet Engineering Task Force.
30. Rosen, E., Rekhter, Y. (2006). BGP/MPLS IP virtual private networks (VPNs). RFC 4364 (Proposed Standard), internet engineering task force by RFCs 4577, 4684, 5462.
31. Deering, S., Hinden, R. (1998). Internet protocol, version 6 (IPv6) specification, RFC 2460 (Draft Standard), Internet Engineering Task Force.

32. Conta A., Deering, S. (1998) Generic packet tunneling in IPv6 specification RFC 2473 (Proposed Standard), Internet Engineering Task Force.
33. Kaliszan, A., Glabowski, M., Hanczewski, S. (2012). A didactic platform for testing and developing routing protocols. In *Proceedings of the eighth advanced international conference on telecommunications (AICT)*. Stuttgart, Germany.
34. Kaliszan, A., Hanczewski, S., Glabowski, M., Stasiak, M., Zwierzykowski P. (2012). Routing and control plane in the parallel internet IPv6 QoS. In *Proceedings 8th international symposium on communication systems, networks digital signal processing (CSNDSP)*. Poznan, Poland.
35. ITU-T Rec. Y.2001. (2001) General overview of NGN. ITU telecommunication standardisation sector.
36. OMGs Corba website. http://www.corba.org
37. Krzywania, R., Dolata, Ł., Krawiec, P., Latoszek, W., Szymanski, A., Wszołek, J., (2012). Polish Future internet distributed laboratory. In *Proceedings of 13th ACIS international conference on software engineering, artificial intelligence, networking and parallel/distributed computing (SNPD)* (pp. 666–671). Japan: Kyoto.
38. IIP project. http://www.iip.net.pl

Halina Tarasiuk received the M.Sc. degree in computer science Szczecin University of Technology, Poland, in 1996 and Ph.D. degree in telecommunications from the Warsaw University of Technology, in 2004. From 1998 she is with Telecommunication Network Technologies Group at the Institute of Telecommunications, Warsaw University of Technology. From 2004 she is an assistant professor at the Warsaw University of Technology. From 1999 to 2003 she was collaborated with Polish Telecom R&D Centre. She participated in several European and national projects (2000–2013). Her research interests focus on Future Internet, NGN and NWGN architectures, node and network virtualization, signalling system performance, admission control and resource allocation methods and queueing mechanisms.

Sławomir Hanczewski received M.Sc. and Ph.D. degrees in telecommunications from Poznan University of Technology, Poland, in 2001 and 2006, respectively. Since 2007 he has been working in the Faculty of Electronics and Telecommunications, Poznan University of Technology. He is an Assistant Professor in the Chair of Communications and Computer Networks. He is the author, and co-author, of more than 50 scientific papers. He is engaged in research and teaching in the area of performance analysis and modelling of queuing systems, multiservice networks, switching systems and computer networks.

Adam Kaliszan received the M. Sc., Ph. D. degrees in telecommunication from the Poznan University of Technology (PUT), Poland, in 2005, 2010, respectively. Since 2008 Adam Kaliszan has been working in the Department of Electronics and Telecommunications, PUT. He is engaged in research and teaching in the area of programming and operating systems. He works also as developer in research projects. Adam Kaliszan is the author/co-author of 30 papers which have been published in communication journals and presented at national and international conferences. Adam Kaliszan is teaching in Poznan University of Technology largely focuses on the M.Sc.in Telecommunication where he gives lectures and conducts laboratory courses on Operating Systems, Network Embedded Systems, Programming in C++ and C#. Adam Kaliszan also participates in industrial education acting as a lecturer in courses on programming.

Robert Szuman graduated from Poznan University of Technology in 2000 and got the M.Sc. degree in Computer Science (Databases and Networks Designing). Since 1999, he has been co-operating with Poznan Supercomputing and Networking Center (PSNC), where he started work in the Network Department as a Network Management Systems Administrator. Now he is working as a Network Specialist in PSNC. His main fields of research interests are network management systems administration and configuration, broadband and optical networks monitoring, Quality of Service in computer networks, traffic analysis and measurement technologies, network management protocols, tools and procedures used by the Network Operation Center (NOC).

Łukasz Ogrodowczyk received the M.Sc. in the Electronics and Telecommunications, majoring in the Networks of Information Transport, from the Poznan University of Technology in 2007. From 2010 he works in Poznan Supercomputing and Networking Center. Co-author of some papers for international conferences (e.g. CSNDSP, Networks, Terena). He was involved in the Polish project Future Internet Engineering. Currently he works in the international FP7 project ALIEN. Lukasz is familiar with embedded systems, some programming languages like C and Python, network protocols and Quality of Service in next generation packet networks.

Iwo Olszewski received the M.Sc. degree in Computer Science, majoring in the Computer Networks and Distributed Systems, from the Poznan University of Technology in 2010. He started working in PSNC in 2006 as an network operator in PIONIER NOC. In 2010 he has joined the NGN Team as an analyst/developer. He has experience in object-oriented programming in Java and C# and deep knowledge about network protocols and devices. He is familiar with EZchip network processors and Juniper Junos platform development. He was involved in the national project "Future Internet Engineering" and FP7 projects NOVI and ALIEN. Co-author of several papers in conference proceedings (CSNDSP, NETWORKS, SNPD, TNC).

Michał Giertych received the M.Sc. degree in Computer Science from Adam Mickiewicz University in 2004. After graduating he has been working for Pozna Supercomputing and Networking Center as a software developer and computer systems analyst. He has been involved in European GANT, Bon-FIRE and Polish national FIE projects. His main areas of interest are design, implementation and testing applications for distributed network and federated cloud environments.

Piotr Wiśniewski is a Ph.D. student at the Institute of Telecommunications at the Warsaw University of Technology, where he received his M.Sc. (2010) and B.SC. (2009) degrees in Telecommunications. He is a research and teaching assistant at the Warsaw University of Technology and a specialist at the National Institute of Telecommunications. His research interests include network virtualization, Quality of Service, Information Centric Networks and Software Defined Networks.