Paper

On Dimensioning and Routing in the IP QoS System

Witold Góralski[a], Piotr Pyda[b], Tomasz Dalecki[b], Jordi Mongay Batalla[a], Jarosław Śliwiński[a], Waldemar Latoszek[c], and Henryk Gut[c]

[a] *Warsaw University of Technology, Warsaw, Poland*
[b] *Military Communication Institute, Zegrze, Poland*
[c] *National Institute of Telecommunication, Warsaw, Poland*

Abstract—This article presents dimensioning and routing solutions in IP QoS System designed during the implementation of the PBZ project: "Next Generation Services and Networks – technical, application and market aspects: Traffic management – IP QoS System". The paper presents the functional architecture together to the description of the functions and methods implemented in the system.

Keywords—IP QoS System, resource dimensioning, routing.

1. Introduction

The architecture of the IP QoS System and traffic control mechanisms have been specified during the PBZ project[1] and the implemented prototype has been tested in the testbed network. The proposed architecture of the IP QoS System is compatible with the next generation network architecture (NGN). Moreover, in terms of quality of service (QoS) assuring, this implementation is compatible with the differentiated services architecture (DiffServ) [1]. Figure 1 shows the IP QoS System architecture with implemented functional modules.

The proposed solution relates to resource management layer, which main objective is to separate the traffic submitted to the four classes of service (CoS): real time, multimedia streaming, high throughput data and standard. The resource management implemented in IP QoS distinguishes three basic processes directed to prepare the network for assuring guaranteed service for the new requests:

- resource dimensioning process between edge routers,
- routing process on the basis of QoS requirements, i.e., QoS-aware routing,
- resource reservation process for new call requests that takes into account quality of service requirements.

These processes we implemented by the following modules: routing management (ROMAN), resource management subsystem (RMS), policy decision – physical entity (PD-PE), transport resource control (TRC) and policy enforcement – physical entity (PE-PE). ROMAN module is responsible for routing in the network. It should be noted that the testbed network implements multiprotocol label switching traffic engineering (MPLS TE) tunnels for carrying traffic of given CoS. ROMAN module sets the path and creates TE tunnels between each pair of edge routers. In the project framework we implemented and tested different routing algorithms for TE tunnels configuration. Besides standard algorithms additional extended Dijkstra's algorithms have been implemented. ROMAN forces the TE tunnel in the routing algorithm by entering the command: *ip explicit-path* with specified intermediate nodes. After configuring tunnel, the module writes the information in a database and provides the information to the RMS module with the list of tunnels. The RMS module performs resource allocation and resource dimensioning, which depend on available resources and matrices of traffic demands, respectively. The primary task of the RMS module is to determine the link capacity and buffer size for each class of services inside a single domain. The resulting capacity and buffer size are set in the edge router and are the parameters used by the call admission control (CAC) function.

The article describes the implementation of modules responsible for proper router configuration within the testbed network. The main objective of the paper is to describe the specification as well as implemented algorithms of the modules responsible for dimensioning process and configuration of MPLS paths. Moreover, we describe how the different modules cooperate with each other and exchange data. The theoretical description comes with exemplary configuration results taken from the testbed network of the IP QoS System. In the conclusion, we summarize the achievements of the implementation of the system by describing the presentation of IP QoS System testbed on national exhibition, and propose system extensions for further development.

2. Functions of IP QoS System Modules

2.1. ROMAN Module

ROMAN module is responsible for the implementation of the routing process in testbed network that is compatible with the DiffServ architecture [1]. In DiffServ networks,

[1]This work is partially funded by Polish Ministry of Science and Higher Education, under contract number PBZMNiSW-02-II/2007 "Next Generation Services and Networks – technical, application and market aspects".

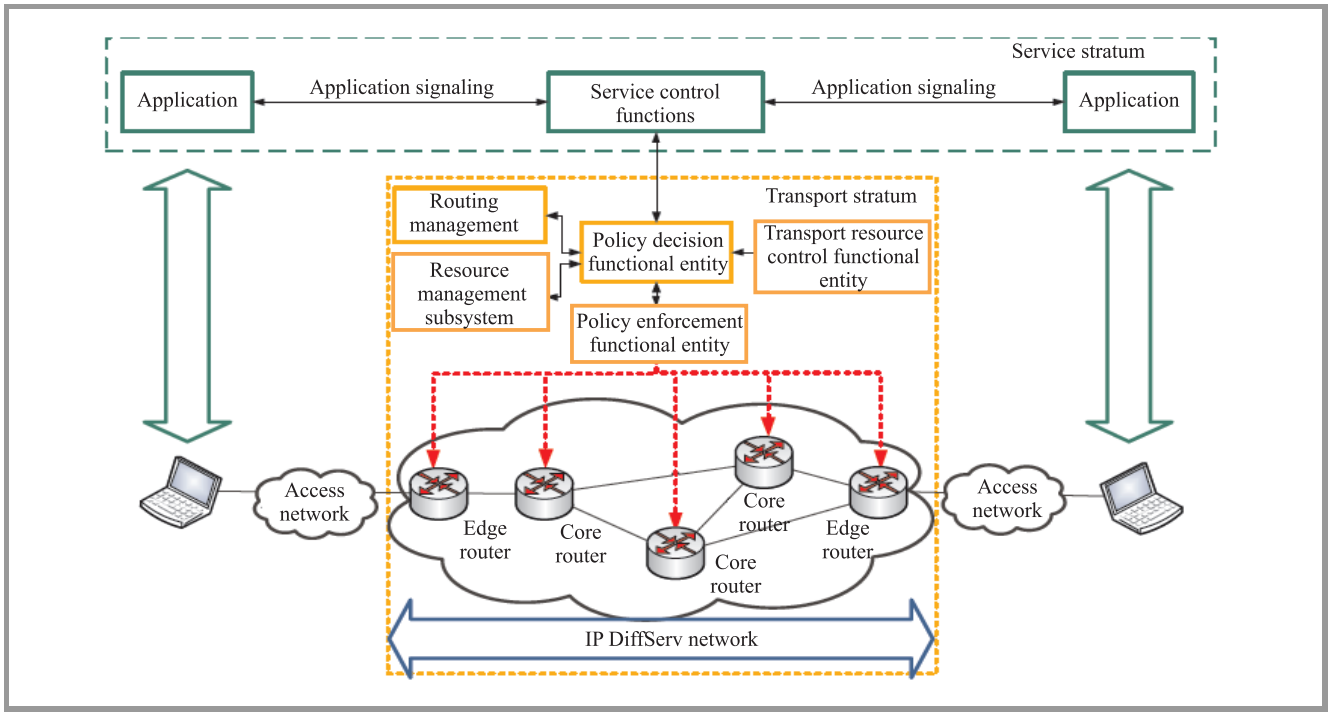


Fig. 1. Architecture of the IP QoS System.

edge routers support functionalities for single streams and core routers are aware only of aggregated traffic in proper CoSes. Testbed network implements MPLS TE tunnels for routing packets belonging to different CoS. Edge routers add and remove MPLS labels for incoming and outgoing packets, respectively. MPLS TE tunnels, or briefly TE tunnels, are defined by the so-called label switched path (LSP). To configure the LSP it is required to specify subsequent nodes from source to destination router. Routers in the MPLS network make forwarding decisions based on their label forwarding instance base (LFIB) tables. These tables contain labels which corresponding input and output interfaces. The paths on which are established the tunnels are determined using a routing algorithm implemented in ROMAN module. ROMAN task is to determine paths and create TE tunnels between each pair of edge routers for traffic classified into proper CoSes. DiffServ architecture assumes that the individual streams of packets sent by applications in the backbone are aggregated into streams of particular CoSes. In DiffServ architecture routers analyze DSCP field in IP headers, and on this basis are handled with appropriate CoS. Packets belonging to the CoSes in the MPLS network are distinguished on the basis of the value of EXP field in MPLS header (see Fig. 2).

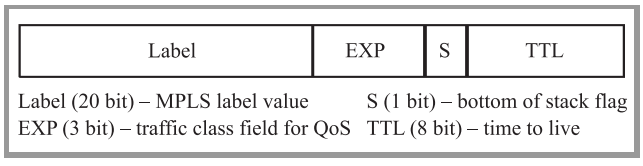


Fig. 2. MPLS header (4 Bytes length).

For this reason we defined mapping between DSCP code values and EXP values of the MPLS header. Method of mapping DSCP codes for aggregated CoSes into MPLS EXP codes is shown in Table 1 [2]. The table is filled in accordance with the EXP-inferred-PSC LSP (E-LSP) model proposed by the IETF [3]. Each router configures per hop behavior (PHB) rules for packets with different EXP field's values. It is possible to do static mapping in the domain but it should be the same in the whole network.

Table 1
Mapping between DSCP in IP QoS System and PMLS EXP field [2]

Type of application	End-to-end class of service	Class of service in IP QoS project	DSCP	MPLS EXP
Signaling	Signaling	Signaling	101000	101
VoIP	Telephony	Real time (RT)	101110	100
Interactive games	RT interactive		100000	
Video on demand	MM streaming	MM streaming	011010 011100 011110	011
FTP	High throughput data	High throughput data	001010 001100* 001110*	010 001*
	Standard	Standard (STD)	000000	000

* DSCP codes and MPLS EXP field used for HTD class of service.

In the project we assumed that the capacity of all the links in the core network is divided into different classes of service. This division is determined by the RMS module, according to the maximum allocation model (MAM) method

described in [4]. The main advantage of the model is its simplicity and, in turn, the model ensures the achievement of isolation between traffic.
ROMAN module performs the following functions:

- retrieves information about the network topology,
- sets required capacity (C_{QoS}) for proper classes of service,
- establishes MPLS TE tunnels,
- provides information about the topology and TE tunnels to the RMS module.

We assume that the input data for ROMAN module is information about current network topology, link capacity and required capacity for CoSes stored in the database. Module retrieves information from one of the router of the network, since all the routers in the network use the OSPF routing protocol [5] and gather information about network topology. In our implementation it is possible to load the status of the network from extensible markup language (XML) configuration file. Due to resource dimensioning model, the value of required capacity for proper CoS cannot be greater than constraint (1):

$$C_{QoS} \leq \frac{C_{\min}}{L_{RD}-1}, \tag{1}$$

where:
$C_{\min}$ – minimal access link capacity,
L_{RD} – number of edge routers.

CISCO routers use PCALC algorithm for MPLS TE tunnels establishment. This algorithm discovers the paths in the network and provides data for explicit route object (ERO) field used in RSVP-TE signaling structure. The proposed solution implemented and tested different routing algorithms for TE tunnels configuration, therefore replacement of PCALC algorithm was mandatory. In addition to standard algorithms such as Dijkstra and Kruskal, additional algorithms have been implemented like extended Dijkstra's described in [6] and self-adaptive multiple constraints routing algorithm (SAMCRA) described in [7], [8]. ROMAN configures TE tunnels by entering the command: *ip explicit-path* with specified intermediate nodes. After configuring the tunnel, the module writes information in the database and provides list of tunnels to the RMS module. Additional implementation details are presented in Section 3.

2.2. RMS Module

RMS module is responsible for implementing the algorithm for resource dimensioning and allocation, depending on available resources and demands in the domain. Performing these tasks requires communication with the ROMAN and PD-PE modules. In the prototype we implemented a simplified static version of resource allocation algorithm.

The primary task for RMS module is to determine the capacity allocated for each class of service in every edge router (ER) of the domain. This capacity is used by call admission control function implemented in the TRC-FE module. In addition, the RMS module sets the buffer size in appropriate port for respective classes of service in accordance to [2].
RMS module receives notification from ROMAN module about changes in network topology. The notification itself does not provide information about new network topology and is the responsibility of the RMS to achieve this information in a pull mode from the ROMAN. Additionally RMS module can be notified that paths has been changed in network by ROMAN module. Like the previous notification, it does not provide additional data structures, therefore the RMS module retrieves structure with up-to-date paths in the network from ROMAN module.
RMS module allocates bandwidth for all classes of service in a chosen path (MPLS TE tunnel). Additionally module allocates bandwidth for the MPLS tunnels that pass through particular link. In addition, buffer sizes are set in each output port and for each class of service (ports through which at least one path passes). The results obtained from the resource allocation algorithm are used in the admission control algorithm.
To describe the algorithm we introduce the following variables and constants:

$l = 1 \ldots L$, link number ($L \equiv$ number of links),
$s = 1 \ldots S$, path number ($S \equiv$ number of paths),
$k = 1 \ldots K$, CoS number ($K \equiv$ number of CoS),
$C_l \equiv$ capacity of link l,
$\delta_{ls} = 1$ if path number s contains link number l, otherwise $\delta_{ls} = 0$,
$M[k,s] \equiv$ matrix of demands including demands for CoS number k in path s.

In particular, the algorithm allocates resources for the classes of service and convert relative input matrix of demands into absolute matrix of demands describing the exact bandwidth for each path and class of service. When for all classes and paths are the same demands then input elements of the matrix are equal to 1.
After running the algorithm, the output matrix contains the bandwidth requirement for each path and class of service. Then, we calculate the value of bandwidth C_{kl} for router output port of link l and class of service k according to the formula:

$$C_{kl} = \sum_{s} M[k,s]\ \delta_{ls}\,. \tag{2}$$

After calculating the bandwidth values for different classes of service we calculate buffer sizes. The length of the buffers allocated for each class depends on the used routers. For example, Cisco routers used in the prototype allowed to work with no more than 64 packets total buffer size for all classes of service. The results presented below take into account this limitation.

Signaling class: Resource dimensioning for signaling traffic is quite complicated. Recent studies showed that a single procedure call statement in the exemplary architecture of

next generation networks needs around 30 kbit/s. If we have reserved bandwidth C_{SIG} for signaling traffic, then we can allow C_{SIG}/30 connection set-up procedures. One procedure connection request, sends simultaneously N messages to the network. In order not to loose signaling packets, the buffer size for signaling CoS should be calculated according to the following formula:

$$B_{SIG} = \frac{C_{SIG}\ [\text{kbit/s}]}{30\ [\text{kbit/s}]}\ N\ [\text{packets}]\,. \tag{3}$$

RT Interactive class: Buffer size for this class must take into account maximum values of delay variation (IPDV) according to the following formula:

$$IPDV_{RT}\ [s] = \frac{B_{RT} \times d_{RT}\ [\text{B/packet}]\ 8\ hop}{C_{RT}\ [\text{kbit/s}]}\,, \tag{4}$$

where:

d_{RT} – the largest packet length of all RT streams,
hop – the number of hops on the longest path,
B_{RT} – buffer size for RT class,
C_{RT} – allocated bandwidth for RT class.

MM streaming and HTD classes of service require small packet loss [2]. Therefore, the buffer size should be large enough to minimize the number of lost packets belonging to these classes of service. On the other hand, while there is no requirement for delay variation the requirement for packet transfer delay (IPTD) is 0.5 s for end-to-end delay [2].
The delay and packet loss levels depend on the load (ρ). Only, we can calculate the maximum buffer size for the delay when the buffer is always full ($\rho \to 1$). Then, the buffer can be calculated using the following formula:

$$IPDV_{MMS/HTD}\,[s] = \frac{(B_{MMS/HTD}+1)\,d_{MMS/HTD}\ [\text{B/packet}]\,8\,hop}{C_{MMS/HTD}\,[\text{kbit/s}]}\,, \tag{5}$$

where:

$d_{MMS/HTD}$ – maximum packet size of all streams MMS or HTD,
hop – the number of hops on the longest path,
$B_{MMS/HTD}$ – buffer size for MMS or HTD,
$C_{MMS/HTD}$ – allocated bandwidth for MMS or HTD.

For the values based on IPTD, the length of the buffer is over-dimensioned ($\rho < 1$). If the buffer size is too small, then we choose a larger buffer size in our implementation (100 packets – as a minimum value).
The input data related to network topology and routing are passed from ROMAN module to RMS module. RMS node can be configured with the appropriate entries in the configuration files. Configuration of QoS requirements is stored in the file "qos.txt", while the value of demands are stored in the file "demands.txt". This file configures the demand for specific paths and specific classes of service: signaling, real time, MM streaming, high throughput data and standard. Demands are expressed as the weights, and allocated capacity is directly proportional to the stated weight. In another configuration file are stored quality of service requirements for different classes of service (Table 2). The file for the relevant class defines the following metrics: IP packet loss ratio (IPLR), IP packet transfer delay (IPTD), IP packet delay variation (IPDV) and packet length d. Additional implementation details are presented in Section 3. Table 2 presents the necessary data of CoS for configuring RMS.

Table 2
The requirements on the quality of service [2]

Number	Class	IPLR	IPTD	IPDV	d
1	Signaling	X	X	X	–
2	Real time	X	X	X	X
3	MM streaming	X	X	–	–
4	High throughput data	X	X	–	–
5	Standard	–	–	–	–

2.3. PD-PE Module

PD-PE module participates in routing process in the domain in the following way. It receives from the ROMAN module information of customers attached to given routers. This function is performed by configureAccessNetworks() method in interface RomanToPd. Current list of customers addresses belonging to the routers overwrites the old one. Function requestAccessNetworks() in PdToRoman interface request list of customer addresses in routers.
PD-PE module provides information for the call admission control function and resource allocation process to the routers, in order to configure the interfaces. Both functions are triggered by the RMS by using configureResources() and configurePorts() functions in RmsToPd interface. During the network resources monitoring process, PD-PE module transmits reports through reportResourceState() function in PdToRms interface.

2.4. PE-PE Module

PE-PE module manages network devices' configuration, which is the final element for configuring the router output interface. For this purpose the interface PdToPe has configurePorts() function that provides router interface configuration. This configuration includes resource allocation (bandwidth and buffer size) for particular class of service. PE-PE module is the last element of the signaling chain for resource allocation. Module communicates with the network devices. Anyway, this communication is not standardized and is specific to each network which actually acts as a driver for the IP QoS System.
For appropriate work of PE-PE module, a database is created containing the following tables: routers, interfaces, class_configurations, pe_points, access_lists, policers, shapers and session. The first four tables provide information about topology of testbed network. It should be noted

that the contents of both interfaces and class_configurations tables will vary depending on the dimensioning of network resources. Subsequent tables access_lists, policers, shapers and session will be filled during admission control process. In routers table is stored basic information about routers in testbed network. Since in the testbed network we implement 2 border routers and 4 core routers, then the routers table contains 6 records, one for each router. Each of these routers have an identifier that is used as a foreign key for accessing the table. The fields username, password and password_enable contain the data necessary to establish a telnet session with the routers. Field ip contains the IP address of virtual loopback interface, which is defined on each router and used as an identifier of the router in the testbed network.

Class_configuration table contains data about classes of service provided in the system. This table contains the following fields: name – the name of the CoS, bitrate – bit rate dedicated to the class on the output link [bit/s] and queue_limit – the queue size for the class. Moreover, the interface_id is a foreign key, which represents the identifier of the network interface. For each interface are defined five classes of service: real time, MM streaming, high throughput data, standard and signaling.

Table Pe_points contains data of all edge nodes in the network. Moreover, the database module of PE-PE contains the tables: policers, shapers i session, which are filled during the per-flow operation of IP QoS System.

2.5. TRC-PE Module

TRC-PE node runs the call admission control algorithm. For this purpose in PdToTrc interface we distinguish the method configureResources() that provide description of resource allocation for each class of service. After starting this method, the following actions are performed:

- TRC-PE module finds in the database points that realize call admission control. In case the point is not found, the method returns a negative result.
- Resource configuration take into account bandwidth, buffer size, packet loss ratio, packet transfer delay and packet delay variation. Based on these parameters is provided maximum load value, which is determined by an call admission control algorithm. This value is stored in the database. If the call admission control algorithm is not supported, the method returns a negative result.
- After the proper run the algorithm returns a positive result.

During configuration of the TRC-PE module was set up a database named pbz_trc. This database stores data about available resources for each class of service on IP QoS System and for all edge routers in the testbed network. Table trc_points contains information for all edge routers (routers ER1 and ER2). Moreover trc_resources table stores information about the classes of service for all the routers specified in the table trc_points. While system is working these tables are filled with records that store data about the current sessions and flows in the testbed network.

3. Implementation of IP QoS System Modules

ROMAN module has been implemented in C# and runs on a virtual MONO platform on Suse Linux. Communication with other modules is provided by the interface using the ICE library. Figure 3 depicts ROMAN module divided in functional blocks.

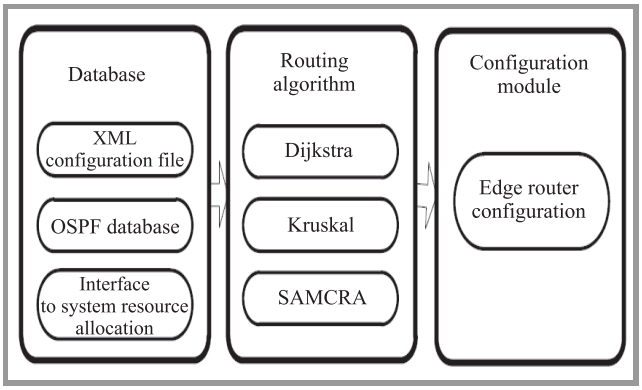


Fig. 3. ROMAN module divided in functional blocks.

The main functionality is included in the library of routing algorithms. This module determines paths between each pair of edge routers for each Class of Service. To find a path between two routers we implemented Dijkstra, Kruskal and SAMCRA algorithms.

To determine the routing topology of the network ROMAN reads OSPF table from one router of the testbed network. This is achieved by using the database library. For testing purposes we can load the network topology from the XML file. In addition, the implemented module provides information to other modules about network topology and established TE tunnels. This functionality of the ROMAN module is used by RMS and PD-PE modules. The interfaces with other modules have been defined using the SLICE language. Figure 4 shows the data structure used by the interfaces.

Another function of ROMAN is to force routing in the network by properly configuring the routers. This is done by sending commands to the edge routers forming the MPLS TE tunnel, giving an ordered list of routers through which the path passes.

RMS module has been implemented in C++. The implementation uses the standard C++ libraries and the ICE library version 3.3.0 for C++. Figure 5 shows the sequence of messages between ROMAN, RMS, and PD-PE modules during the update of paths and network topology.

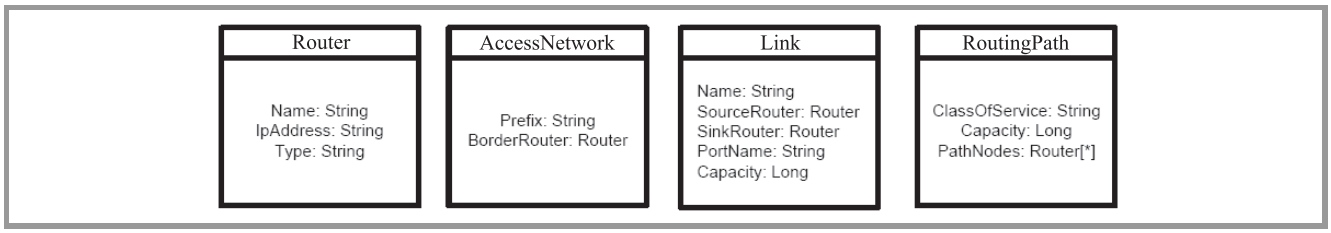


Fig. 4. Data structures used by ROMAN module.

The RomanToRms interface defines the methods topologyHasChanged() and routingPathsHaveChanged() which provide information from ROMAN to RMS module to update network topology and paths, respectively. Moreover, the RmsToRoman interface defines the methods requestTopology() and requestPath() providing download of current network topology and paths.

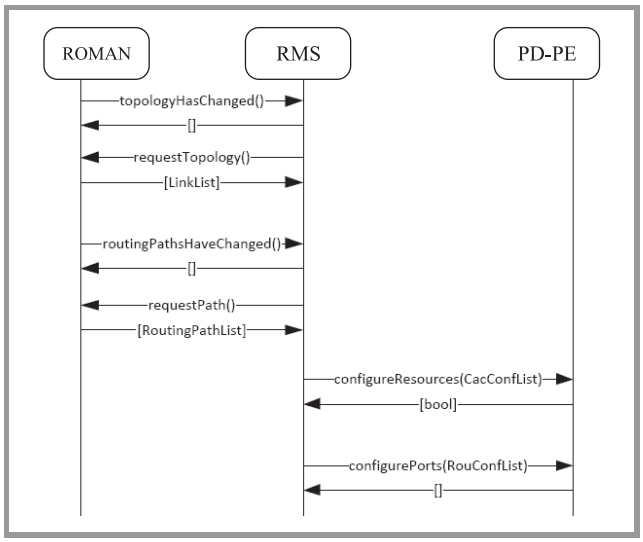


Fig. 5. Sequence of messages – update of paths and topology.

After RMS module receives the new data, it starts call admission control function that calculates the new limits and sets up router output ports. Subsequently, new data are sent to the PD-PE node using the methods configureResources(CacConfList) and configurePorts(RouConfList). CacConfList structure contains the configuration of edge routers for the CAC function (calculated on the basis of the capacity). RouConfList structure contains the configuration used for output ports of the routers.

The RmsToPd interface defines the methods configureResources(CacConfList) and configurePorts(RouConfList) which enable the configuration message from RMS to PD-PE module; specifically, CacConfList contains data for CAC configuration and RouConfList for output ports configuration.

PD-PE, TRC-PP and PE-PE modules were written in Python language. For implementation was used ICE library version 3.3.0 in Python library: readpool, SQLAlchemy, psycopg2, database PostrgreSQL version 8.3.

4. Testbed Network

The described modules of the IP QoS System have been installed in the testbed network. Figure 6 shows the topology of the testbed network in which laboratory tests are performed. Preliminary tests showed that the implemented modules work together correctly and properly configure the network mechanisms for providing DiffServ architecture.

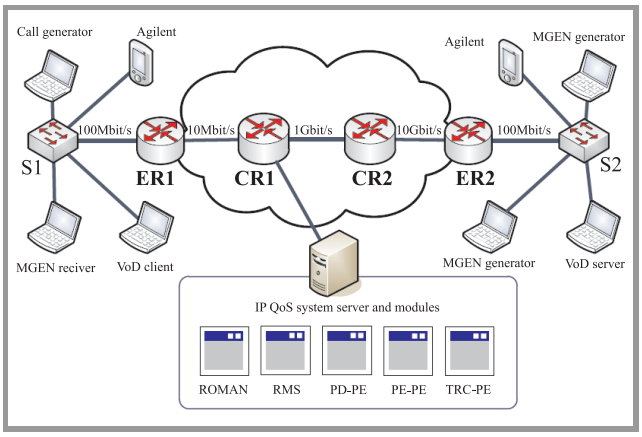


Fig. 6. Testbed network with implemented modules.

The implemented IP QoS System correctly configures the testbed network, which is able to guarantee the QoS parameters set for the proper classes of service. In the following text we expound some issues related to network configuration by the PE-PE and ROMAN modules. Please note that only the ROMAN and PE-PE modules configure the network mechanisms in the network devices. The other modules in the IP QoS System allow proper operation of the whole system.

4.1. MPLS Path Configuration in Testbed Network by ROMAN Module

The implemented software of IP QoS System is tested and demonstrated in the laboratory network as described in [9], [10]. ROMAN requires that edge and core routers

had been pre-configured to send MPLS traffic. In particular, it should be possible to implement OSPF routing and MPLS on the physical interfaces:

 # interface < configured interface name e.g. "GigabitEthernet0/1" >
 # mpls ip
 # mpls traffic-eng tunnels
 # ip rsvp bandwidth < interface bandwidth > < flow bandwidth >

The next step is to create a tunnel between a pair of boundary routers:

 # interface Tunnel1
 # description tunel1
 # tunnel destination < loopback interface ip address of router terminating the tunnel >
 # tunnel mpls traffic-eng path-option 1 explicit name <path name>
 # tunnel mpls traffic-eng record-route
 # no routing dynamic

Each router in the laboratory network has IP addresses associated with physical interfaces and logical address of the router, which is unique throughout the network (Loopback0 logical interface). ROMAN uses Loopback0 interface addresses for configuring MPLS tunnels.
In the laboratory ROMAN application runs in the initial phase of network configuration. This module detects network topology, discovers paths from routing and sets MPLS tunnels. Configured topology with MPLS paths are sent to the RMS module.

4.2. Configuration of the Monitoring Mechanism in Testbed Network by PE-PE Module

Monitoring mechanisms in testbed network are implemented by a single token bucket for real time class of service. To configure this mechanism we should set peak rate value with burst size value [11]. Conforming packets are marked with proper DSCP value, whereas exceeding ones are rejected. Sample commands for configuring Cisco routers mechanism for version 12.1 are listed below:

 # configure terminal
 # ip access-list extended <rule name>
 # permit UDP host <source IP address> eq <port number> host <sink IP address> eq <port number>
 # exit
 # interface <router interface that will be configured>
 # rate-limit output access-group <rule name> <peak rate> <burst size> <burst size> conform-action set-dscp-transmit <number DSCP> exceed-action drop

As we can see from the above list, first we define the group to which we assign the UDP stream between two routers and ports in the network. Then we configure the router interface properly to indicate conforming packets (compatible with token bucket mechanism) with proper DSCP value.

4.3. Configuration of the Scheduling Mechanism in the Testbed Network by PE-PE Module

The first step is the definition of the classes of service. For the exemplary real time CoS, the instructions would be as follows:

 # class-map PbzRealTime
 # match dscp ef cs4

Next, the router must be configured (example for real time class configuration) [11]:

 # policy-map <policy name>
 # class PbzRealTime
 # priority < bit rate allocated to the class of service >
 # exit
 # exit
 # interface < router interface that will be configured >
 # service-policy output <policy name>
 # hold-queue < total buffer size allocated to router interface > out

The mechanisms used in the testbed correspond to these ones which are accessible by Cisco routers. The presented commands are intended to illustrate how we use router mechanisms in the testbed network. It should be noted that implemented IP QoS System can automatically configure a testbed network in accordance with the requirements of guaranteed quality of service.

5. Summary

During the project we carried out preliminary tests of functionality and cooperation of IP QoS System modules. These tests confirmed the correct implementation and cooperation of the modules described in this article. During the tests, we examined not only network configuration, but also the performance of implemented system.
IP QoS System was presented at the conference Krajowe Sympozjum Telekomunikacji i Teleinformatyki 2010 – KSTiT 2010. The exhibition demonstrated the performance of implemented IP QoS System with all modules, calls generator and network analyzer. The exhibition presented a test VoD application with QoE Telchemy analyzer and test VoIP application with MOS Agilent analyzer.
The described implementation of IP QoS System shows that the proposed solution could be used by network operators. However, it should be noted that the implemented system is designed for research and not for commercial purposes. For this, a solution for commercial purposes should be written from the beginning, in order to ensure not only correct work but, especially, effective performance of the system.

At last, let us remark that the proposed solution is limited to single domain network and it does not take into account different network access technologies. The proposed system in future studies could be extended by a further element like access networks such as, e.g., wireless network 802.11 standard.

References

[1] "An Architecture for Differentiated Services", RFC 2475.

[2] H. Tarasiuk, W. Burakowski, A. Jajszczyk, and R. Stankiewicz, "Specyfikacja algorytmów i mechanizmów sterowania ruchem na poziomie pakietów w sieci IP QoS", Raport PBZ-MNiSW-02-II/2007/WUT/B.2/B.6, 2009.

[3] "Multi-Protocol Label Switching (MPLS) Support of Differentiated Services", RFC 3270.

[4] "Maximum Allocation Bandwidth Constraints Model for Diffserv-aware Traffic Engineering", RFC 4125.

[5] "OSPF Version 2", RFC 2328.

[6] P. Pyda, T. Dalecki, "Realizacja rutingu w sieci IP QoS", *Przegląd Telekomunikacyjny i Wiadomości Telekomunikacyjne*, zeszyt 8–9, s. 1922–1923, 2009.

[7] P. Van Mieghem , H. De Neve, and F. A. Kuipers, "Hop-by-hop Quality of Service routing", *Comput. Netw.*, vol. 37. no. 3-4, pp. 407–423, 2001.

[8] F. A. Kuipers, "Quality of service routing in the Internet: Theory, complexity and algorithms", Ph.D. thesis, Delft University Press, The Netherlands, 2004.

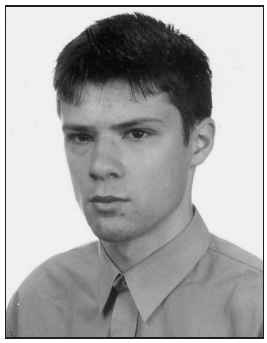
[9] H. Gut, W. Latoszek, M. Gajewski, J. Saniewski, E. Niewiadomska-Szynkiewicz, T. Wiśniewski, P. Arabas, and M. Rotnicki, "Specyfikacja i instalacja sieci laboratoryjnej IP QoS", Raport PBZ-MNiSW-02-II/2007/WUT/B.8, 2009.

[10] H. Gut, W. Burakowski, W. Latoszek, P. Gielmuda, J. Śliwiński, H. Tarasiuk, W. Góralski, A. Bęben, and P. Krawiec, "Integracja oprogramowania i instalacja w sieci laboratoryjnej IP QoS – część II", Raport PBZ-MNiSW-02-II/2007/WUT/D.6, 2010.

[11] W. Burakowski, J. Śliwiński, A. Bęben, H. Tarasiuk, P. Krawiec, "Implementacja modułu do wspomagania konfiguracji sieci", Raport PBZ-MNiSW-02-II/2007/WUT/D.4, 2010.

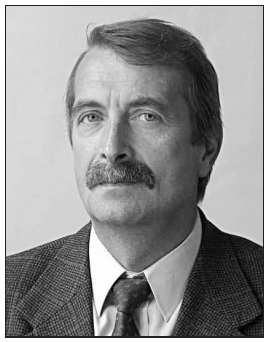
Tomasz Dalecki received the M.Sc.Eng. degree from the Warsaw University of Technology in 2002. Since 2003 he works in Military Communication Institute. His research interests cover management systems design and implementation and security in IP-based systems.

E-mail: t.dalecki@wil.waw.pl
Military Communication Institute
Warszawska st 22A
05-130 Zegrze Płd., Poland



Waldemar Latoszek was born in Otwock, Poland, in 1981. He received engeneer degree from Warsaw University of Technology in 2005. Since 2005 he works in National Institute of Telecomunication. His research interests cover network architectures, testbeds, traffic control.

E-mail: w.latoszek@itl.waw.pl
National Institute of Telecommunication
Szachowa 1
04-894 Warsaw, Poland



Henryk Gut was born in village Zub-Suche, not far from Zakopane, in 1951. He received M.Sc. degrees in Telecommunications from Warsaw University of Technology in 1975. Directly after graduation he began working for National Institute of Telecommunication, where he is employed until now. During working for NIT he was dealing with following topics: technical structure and topology optimization of data transmission network; methods and systems for bit, frame and clock synchronization; modeling of primary bit error processes in binary data channels; designing of terminal and network equipments for national data network SYNCOM and paging network POLPAGER. In period 1998–2006, his research and development activity was focused on utilization power lines and in-home electrical installations as an transmission medium for broadband access and in-home data networks. Recently he is participating in realization two national grate projects: "Next Generation Services and Networks – technical, application and market aspects. Network Traffic Management – IP network with full QoS guarantee" and "Future Internet Engineering". He is author or co-author of about 35 papers published in national journals and conference proceedings and is co-author of a few dozen internal technical reports of NIT.
E-mail: h.gut@itl.waw.pl
National Institute of Telecommunication
Szachowa 1
04-894 Warsaw, Poland

Witold Góralski – for biography, see this issue, p. 20.

Piotr Pyda, Jordi Mongay Batalla – for biographies, see this issue, p. 11.

Jarosław Śliwiński – for biography, see this issue, p. 10.