

An End-to-End QoS Framework with Self-Adaptive Bandwidth Reconfiguration

A. Peculea, B. Iancu, I. Ignat, V. Dadarlat, Z. Baruch, E. Cebuc
Technical University of Cluj-Napoca, Computer Science Department
26-28 Gh. Baritiu St., 400027 Cluj-Napoca, Romania

{Adrian.Peculea, Bogdan.Iancu, Iosif.Ignat, Vasile.Dadarlat, Zoltan.Baruch, Emil.Cebuc}@cs.utcluj.ro

Abstract – This paper presents the design and implementation of an end-to-end QoS framework with self-adaptive bandwidth reconfiguration. The application implements two types of routers, edge and interior. Edge routers identify traffic flows, mark the packets with their corresponding type of service, and perform per-flow admission control and end-to-end bandwidth reservation. Both types of routers implement a link state routing protocol and provide class-based service differentiation. The main advantage of this framework is that it provides end-to-end QoS guarantees with less overhead than the other. The application is developed in C# language for .NET platform and represents a part of a complex development tool designed for developing QoS frameworks and technologies such as traffic identification and marking, queue and congestion management, traffic shaping and policy, specific protocols.

Keywords – End-to-End QoS Framework, Development tool, Admission control, Bandwidth reconfiguration, QoS routing

I. INTRODUCTION

In high-speed next generation IP networks, audio conferences, video conferences, VoIP, video telephony, are becoming imperative demands in a technological dependent society. With this high increasing demand, a new technology emerged, called Quality of Services, which ensures that one less critical IP application (e.g. web browsing) does not interfere with another critical IP application (e.g. video conference).

The service represents the traffic performance provided to the client and is characterized by the QoS (Quality of Service) parameters. In order to ensure these parameters, the provider implements QoS using specific techniques. The main frameworks and technologies used for QoS are Integrated Services, Differentiated Services, Multiprotocol Label Switching, On-Demand QoS Path framework (ODP) and traffic engineering. QoS routing represents a part of the last mentioned technology.

In this paper we describe the design and implementation of an end-to-end QoS framework with self-adaptive bandwidth reconfiguration. The ease of the QoS framework testing and the fact that this application will be integrated in a complex development tool designed for QoS frameworks and technologies development represent the main advantages of this system. The application implements a link-state QoS routing protocol. The system implements two types of routers, edge and interior. Edge routers identify traffic flows, mark the packets with their corresponding type of service, place them into their cor-

responding forwarding queues, and perform per-flow admission control and end-to-end bandwidth reservation. Both types of routers implement a link state routing protocol and provide class-based service differentiation.

The rest of this paper is organized as follows: Section II provides background information related to QoS technologies. Section III describes related work by presenting ODP [5] and its disadvantages, shows that the proposed framework is new and original and proposes new approaches to overcome these problems. Section IV describes the End-to-End QoS Framework with Self-Adaptive Bandwidth Reconfiguration. Section V presents further work and concludes the paper.

II. BACKGROUND

QoS represents the network capability to deliver better services for the selected flows over different technologies. The main goal of QoS is to provide priority including dedicated bandwidth, controlled latency and jitter, and improved loss characteristics. Also QoS parameters delivering for one or several flows must not determine significant decrease in performance or even elimination of the other flows. A service represents the traffic performance provided to the client and is characterized by the QoS parameters. Loss rate, delay, delay variation or jitter, network availability and bandwidth are the main QoS parameters. QoS techniques represent the specific methods of implementation, including traffic identification and marking, queue and congestion management, link efficiency, traffic shaping and policy, specific protocols. The service and QoS techniques are different but strongly interconnected, since these techniques are used for service construction.

The main frameworks and technologies used for implementing QoS are Integrated Services, Differentiated Services, Multiprotocol Label Switching, ODP, and traffic engineering. These technologies are detailed described in [1] [5].

Integrated Services have developed a new architecture for resource allocation in order to satisfy the requirements of the real time applications. The main idea is resource reservation for each flow. A flow is identified by five fields in packet headers: source and destination IP addresses, protocol ID and source and destination ports. These five fields are often referred as five-tuple. The aim of the Integrated Services is the preservation of the IP networks datagram based model and in the same time the reservation of the resources for real time applications. In Integrated Services architecture a set of mechanisms and

protocols is used for explicit reservation of the resources. Before packet transmission, the applications reserve the necessary resources along the path. The reservation setting begins with the description of the flow characteristics and necessary resources. The network can accept the new application only if there are available resources. After the reservation establishment, the application can send packets along the reserved path. Integrated Services architecture assumes that delay is the main QoS parameter guaranteed by the network.

Differentiated Services Architecture provides several service levels by classifying the traffic into a small number of forwarding classes and allocating resources based on these classes. The individual forwarding class represents the aggregated traffic and is encoded in the IP packet header. Edge nodes classify the packets and condition the traffic. Interior nodes forward the packets based on the forwarding classes in the packet header. The forwarding treatment is described by per-hop behavior (PHB), each PHB being represented by a 6 bit value named Differentiated Services Codepoint (DSCP). All the packets with the same DSCP are referred as behavior aggregate. IP packet header has an 8 bit field named IP TOS. This field is composed of 3 bits precedence, 3 bits type of service and 2 unused bits. Differentiated Services standard redefines IP TOS field in order to indicate per-hop behaviors. The replacing field, named DS, replaces IPv4 TOS and IPv6 traffic class bytes definitions. First 6 bits of DS field are used as DSCP in order to encode the PHB and the other 2 bits are not used.

Multiprotocol Label Switching (MPLS) presents a series of advantages as QoS support through connection orientation, traffic engineering support, VPN support or multiprotocol support.

Traffic engineering optimizes the performance of the network by reducing the congestion and improving resource utilization through traffic distribution management. Constraint based routing is one of the components of a traffic engineering system. The topic is presented in [2] and [3]. Constraint based routing uses other constraints and requirements besides destination address in routing selection. If the constraints are imposed by QoS requirements we have QoS routing and if the constraints are imposed by policies we have policy routing. QoS routing has two main objectives: to ensure the QoS requirements and to optimize the usage of the network resources. There have been developed several QoS routing algorithms, but this topic is still of great interest. This framework was designed to allow for developing other more performant QoS routing algorithms.

According to the amount of global state maintained, routing protocols can be classified in distance vector or link state. In the case of distance vector routing protocols, routers periodically exchange distance vector routing information with their neighbors and compute the paths using Bellman-Ford [4] algorithm usually. In the case of link state routing protocols, the state of all local links is broadcast to all routers and the paths are computed using Dijkstra algorithm [4] usually.

Search for paths can be performed off-line (pro-active, pre-computation) or on-line (reactive, on-demand). Routers that use the off-line method periodically compute all routes using current information. In the case of the on-

line method the routes are computed when the new traffic demand arrives.

The main QoS routing challenges, as presented in [2], are: stability, robustness and scalability and routing cost. The frequency of routing updates must be a compromise between the traffic and routing overhead and the accuracy of the information.

III. RELATED WORK

In [5] an On-Demand QoS Path framework (ODP) is defined, which provides end-to-end QoS guarantees to individual flows. ODP exercises per-flow admission control and end-to end bandwidth reservation at the edge of the network and only differentiates traffic classes in the core of the network. Also, ODP monitors the bandwidth utilization of the network and performs dynamic bandwidth reconfiguration in the network core. The paper shows that ODP provides end-to-end QoS guarantees to individual flows, thus eliminating the DiffServ drawback, with much less overhead than IntServ.

ODP assumes that two types of routers: edge routers and core routers exist in a network. Edge routers have the task to take admission decision for each incoming individual flow, to map individual flows to different traffic classes and to transmit packets to the network. Core routers are DiffServ routers. Core routers have the responsibility to recognize traffic classes and to provide class-based service differentiation.

ODP uses a hierarchical approach in order to organize link bandwidth. The highest level is represented by the Provisioned Link (PL), each physical link being divided into PLs. The network traffic is divided into several traffic classes, as DiffServ does, and one PL is dedicated to one traffic class. At each provisioned link, a certain amount of its bandwidth is allocated to each of the edge routers in the network, thus further dividing PLs into multiple trunks. There is a one to one mapping between a trunk and an edge router. A trunk (of a given provisioned link) supports flows (of the given traffic class) originating from the same source edge router, regardless of their destination. A source edge router keeps track of available bandwidth of its assigned trunks in the network and performs admission control locally. The admission decisions are instantaneously when a flow with QoS requirement arrives at an edge router.

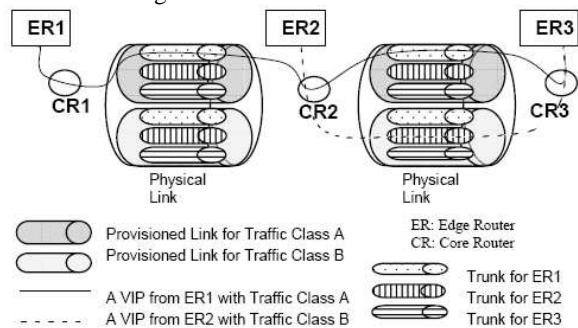


Figure 1. Hierarchical Bandwidth Organization [5]

A path from a source edge router to a destination edge router for a specific traffic class is called a Virtual IP Path (VIP) [5]. It represents a concatenation of multiple trunks

over a source-destination path belonging to the source edge router.

The bandwidth is assumed to be statically assigned to provisioned links based on a long-term traffic prediction made by a central network management server (NMS) or by the network administrator during network initialization. Three approaches are used in ODP to maintain bandwidth utilization at the provisioned link level and make reconfiguration decision: a centralized (Central Control - assumes a central network entity in the network), a semi-centralized (Router-Aided approach - core routers take these decisions) and a distributed architecture (Edge-to-Edge approach -where edge routers maintain bandwidth utilization and reconfigure bandwidth). The amount of bandwidth assigned to each edge router is re-configured on demand basis. ODP permits to dynamically adjust the amount of bandwidth assigned to trunks. Source edge routers can request additional trunk bandwidth or releases unused trunk bandwidth based on the trunk bandwidth utilization. The bandwidth adjustment allows flows in the same traffic class to share the provisioned link bandwidth (regardless of their originating source edge routers and destination edge routers). By periodically examining of the trunk table, a source edge router obtains the bandwidth utilization of its trunks. If the bandwidth utilization of a trunk falls under a predetermined lower threshold, the source edge router adjusts the bandwidth of the trunk by releasing bandwidth, updates the trunk and sends a control message to the network entity (or entities) that maintain(s) the provisioned link table.

The admission technique follows the next steps: if new flow arrives at a source edge router, first a VIP table examination is required in order to identify a VIP to the destination edge router. A second step is the examination of the trunk table to obtain the bandwidth utilization of each trunk on the VIP to the destination edge router. Here we have two cases: if the VIP has enough bandwidth for the incoming flow, the source edge router accepts it and makes end-to-end bandwidth reservation for this flow by updating the bandwidth field in the local trunk table at each trunk on the chosen VIP; b. if the VIP does not have enough bandwidth for the incoming flow, the source edge router will request additional bandwidth and function of the response that it will receive, the flow will be admitted or rejected.

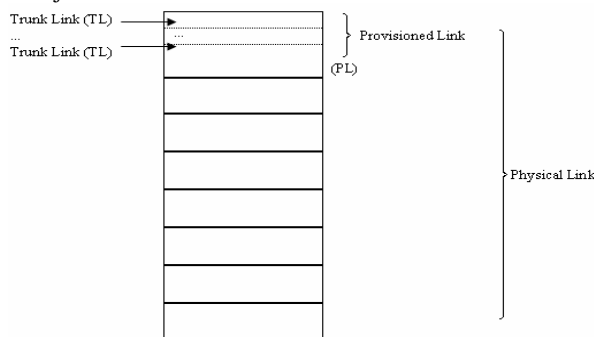


Figure 2a. ODP bandwidth allocation

The main disadvantages of ODP [5] are: (1) the bandwidth adjustment is only local to the traffic class and doesn't allow for redistribution between classes, (2) there

is a need for minimum service guaranties, because after the redistribution of bandwidth, the trunk that released the excess bandwidth will not be able to go back to the initial bandwidth state until the traffic on the trunks that reserved the additional bandwidth slows down, (3) network overhead, due to the framework's messages passing, (4) increased admission time when bandwidth needs to be reserved for trunks.

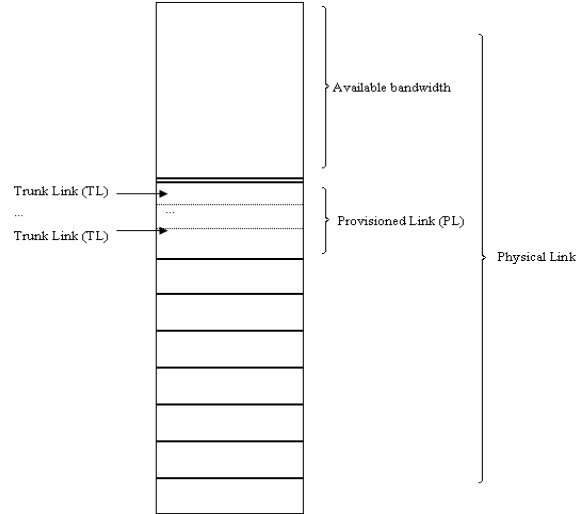


Figure 2b. Proposed bandwidth allocation

To overcome all the above drawbacks mentioned, we propose to develop a QoS framework with self-adaptive bandwidth reconfiguration, which provides minimum service guaranties and allows for bandwidth redistribution between classes. In Fig. 2a and 2b a graphic comparison is presented between ODP's bandwidth allocation and reconfiguration and the proposed solution. The framework also reduces the network overhead by incorporating all the framework's messages in the routing protocol. The problem of increased admission time when bandwidth needs to be reserved for trunks, is resolved thorough the use of the available bandwidth part, which will be descried in the next section. If the used bandwidth for a trunk reaches an upper threshold, the trunk will automatically acquire additional bandwidth.

IV. CURRENT WORK

The current work is concentrated in developing the main functionalities for the QoS framework with self-adaptive bandwidth reconfiguration and also in researching new techniques, to overcome the drawback stated in the previous section.

As previously showed in Fig. 2b the paper proposes a new bandwidth allocation and reconfiguration technique, which allows for minimum service guaranties. The drawback in ODP approach is that the traffic classes use all the bandwidth (see Fig. 2a). The bandwidth redistribution is done only between the trunks of the same provisioned link and does not allow redistribution between classes. If a PL is not intensively used it cannot put the available bandwidth at others PLs disposal.

A second inconvenient is represented by the fact that the trunk that releases unused bandwidth will not be able

to reclaim it until the traffic slows down, and other trunks start to release bandwidth, thus not being able to provide a minimum service guarantee for that specific trunk in the traffic class.

In our approach, the physical link is divided into two main sections: an available bandwidth part and the provisioned links part - needed for the traffic classes. By having two separate sections, the framework guarantees a minimal service level for each trunk and offers a common bandwidth that can be used by every trunk, regardless of their traffic class.

If the used bandwidth of a trunk exceeds an upper threshold, the trunk will acquire additional bandwidth from the available bandwidth section. When the acquired bandwidth is no longer needed the trunk releases it, so that other trunks can benefit from it. The framework will implement equitable bandwidth access mechanisms to all PLs and trunks, so that no trunk will monopolize the section inducing a starvation problem to the others.

The network overhead, due to the framework's messages passing of ODP is solved by incorporating all the framework's messages in the routing protocol, thus avoiding adding additional traffic to the network.

The problem of increased admission time is resolved, due to the trunk's capability to automatically acquire additional bandwidth.

The proposed development tool implementation (Fig. 3.), designed for developing QoS frameworks and technologies, is composed of (1) a network interface module that contains a packet capturer, analyzer and forward system, (2) a routing application and (3) an inter-router communication protocol. The packet capturer, analyzer and forward system's role is to determine the available network interfaces, to capture all the packets that flow through the network, to analyze them, to collect the required information and, also, to forward the incoming traffic to the appropriate interfaces. The network interfaces have to work in promiscuous mode, thus capturing all the packets. Once the system has the capability to capture, analyze and forward traffic, the routing application will create the network topology, compute the metrics and then build the routing tables, used to send data through the best available path. The communication protocol helps the process of routers' discovery and allows them to exchange network topological information.

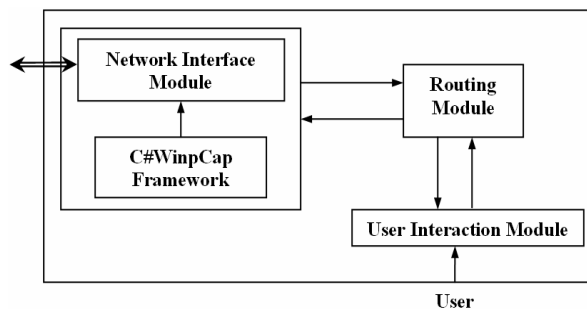


Figure 3. System architecture

Firstly, the inter-router communication protocol will be described and further, the development tool architecture will be presented.

The inter-router communication protocol uses six types of custom ICMP control messages: advertise, advertise acknowledge, infoLSDB, newLSDB, delay and delay-Timer. The structure of the ICMP messages is presented in Fig. 4.

General ICMP packet format:

Type	Code	Checksum	Data			
Advertise:						
9	0	Checksum	Router ID			
Acknowledge:						
9	1	Checksum				
Delay:						
9	2	Checksum	ST _i	RT _j	ST _j	RT _i
InfoLSDB:						
9	4	Checksum	LSDB Data			
NewLSDB:						
9	5	Checksum	Router ID	LSDB Data		
DelayTimer:						
9	10	Checksum	ST _i	RT _j	ST _j	RT _i

Figure 4. Message types

where, the Type and Code fields from the ICMP header are used to label the control messages, and the data field stores all other required information.

A newly started router has no knowledge of the network topology. It uses the *Advertise* message to send its ID to neighboring routers (Fig. 5.). Any directly connected router will answer with an *Acknowledge* message, after adding the sender's router information in its own topological database, called LSDB (Link-State Database). If the router does not receive an *Acknowledge* message, will assume that a host is connected to him and will mark this fact in its own LSDB. Upon the receiving of the first acknowledge packet, the newly started router must perform a synchronization step. This is useful to compute accurate delays - one of the metrics used by the routing application. Only synchronized routers can participate in further actions.

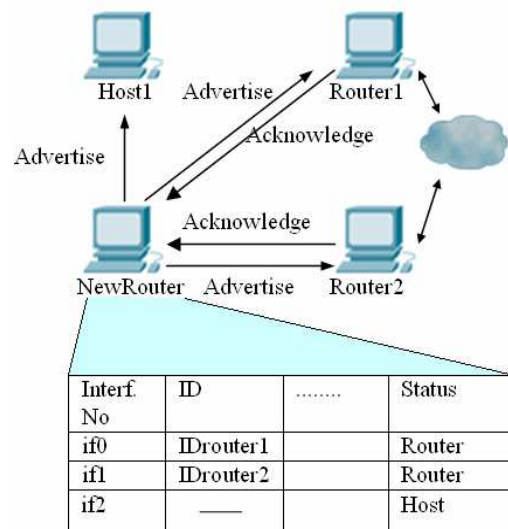


Figure 5. Inter-router communication

Two types of messages are used to compute delays: *Delay* and *DelayTimer*. The difference between them is the moment when they are used. The *Delay* message is used for building and maintaining the routing tables and the *DelayTimer* message is called after a predefined pe-

riod of time to recalculate the delays that could have possibly changed, due to traffic congestion on the communication lines. In order to exchange topological information, *InfoLSDB* and *NewLSDB* messages are used. The first message is sent by the neighboring routers to the new router and contains their current LSDBs. The *NewLSDB* message is used by all the routers to announce any alteration in the current topology.

The development tool designed for developing QoS frameworks and technologies uses a component based architecture, consisting of two main modules: (1) a network interface module, used for the detection and handling of the network devices, and also for packets' control - using a WinCap wrapper for C# .NET; (2) a routing module, which is the core of the application, having the complex task to create and maintain the routing tables, mark traffic packets, put them on queues, determine the output interface and finally, forward the traffic. It also has the responsibility of per-flow admission control and end-to-end bandwidth reservation.

The user interface module, presented in Fig. 6, is a console application, and not a GUI, thus reducing at minimum the user's interaction.

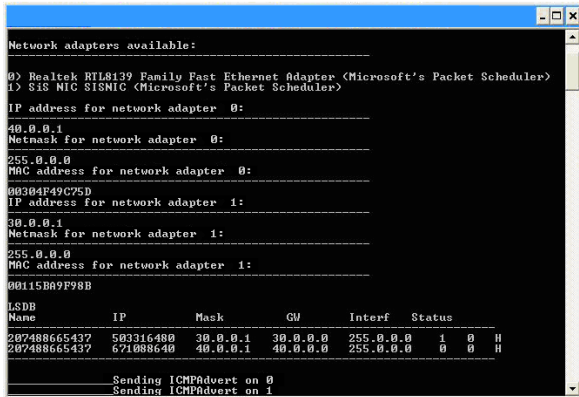


Figure 6. User interface module

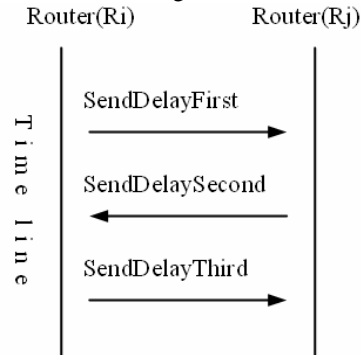
The network interface module uses our C#WinCap framework - a reusable standalone application, which is a wrapper over WinCap, the industry-standard tool for link-layer network access in Windows environments [6]. WinCap functions allow applications to capture and transmit network packets bypassing the protocol stack. The C#WinCap framework interacts with the system providing specific tasks: (a) network device detection; (b) acquires the properties of the network adapters, such as: number of adapters, adapter's name, IP addresses, network masks; (c) provides functions to manipulate a network adapter: open, close, capture packets (with a promiscuous option), send packets, create and analyze traffic packets.

The routing module is the main component of our system. It has a complex task in traffic flow management, end-to-end bandwidth reservation, and in routing table's management. The communication between routers is necessary for routers' discovery, topological information exchange, delays computation, packet marking, packet forwarding and admission control and bandwidth management. To achieve this task, inter-router communication will use the previously defined ICMP messages.

A router's first step is to make it self known to the other routers in the network. It achieves this by sending, at start up, an *Advertise* message to its directly connected routers using the predefined multicast IP address 224.0.0.7. All the routers that receive this message will add the source address and the routerID of the packet in their LSDB. They will also respond with an *Advertise Acknowledge* message. When the newly started router receives the message, it will synchronize with the sender using a SNTP [7] client application. The synchronization is performed only once and it is an important step, thus achieving a total synchronization of all the routers.

The development framework uses the bandwidth and delay metrics for QoS routing algorithms testing, however in the case of the end-to-end QoS framework with self-adaptive bandwidth reconfiguration this metrics are not applicable. For it we consider, as in the case of ODP, static routes, computed based on the number of hops metric.

The next step is to compute the measurement metrics: delays and bandwidths, which are being used for the development and testing of new QoS frameworks and technologies. The delay is computed on all the lines connected to other routers. The application uses a simple, yet efficient algorithm, based on a "triple hand-shaking" technique, illustrated in Fig. 7.



SendDelayFirst:

TpRi	-1	-1	-1
------	----	----	----

SendDelaySecond:

TpRi	TsRj	TpRj	-1
------	------	------	----

SendDelayThrid:

TpRi	TsRj	TpRj	TsRi
------	------	------	------

Figure 7. Delay Measurement Technique

The delay is computed using the following formulas:

$$d_{ij} = TsRj - TpRi \text{ and } d_{ji} = TsRi - TpRj$$

where d_{ij} - delay from routeri to routerj

d_{ji} - delay from routerj to routeri

Ri - the new router

Rj - the old router

Router Rj, after computing d_{ji} , will send to Ri an *InfoLSDB* message containing his LSDB. Ri will wait to receive a number of *InfoLSDB* messages equal to the number of *Acknowledge* received. At this moment the new router has the most recent network topology. Using a *NewLSDB* message Ri will send his topological information to the routers he knows about, and so all of them are

updated with the latest information. To avoid routing loops a split horizon method is used [8].

The next step requires all routers to compute their routing tables. The application is using Dijkstra's shortest path algorithm to generate the minimum cost paths to all hosts, based on the measured metrics (costs).

Another important aspect is traffic flow identification and marking. Two types of routers were defined: edge routers (ER) that are connected to hosts and interior routers (IR) connected only to other routers. Only ERs will mark packets, IRs' purpose is simply to route the traffic accordingly to their routing tables.

For traffic marking, the most significant three bits from the TOS (Type Of Service) field of the IP header will be used, therefore creating eight classes of priority. Each of these classes will be designated a queue, where the packets that need to be forwarded are placed. The traffic is classified based on two main criteria: transport protocol and port number. A flow admission table is created at the edge routers, where all the flows that are proper with the bandwidth requirements are stored.

Another important step, which it is used for the testing and development of new QoS technologies, is the routing table update operation, where routers use a timer to recalculate, at predefined periods of time, the metrics computed previously. If there are changes in these parameters a *NewLSDB* message will be sent, so that all routes learn about this change and recompute their routing tables.

The current developed framework was tested using a looped topology composed of routers and hosts. This specific topology was used to test the split horizon method and also to verify the behavior of the routing algorithm. The routing tables were updated as a result of the recomputed link-state metrics, therefore demonstrating the correctness of the implementation.

V. FURTHER WORK

In this paper is described the design of an end-to-end QoS framework with self-adaptive bandwidth reconfiguration and the implementation of a development tool designed for developing QoS frameworks and technologies.

The end-to-end QoS framework with self-adaptive bandwidth reconfiguration overcomes the disadvantages of ODP by providing minimum service guarantees and bandwidth redistribution between classes, solving the network overhead by incorporating all the framework's messages in the routing protocol and also, the increased admission time problem is resolved, due to the trunk's capability to automatically acquire additional bandwidth.

The development tool uses a component based architecture, consisting of two main modules: the network interface module that avails the C#WinPcap framework facilities, for the detection and handling of the network devices, and also for traffic flows control. Second, the routing module, which is the core of the application, having the complex task to create and maintain the routing tables, identify traffic flows, mark traffic packets, put them on specific queues, determine the output interface and finally, forward the traffic. It also has the responsibility of per-flow admission control and end-to-end bandwidth reservation.

A communication protocol between routers was also defined, which is based on several custom multicast and unicast type of packets. One of the main advantages of this application is the ease of the QoS framework testing, because of the modular design approach. The application developed at this point was successfully tested and the results proved the correctness of the implementation.

The testing and evaluation of the QoS framework with self-adaptive bandwidth reconfiguration will use the defined metrics in [5]: blocking rate - the ratio of the number of blocked flows to the number of all arriving flows; signaling overhead - measured in the total number of control messages transmitted during the entire simulation duration; average connection setup time- the average time interval from when a flow arrives at a source edge router to when the admission decision is received by the flow; average packet transmission delay - the average time interval from when a source edge router transmits a packet until when the destination edge router receives the packet; average packet transmission delay variance - the variance in the packet transmission delays.

Also, we will test the proposed end-to-end QoS framework with self-adaptive bandwidth reconfiguration with the previous defined metrics using complex network topologies. The framework's expected results are to provide end-to-end guarantees and eliminate the previous mentioned drawbacks of ODP.

VI. ACKNOWLEDGMENTS

This work was supported by the "QAF - Quality of Service Aware Frameworks for Networks and Middleware" type A research project (phase 2007-2008) within the framework "National Research, Development and Innovation Programme" initiated by The National University Research Council - Romania (CNCSIS).

REFERENCES

- [1] Z. Wang, *Internet QoS: Architectures and Mechanisms for Quality of Service*, Morgan Kaufmann, San Francisco, 2001.
- [2] O. Younis, S. Fahmy, *Constraint-based routing in the Internet: Basic principles and recent research*, IEEE Communications Surveys & Tutorials 5 (2003), <http://www.comsoc.org/livepubs/surveys/public/2003/sep/pdf/fahy.pdf>
- [3] W. Sun, *QoS/Policy/Constraint Based Routing*, http://www.cse.wustl.edu/~jain/cis788-9/ftp/qos_routing/index.html
- [4] A. Tanenbaum - *Computer Networks Fourth Edition*, Prentice Hall, 2005.
- [5] Mei Yang; Yan Huang; Kim, J.; Meejeong Lee; Suda, T.; Daisuke, M. - *An end-to-end QoS framework with on-demand bandwidth reconfiguration*, www.ieee-infocom.org/2004/Papers/42_4.PDF, 2004
- [6] "WinPcap: The Windows Packet Capture Library", <http://www.winpcap.org/>
- [7] RFC 2030, "Simple Network Time Protocol (SNTP) Version 4 for IPv4, IPv6 and OSP", <http://www.ietf.org/rfc/rfc2030.txt>
- [8] W. Stallings - *Data and Computer Communications*, Prentice Hall, 2004.