QoS Management in Experimental Environment

Drago Žagar, Goran Martinović and Slavko Rupčić
Faculty of Electrical Engineering
University of Osijek
Kneza Trpimira 2B, Osijek
Croatia
drago.zagar@etfos.hr, goran.martinovic@etfos.hr, slavko.rupcic@etfos.hr

Abstract -- New applications usually combine interactivity and multimedia resulting in a huge diversity of possible services. The negotiated quality of service should be result of application/user requirements and network possibilities. The most important topic in QoS provision is how to efficiently manage network resources. This paper describes an experimental implementation of the QoS network management for serving the users by some defined QoS characteristics. The applications developed for negotiation and maintenance of the desired quality of service between the server and clients are analyzed. The main classes of developed applications as well as the connections between applications´ objects are described. Finally, some results describing functionality of QoS management in experimental environment are shown.

Key-Words: - Quality of Service; Negotiation; Network resources; IP Networks; QoS Management; QoS Parameters

1 Introduction

By combining an interactivity and multimedia we could obtain a huge diversity of possible services, from extended functionality of current applications to new virtual reality applications. The recent researches in digital coding and signal compression, broadband communications and digital signal processing translate the interactive multimedia services from vision to reality, and therefore claim new demands on the network infrastructure [6][8][9].

The specification of QoS parameters` values determines the type of service. We could distinguish at least three different types of services: guaranteed, predictive and best-effort [4].

The guaranteed services enable QoS guarantees because of their specific (limited) QoS parameter values, either in deterministic or in statistic representation. Deterministic limitations could be given by some single value (e.g. mean value, negotiated value, start value, target value…), a pair of values (e.g. minimum and mean value, the lowest and target quality…) and the interval of values (e.g. a lower border is the minimum and a higher border is the maximum value). The guaranteed services could also deal with statistical QoS parameter borders, as e.g. statistical border of the error level.

The predictive (historical) services are based on the prior network behavior, and therefore the QoS parameters will be estimated by previous service values.

The best-effort service is based on a partial or none guarantees. The QoS parameter values are generally not necessary but could be defined by some deterministic or statistic border values. Most of the currently used network protocols offers best-effort services [11][13].

In order to regularly operate, the services for multimedia-networked applications use resources. Resources of special interest are resources that could be shared between the applications, the system and the network, as e.g. CPU cycles or network bandwidth. It is important to emphasize that all resources that could be shared by many processes in every layer of the Multimedia Communication System (MCS) could be a part of three main system resources: bandwidth of the communication channel, buffer space and CPU power [10][12].

QoS parameters specify the amount of resources dedicated to services, whereas queueing disciplines manage the shared resources of MCS (e.g. QoS latency parameter defines service transmission between the source and the destination by respecting the packet distribution (bandwidth assignation), queueing (buffers assignation) and scheduling (CPU cycles assignation))[2][3][7].

The defined relation between QoS and the resources is built in resource management in terms of different mappings between QoS parameters and their related resources. The description of possible realization of resource management and resource sharing shows the relation between the QoS and the resources. Here we should emphasize that the resource management and resource sharing are based on the interaction between clients and their resource managers. The client requires resource assignation by specifying QoS (this implicitly includes mapping between QoS specifications and the requested resource). The resource manager verifies the

use of resources and decides to accept or not the request for resource reservation. All active reservations states are saved, and sharing of every single resource is therefore guaranteed [4][13][14].
This paper describes an experimental implementation of the QoS network manager for serving the user with some guaranteed services on demand. Characteristics of the used media will be analyzed, and developed applications for negotiation and ensuring the desired quality of service between the server and clients will be described. The main classes of developed applications as well as the connections between applications´ objects will be described.

2 QOS NEGOTIATION

If we presume that the user has defined requests for multimedia application, these requests should be distributed to all resource manager entities involved in system. The QoS parameter distribution requires two services: negotiation about QoS parameters and QoS parameter adaptation (if different system components have different QoS specifications).
If we want to characterize the real negotiation, we should define the negotiation sides, and how they do negotiation. Every QoS negotiation has two sides. We will analyze the peer-to-peer negotiation, which could be negotiation between the application and the system, or negotiation between the user and the application.
The aim of the negotiation process is to determinate common QoS parameter values between the service user and the service provider. Furthermore, we assume the QoS negotiation where the QoS parameter values are determined by deterministic borders (minimum and mean value). We could distinguish several types of negotiation between the service user (caller, callee) and the service provider [3]:

- ***Bilateral Peer-to-Peer negotiation***

This type of negotiation is carried between two service users (peers), and the service provider is disabled to modify the QoS value proposed by the user.

- ***Bilateral Layer-to-Layer negotiation***

This type of negotiation is performed exclusively between the service user and the service provider and covers two possibilities:

1. Between the local service user and the service provider;
2. Between the main computer-sender and the network, e.g. when the sender wants to transmit the multimedia flows.

- ***Unilateral negotiation***

By this negotiation, the service provider as well as the called user is not allowed to change the QoS proposed by the caller. This type of negotiation is reduced to "take it or leave it". Furthermore, this negotiation also enables that receiver could accept the proposed QoS although it cannot accept all QoS parameters and so it can participate in communication only with lower QoS.

- ***Hybrid negotiation***

The broadcast/multicast communication implies that every receiving host could have different possibilities than the sending host (conference session). Therefore, the QoS parameter values could be negotiated between the sender host and the network, by using bilateral layer-to-layer negotiation and simple negotiation between the network and the receiving host.

- ***Triangular negotiation for information exchange***

By this type of negotiation the user caller introduces the initial request by mean values of the QoS parameters. These values could be changed by the service provider/callee across the data path by indication/reaction before setting the final value in confirm message. At the end of the negotiation process both sides have the same value of QoS parameters.

- ***Triangular negotiation for bounded target***

This type of negotiation is very similar to the former, with distinction that the QoS parameter values are set by two values: the desired QoS value (mean value) and the lowest acceptable value (minimum value). The target is negotiated about the desired QoS value, i.e. the service provider is not allowed to change the lowest acceptable QoS value (if this value cannot be assured, the connection will be rejected), but it could change the desired QoS value. The callee will decide about the final QoS value, and that value will be reported to the caller.

- ***Triangular negotiation for agreed value***

In this case QoS parameters are specified through the minimum requested value and the border value. The aim of this type of negotiation is to agree the QoS value, which is the lowest required value of the QoS parameter. The service provider could change the lowest required QoS value toward the border value. The final decision is on the callee and informs the caller by answer/acknowledgement.
Only few protocols for call establishment have built in negotiation mechanisms. ST-II (Internet Stream Protocol, Version 2) enables an end-to-end guaranteed service through the Internet. It uses triangular negotiation for bounded values for parameters concerning a throughput. The parameters concerning latency are not negotiated.
The other protocols, as e.g. RCAP (Real-time Channel Administration Protocol), RSVP and some others use triangular negotiation for different values of QoS parameters. QoS broker is an end-to-end protocol that establishes the connection using bilateral negotiation in the application layer between service users, unilateral negotiation with the operating system and triangular negotiation in the transport subsystem layer.

3 AN EXPERIMENATAL QOS NETWORK

As an experimental QoS network for serving the users by services on demand we have used the three computers connected in the Intranet network (Figure 1.). To enable the effects of different classes of services by different load in the network we have used Bandwidth Controller Version 0.15 beta. Two computers were installed as clients and the third one was installed as server. All computers were driven by the Microsoft Windows OS.

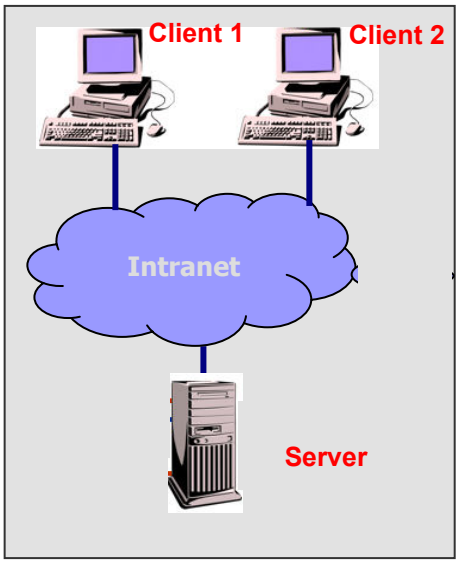

Fig. 1. Experimental QoS network

The traffic on a single computer has been monitored by Network Traffic Analyzer, by which we have monitored the bandwidth, packet size distribution and used protocols. To analyze the packets we have used Ethereal – Network Protocol Analyzer ver. 0.10 and MaaTec Network Analyzer ver. 1.4. On the server we have created a Microsoft® Access database used for user authorization, definition of predefined QoS values and some other data about users. The database includes two tables: services and users. Table services consist of type, label and service priority, while table users consist of users' data (user name, label of activity, the best allowed QoS and the maximum allowed bandwidth dedicated to the user). During the traffic monitoring all clients were connected to the server, and reproduced the streaming video data using Windows Media Player ver. 8.0. The used video stream had the following characteristics:

Video Stream: MPEG
File Size: 681,537,790 Bytes
Duration: 73:59.953
Buffer Time: 1.5 seconds
Max Bit Rate: 1596.4 Kbps
Avg Bit Rate: 1228.0 Kbps

Stream: 0
MIME type: video/vnd.rn-mp1s
Max Stream Bit Rate: 1450.8 Kbps
Avg Stream Bit Rate: 1116.0 Kbps
Stream: 1
MIME type: audio/vnd.rn-mp1s
Max Stream Bit Rate: 145.6 Kbps
Avg Stream Bit Rate: 112.0 Kbps

The Quality of Service between clients and the server has defined by QoSServer and QoSClient applications. The applications QoSServer and QoSClient have been developed for *Microsoft Windows* OS. For communications between QoSServer and QoSClient we have used the control messages (Chapter V.). For the control of parameters set on the server we have used *Traffic Control*. To set up the negotiated QoS we have used *Traffic Control Interface* (TCI). To initiate the TCI we had to call the *TcRegisterClient* function that connects our application to TCI. Afterwards we had to set the filter for the flow. In applications that use Resource ReserVation Protocol, RSVP signalization calls the basic control in local TC (*Traffic Control*) components by using TCI [5].

The key element of TC is setting the parameters for flow specification (*flowspec*) and after that TC treats all packets from one group as one single flow. TC uses the information from *flowspec* to create the flow with defined QoS parameters, and than creates filters for directing the selected packet into that flow (known as *filterspec*) (Figure 2.).

TC API is a programming interface to flow control components that regulate network traffic on the server. It enables aggregation of traffic from many sources (on the same server) in one TC flow (e.g. all traffic for destination address 1.2.3.0 could be placed in the same flow, regardless of the sending and receiving ports.

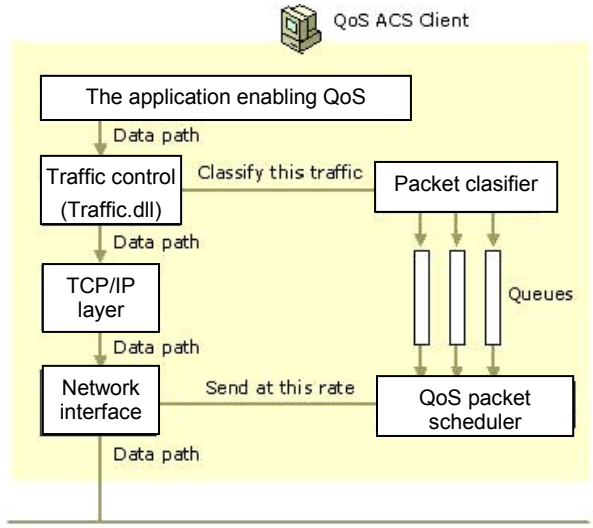

Fig. 2. Components of Traffic Control

4 *QoS*SERVER

QoS Server application negotiates with users about the desired QoS and sets the negotiated quality of service between the server and the user. This application checks the data about the user in the database and compares the requested QoS with maximum allowed QoS for the specified user. After that QoSServer negotiates with the user and sets negotiated QoS between the client and the server (Fig. 3.).[1]

4.1 Description of classes and methods

CDBConn *Class*
This class is used for connection to and communication with the database.
CDBConn() and ***CDBConn(CString sDSN, CString sUID, CString sPWD)*** – are class constructors by which the application connects to database. The second constructor for parameters takes the name of the ODBC connection for database (which is defined in QoSServer.ini file), as well as the username and the password.
CDBUserListDlg *Class*
This class is used for adding and deleting users in database and the overview of users in database.
CQoSServerDlg *Class*
The class *CQoSServerDlg* is used to connect the server and the client, to manage the messages coming from the client, to check the authorization of the client and to negotiate between clients and the server.
Ctcmon *Class*
This class is used to set the negotiated QoS between the server and the client.
mysocket *Class*
This class uses Windows socket API for connecting the server and clients.

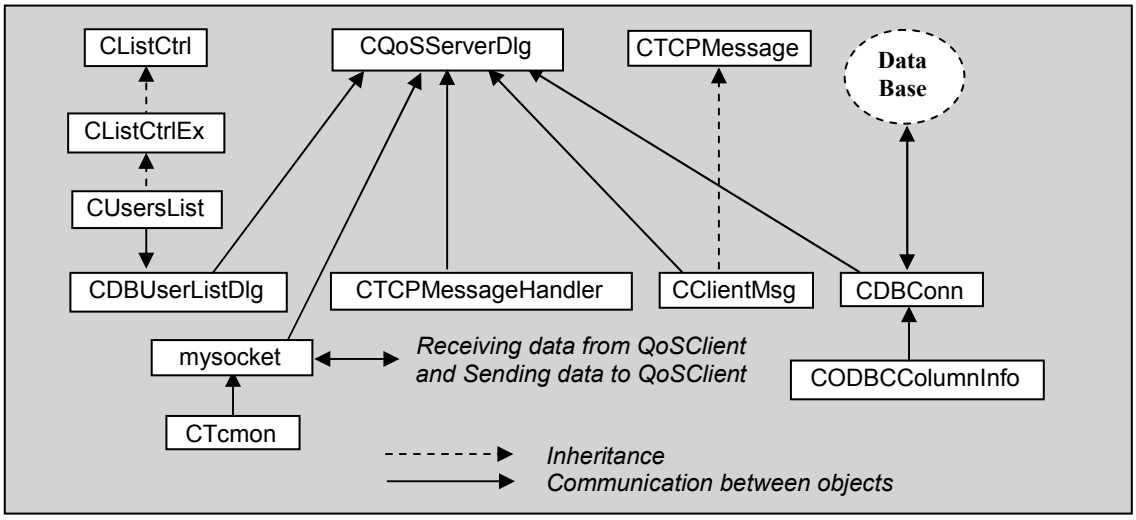

Fig. 3. Relations between objects in QoSServer

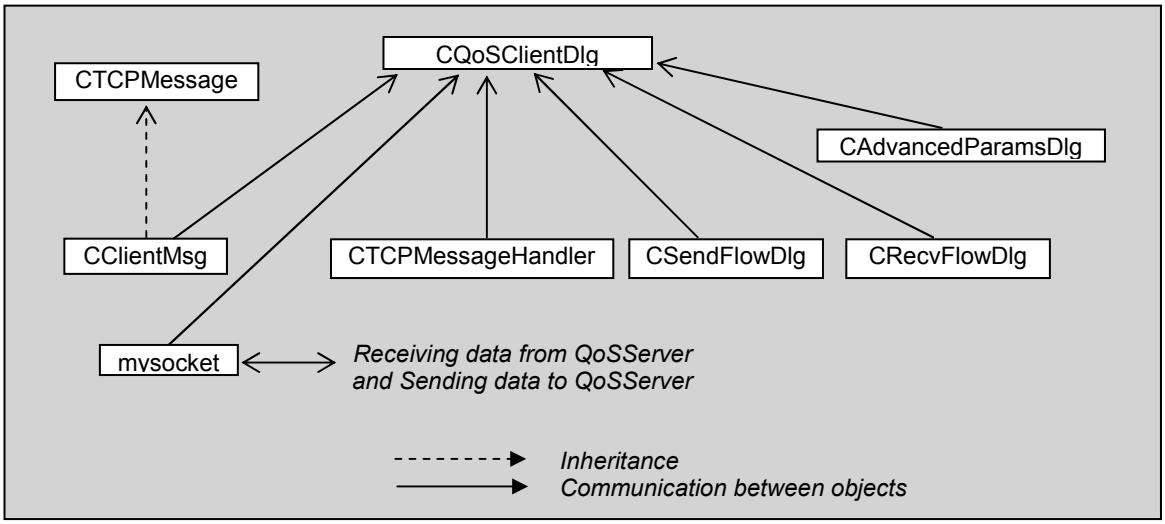

Fig. 4. Relations between objects in QoSClient

5 *QoS*CLIENT

The QoSClient application enables the user to define the required QoS and the type of negotiation with the server. This application supports the following negotiations: unilateral, bilateral layer-to-layer, triangular for information exchange and triangular for bounded values. The required QoS could be defined for incoming and outcoming flow.

Because of that QoSClient application sends a request for QoS to the server that after the negotiation sets the negotiated flow, the incoming flow (*Receiving Flowspec*) in QoSClient application actually represents the outcoming flow on the server, and the outcoming flow (*Sending Flowspec*) in QoSClient application represents the incoming flow on the server. Besides the basic QoS parameters there could also be defined the additional parameters for every single flow (Figure 4., Figure 5.).

The basic QoS parameters that could be defined are:

Token Rate specifies the data rate by which the data could be transmitted per flow. If the server cannot support the required rate, the application will wait or discard the flow. It is therefore very important that the application rationally assesses its traffic needs (e.g. in video applications *Token Rate* is usually set at the mean data rate). If the *Token Rate* is set at -1, there will be no effect on the data rate.

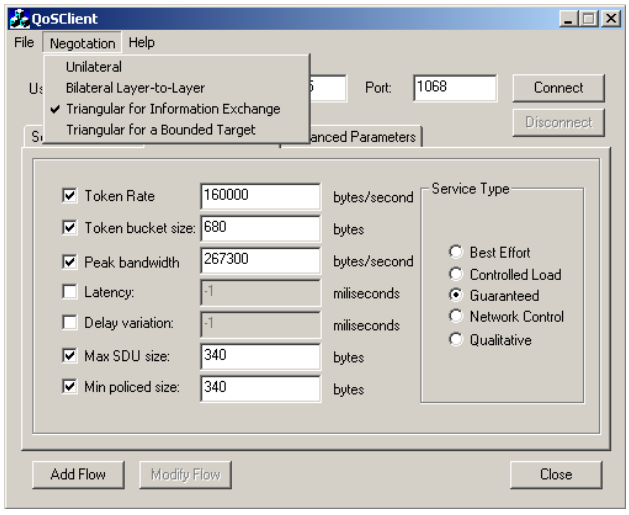

Fig. 5. The basic QoSClient interface

Token Bucket Size specifies the maximum credit size the flow could use. In video applications the *Token Bucket Size* will probably be the maximum mean frame size. In applications with a constant data rate *Token Bucket Size* should enable only small changes.

Peak Bandwidth specifies the upper border of allowed temporarily based transmission per flow.

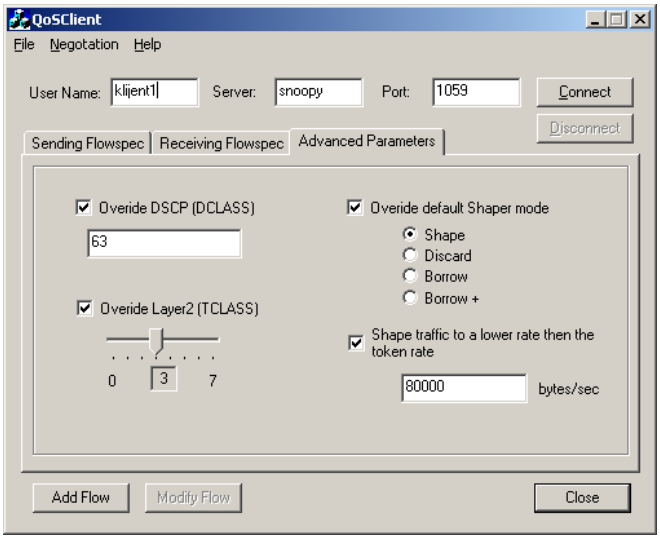

Fig. 6. Specification of additional parameters

Latency specifies the maximum acceptable delay between the sender and receivers.

Delay Variation specifies the difference between maximum and minimum packet delay. The applications could use this parameter to determine the receiving buffer size.

Service Type precises the service level negotiated for the flow. It could be one of the following service types:

Figure 6. *Specification of additional parameters*

BESTEFFORT – The defined parameters are used as QoS constraints. The flow control tries to obtain the requested QoS level but does not guarantee anything.

CONTROLLEDLOAD – It ensures the end-to-end QoS that tightly approximates the quality given by the best effort service in low load in the network. The applications that use the CONTROLLEDLOAD service could presume that the network will deliver a very high percentage of sent packets, and that latency by a very high percentage of packets will not exceed the minimum latency of any successfully delivered packet.

GUARANTEED – Guarantees that the packet will be received within some guaranteed delivery time and that it will not be discarded because of queuing. This service is dedicated to applications that need hard guarantees.

NETWORK_CONTROL – This type of service is used only for control packets' transmission (as RSVP signaling messages). This type of service is of highest priority.

QUALITATIVE – This service is used when the application requires a service that is better than BESTEFFORT, but cannot quantify its requirements.

Max Sdu Size Defines the maximum allowed or used packet size in the traffic of a specific flow.

Minimum Policed Size Defines the minimum packet size for which the minimum required QoS will be enabled.
The additional parameter (Figure 6.) that could be defined for the required QoS are:
Overide DSCP (DCLASS) The object of traffic control QOS_DS_CLASS enables the standard values of DSCP (DiffServ Code Point) for IP packets associated to a specific flow. A DSCP value is assigned according to the service type. The range of allowed values is from 0x00 to 0x3F.
Overide default shaper mode QoS object, QOS_SHAPING_RATE, defines the shaping characteristics that will be used to the specific flow.
SHAPE - TokenRate must be determined. The packets that are not acknowledged will be saved in the shaper till acknowledgement.
DISCARD - TokenRate must be determinate. The packets that are not acknowledged will be discarded.
BORROW – the flow receives the resources remained after all packets of higher priority have been served. If the TokenRate is defined, the packets could be unacknowledged and their priority will be degraded below best-effort.
BORROW_PLUS – is very similar to BORROW mode, but no packets will be labeled as unacknowledged.
Overide Layer2 (TCLASS) The traffic control object, QOS_TRAFFIC_CLASS, is used for changing the standard priority values assigned to packets in the specific flow. The priority is normally assigned according to the service type.
To enable the packets to determine their priority in layer 2 headers (as e.g. 802.1p header) it is necessary to enable the change of priority. The priority values could be between 0 and 7.
Shape Traffic to a lower rate than the token rate QOS object, QOS_SD_MODE, defines characteristics of the traffic control–packet shaper. By using this object the TokenRate could be limited to a lower value than defined in basic QoS parameters.

5.1 Description of classes and methods

CadvancedParmsDlg Class
CAdvancedParmsDlg class is used for definition of additional QoS parameters.
CqoSClientDlg Class
CQoSClientDlg uses for connection and communication between the server and the client, management of messages coming from the server, collection and sending of the required QoS parameters to the server, definition of the negotiation mode and negotiation with the server about the required QoS.
CrecvFlowDlg Class
CRecvFlowDlg is used for QoS parameters definition for the incoming flow.
CsendFlowDlg Class
CSendFlowDlg is used for QoS parameters definition for the outcoming flow.
CtriangNegotDlg Class
CTriangNegotDlg is used for definition of the lowest acceptable value of the QoS parameter for triangular negotiation for the bounded target.
Mysocket Class
Mysocket uses Windows Sockets API for connecting the server to the clients.

5.2 Messages for communication between QoSServer and QoSClient

For creation and operation of messages (Figure 7., Table 1.) used for QoSServer and QoSClient communication we use the following classes: *CClientMessages*, *CTCPMessage*, *CTCPMessageHandle*.
Message flag – specifies the start of the message
TXID – if the message has many fragments, *TXID* determines to which message the specific fragment belongs. All fragments of a single message have the same*TXID*.

Message flag	**0xbf**
Message flag	0xbf
Message flag	0xbf
Message flag	0xbf
TXID lower byte	BYTE value
TXID higher byte	BYTE value
Part indication	0x01 = Client part
Reserved	0x00 (for future use)
Message type	Message type code
Optional parameter code	Optional parameter code
Optional parameter length	Optional parameter length
Optional parameter data	Described at 3
End of optional parameters	0x00

Fig. 7. General message format

Part indication – denotes to which application a respective message belongs.
Reserved – this parameter is currently not in use.
Message type – denotes the message.
Optional parameter code – denotes message parameters.
Optional parameter length – the length of message parameters.
Optional parameter data – the parameters' message data.
End of optional parameters – the end of the message.

MESSAGE	MESSAGE	PARAMETERS
LOGIN	0x01	USERNAME
LOGIN_RESP	0x04	RESULT
QOSREQ	0x02	COMPUTERIP
		SERVICETYPE_SEND
		TOKENRATE_SEND
		MINTOKENRATE_SEND
		TOKENBUCKETSIZE_SEND
		PEAKBANDWIDTH_SEND
		LATENCY_SEND
		DELAYVARIATION_SEND
		MINPOLICEDSIZE_SEND
		MAXSDUSIZE_SEND
		SERVICETYPE_RECV
		TOKENRATE_RECV
		MINTOKENRATE_RECV
		TOKENBUCKETSIZE_RECV
		PEAKBANDWIDTH_RECV
		LATENCY_RECV
		DELAYVARIATION_RECV
		MINPOLICEDSIZE_RECV
		MAXSDUSIZE_RECV
		DCLASS
		MODE
		SHAPINGRATE
		TCLASS
		MODFLOW
		NEGOTATION
		MINSERVTYPE_RECV
		MINSERVTYPE_SEND
QOSREQ_RESP	0x03	COMPUTERIP
		RESULT
		REMAINBANDW
		MINSERVTYPE

Table 1. Codes of messages

5.3 Parameters' codes (Table 2.)

CTCPMessage Class
CTCPMessage is used for message creation.
CTCPMessageHandler Class
CTCPMessageHandler is used for saving messages coming to the buffer and for catching one by one message from the buffer.
CClientMessages Class
CClientMessages is used for composing messages for client/server communication and for extraction of parameters from the message.

6 EXPERIMENTAL RESULTS

6.1 The server

During traffic monitoring two client applications have been connected on the server. Both clients played the video stream from the server. One of the applications had set Guaranteed class of service (Client 1), and the second had set Best Effort class of service (Client 2).

PARAMETER TYPE	GLOBAL PARAM. TYPE	PARAM. TYPE CODE	LENGTH
EOP	BYTE	0x00	1
USERNAME	ASCII	0x01	-
SERVICETYPE_SEND	DWORD	0x02	4
SERVICETYPE_RECV	DWORD	0x03	4
TOKENRATE_SEND	DWORD	0x04	4
TOKENRATE_RECV	DWORD	0x05	4
TOKENBUCKETSIZE_SEND	DWORD	0x06	4
TOKENBUCKETSIZE_RECV	DWORD	0x07	4
PEAKBANDWIDTH_SEND	DWORD	0x08	4
PEAKBANDWIDTH_RECV	DWORD	0x09	4
MINPOLICEDSIZE_SEND	DWORD	0x0a	4
MINPOLICEDSIZE_RECV	DWORD	0x0b	4
MAXSDUSIZE_SEND	DWORD	0x0c	4
MAXSDUSIZE_RECV	DWORD	0x0d	4
RESULT	BYTE	0x0e	1
REMAINBANDW	DWORD	0x0f	4
COMPUTERIP	ASCII	0x10	-
LATENCY_SEND	DWORD	0x11	4
LATENCY_RECV	DWORD	0x12	4
DELAYVARIATION_SEND	DWORD	0x13	4
DELAYVARIATION_RECV	DWORD	0x14	4
DCLASS	DWORD	0x15	4
MODE	DWORD	0x16	4
SHAPINGRATE	DWORD	0x17	4
TCLASS	DWORD	0x18	4
MODFLOW	BYTE	0x19	1
NEGOTATION	BYTE	0x1a	1
MINTOKENRATE_RECV	DWORD	0x1b	4
MINTOKENRATE_SEND	DWORD	0x1c	4
MINSERVTYPE_RECV	DWORD	0x1d	4
MINSERVTYPE_SEND	DWORD	0x1e	4
MINSERVTYPE	DWORD	0x1f	4

Table 2. The Types of parameters

If the quality of service is not defined, all users use the necessary bandwidth (if available) for the best possible data transmission. The server behaves as if all connected flows had defined Best-effort service. Figure 8. and Figure 9. show that the engaged resources (bandwidth) on the server are larger than when the QoS parameters for the clients are set. This difference is a consequence of Client 1 having bandwidth set on 80.000 Bytes/sec, using Guaranteed service, while Client 2 has bandwidth set on 160.000 Bytes/sec, using Best-effort service. Figure 9 shows that the engaged bandwidth is about 240.000 Bytes/sec, and in some moments it suddenly raises or falls.

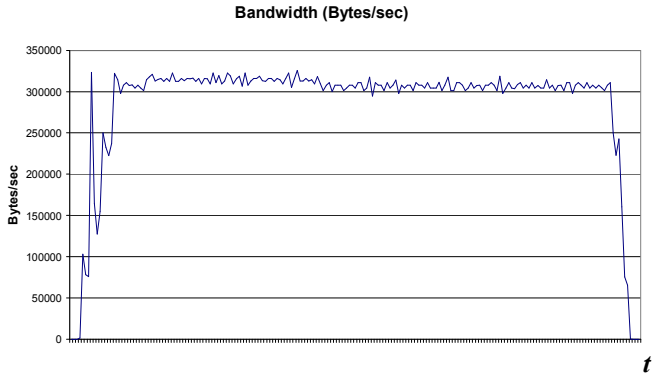

Fig. 8. Bandwidth on the server –QoS not set up

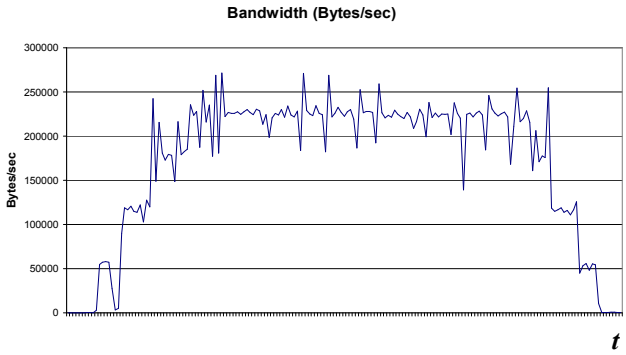

Fig. 9. Bandwidth on the server –QoS set up

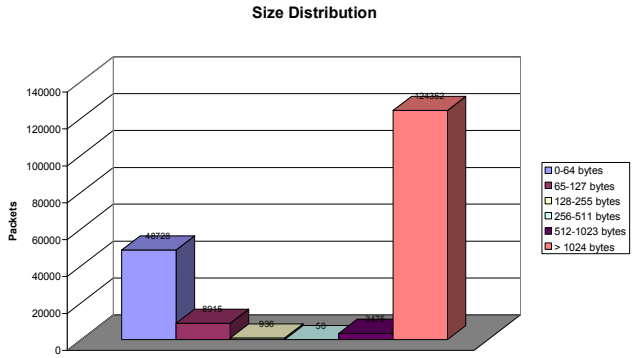

Fig. 10. Packets' size distribution – QoS not set up

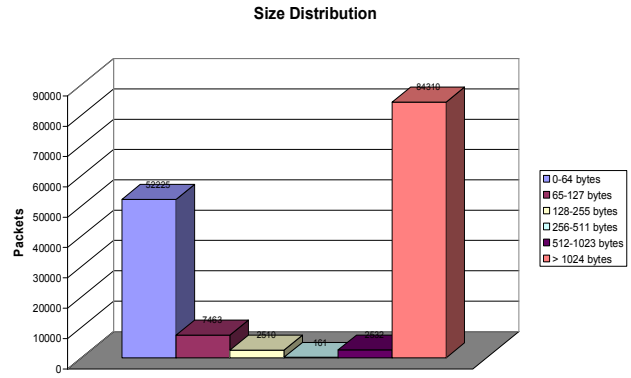

Fig. 11. Packets' size distribution – QoS set up

These changes are a consequence of different types of services defined on clients. E.g. the client with the Best-effort service in some moments uses a larger amount of bandwidth if necessary and if available on the server.
By comparing Figures 10. and 11. we can conclude that the number of larger packets (> 1024 bytes) is significantly decreased, and the number of smaller packets is increased when the QoS is set.

6.2 QoS parameters for Client 1

Fig.9. shows QoS parameters defined for Client 1. If the QoS is not defined Client 1 partially uses the bandwidth greater than 350.000 Bytes/sec. Figure 9. shows that by the defined quality of service the used bandwidth never exceeds 80.000 Bytes/sec, which is the defined bandwidth value for Client 1 (Guaranteed class).

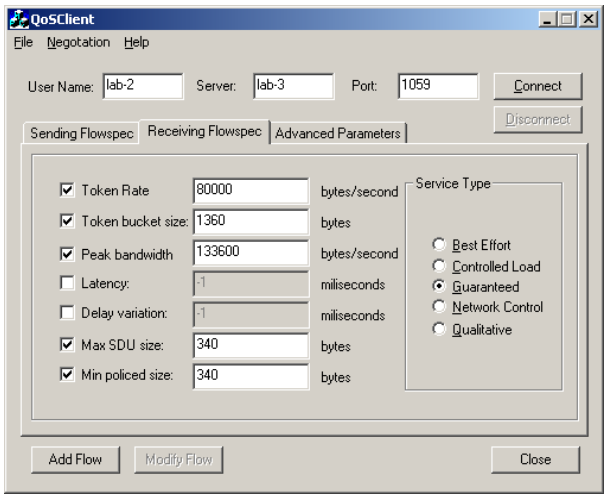

Fig 12. Flow parameters for Client 1

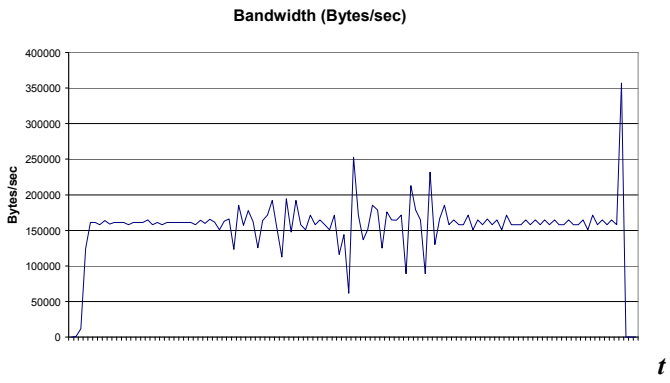

Fig.13. Bandwidth on Client 1 – QoS not set up

6.3 QoS parameters for Client 2

For Client 2 the defined type of service was Best-effort, which uses defined parameters as directions for quality of service but does not guarantee anything. When the QoS is not defined, the data are transferred through the

network as best-effort service, but by low level network load, as in our experimental network, the difference in the used bandwidth is negligible.

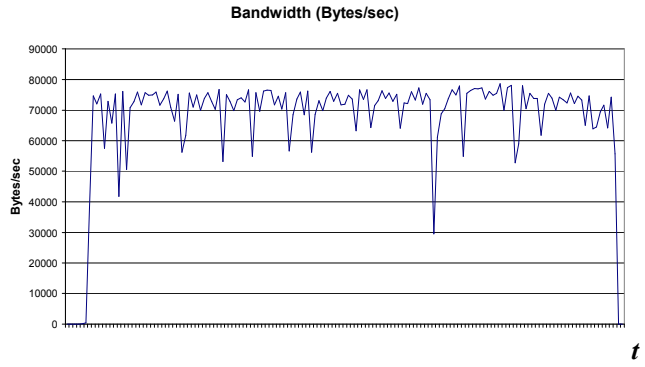

Fig.14. Bandwidth on Client 1 – QoS set up

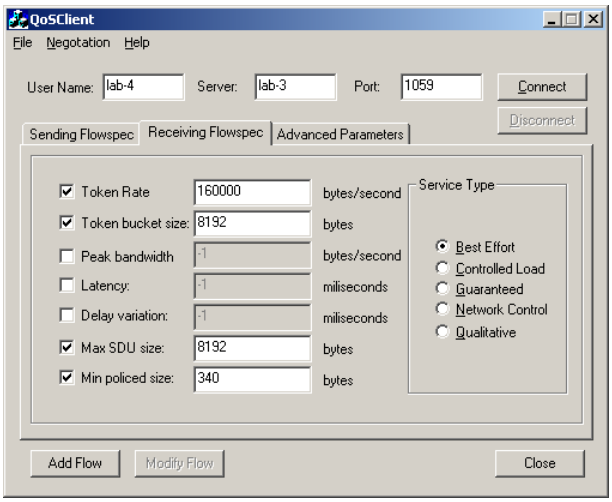

Fig.15. Flow parameters for Client 2

7 Conclusion

This paper deals with quality of service realization and experimental implementation in the communication network. Quality of service is of great interest when transmitting sensitive real-time data, as e.g. video and audio. If we could define and set the QoS parameters to achieve a prerogative treatment of our data, then we could achieve the adequate quality for our applications. To achieve end-to-end quality of service it is very important that all network components support the QoS protocols.

Interactive multimedia services will be more and more available, but outside necessary hardware requests, these new systems will succeed only if a user interface will be user-friendly (even the users without technical knowledge), if the quality of service will be of satisfactory level, if the contents will be versatile and if the service price will be acceptable for majority. It is obvious that the users will be satisfied with the guaranteed quality of service, but this class of service allocates maximum system resources and therefore it is most expensive. If the users will not pay for the quality of service, they will always ask for the best possible QoS. That leads to service diminution and to well-known best-effort service. By introducing the appropriate QoS accounting the QoS negotiation will become the real method for QoS achievement.

This paper introduces an experimental implementation of the QoS system with some guaranteed services on demand. Characteristics of the used audio and video were analyzed, and applications for negotiation and ensuring the desired quality of service between the server and clients were developed. The main classes of developed applications as well as the connections between application objects were described. Experimental measurements show that for some specific data flow (video data) with the defined Guaranteed type of service, parameters affecting the QoS will be maintained within defined borders. The identical flow having defined Best-effort service will receive an acceptable service in the lightly loaded network and an unacceptable service in the heavy loaded network.

References:

[1] Y. Bernet. „*Networking Quality of Service and Windows® Operating Systems*“,Sams, November 2001

[2] W. Fei, W. Wen-Dong, L. Yu-Hong, C. Shan-Zhi „A New Architecture Integrating Web QoS and Network QoS*”, WSEAS Transactions On Computers Issue 5, Volume 5, May 2006 ISSN 1109-2750*

[3] B. Dekeris, L. Narbutaite. „IP QoS evaluation using interoperability of differentiated and integrated services“, *WSEAS Transactions On Communications, Issue 2, Volume 4, February 2005, ISSN 1109-2742*

[4] Steinmetz R., K. Nahrstedt. „*Multimedia: Computing, Communications, and Applications*“, Prentice Hall, July 1995.

[5] Y. Bernet, J. Stewart, R. Yavatkar, D. Andersen, C. Tai, B. Quinn, K. Lee. „*Winsock Generic QOS Mapping*“ ,1998

[6] T. Zahariadis, C. Rosa, M. Pellegrinato, A. Bjork Lund, G. Stassinopoulos. „Interactive Multimedia Services to Residential Users“, *Proceedings of Second IEEE Symposium on Computers and Communications,* June 1997

[7] M. Hsieh, E. Hsiao-Kuang Wu. „Postgate: QoS-aware Bandwidth Management for Last-mile ADSL Broadband Services”, *WSEAS Transactions On Communications,* Issue 5, Volume 5, May 2006 ISSN 1109-2742

[8] M. Albrecht, M. Kőster, P. Martini, M. Frank. „End-to-end QoS Management for Delay-sensitive Scalable Multimedia Streams over DiffServ“, *Proceedings of the 25th Annual Conference on Local Computer Networks* (LCN'00), Tampa, FL, USA, November 2000

[9] A. Vogel, B. Kerhevé, G. v. Bochmann, J. Gecsei. „*Distributed Multimedia and QOS: A Survey*“ 1995

[10] C. Ruey-Shun, T. Yung-Shun, K. C. Yeh, H. Y. Chen. "Using Policy-based MPLS Management Architecture to Improve QoS on IP Network", *WSEAS Transactions On Computers*, Issue 5, Volume 7, 2008 ISSN: 1109-2750

[11] W.F.Poon, K.T. Lo, J. Feng. „*First segment partition for video-on-demand broadcasting protocols*“, 2002

[12] J. Levendovszky, C. Orosz. "Generalized Statistical Bandwidth for Optimal Resource Management", *WSEAS Transactions On Communications*, Issue 10, Volume 5, October 2006, ISSN 1109-2742

[13]A.Iyer. „*A Study of IP QoS*“, http://qos.ittc.ukans.edu/study/Mainpage.html

[14] G.Armitage. „*Quality of Service in IP Networks: Foundations for a Multi-Service Internet*“, Sams 2000.