

# QoS Parameters' Benchmarking System with Complex Traffic Patterns Definition

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

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

*Abstract* – In this paper a QoS (Quality of Service) parameters' benchmarking system with complex traffic patterns definition is presented (the design and implementation). Benchmarking is used to compare the network's behavior when tested with different traffic patterns. The application generates complex traffic patterns and stores them in a database for future measurements. Also, it measures and computes the QoS parameters – delay, delay variation, bandwidth, and stores and displays the results both in numerical and graphical form. The main advantages of this system are the traffic patterns complexity, the accuracy of the measurements and the user interface. The application represents a part of a complex framework designed for developing QoS technologies such as traffic identification and marking, queue and congestion management, traffic shaping and policy, specific protocols and is developed in C# language for .NET platform and uses a SQL Server connection.

*Keywords* – QoS parameter, Complex traffic pattern, Measurements, Database

## I. INTRODUCTION

Benchmarking can be defined as a standard test to measure the performance of computer hardware or software [Encarta definition]. The purpose of a benchmark is to evaluate the performance and functionality of different systems and the complexity of a benchmark depend on the features that are measured.

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 QoS parameters are measured using different commands or benchmarking systems.

In this paper we describe the design and implementation of a QoS parameters' benchmarking system with complex traffic pattern definition. This application uses a management station that defines the traffic patterns in the network, stores them, and allow for the user to load and send these patterns to the agent stations. These stations generate the traffic, measure the information necessary for QoS parameters computation, and send the information to the master station. The master station computes the

QoS parameters and stores and displays these results both in numerical and graphical form. The application developed in C# language for .NET framework presents a user friendly interface that allows for complex traffic patterns definition, storage capabilities using a SQL Server connection and database, and detailed results display.

The rest of this paper is organized as follows. Section II provides background information related to QoS parameters. Section III describes the software application for QoS parameters benchmarking system. Section IV presents the experimental results and section V concludes the paper.

## II. BACKGROUND

Several definitions for Quality of Services (QoS) have been identified throughout the years in different standards and bibliographical references, without any unique and exhaustive formal definition. The most illustrative definitions of the concept are [1]:

- [ISO 8402/1986] - Quality is the totality of features and characteristics of a product or service that bears on its ability to satisfy stated or implied needs. The product, respectively the service, represents the result of the activities or processes within the system.
- [ITU-T, 1994] - defines the collective effect of services' performance, which determines the level of satisfaction of the service user.
- [ISO/IEC X641] - it defines the qualities that refer to the way an object or a group of objects (components) collectively works, or reflect the qualitative performance of the service offered through the network

At network level, 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, which represent the main QoS parameters. A service represents the traffic performance provided to the client and is characterized by the QoS parameters. QoS techniques represent the implementation specific methods,

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.

Several tools are available for network performance measurement and can be classified in two main types: monitoring and benchmarking tools [2]. Monitoring tools capture packets and display their headers and data. Benchmarking tools generate test traffic in the network and measure parameters like throughput, delay or jitter. These tools are used to evaluate the results of the QoS implementations in order to verify if the QoS goals have been reached. Examples of these tools are ping, traceroute, tcp, Netperf, NetPIPE, Iperf, Patchar, Bprobe, Cprobe, Cap or DBS.

Several techniques have been developed for delay measurement. One of the most used methods is sending packets from source to destination and back. In this case the measured parameter is round trip time (RTT) representing the total time necessary for the packet to travel from source to destination and backward [3]. This method is not accurate for one-way delay measurements since the delay from source to destination might be different than the delay from destination to source. Another method that computes one-way delay parameter is based on timestamps introduced by the transmitters [3] [4]. Both the transmitter and the receiver must have synchronized clocks, so the receiver reads the receiving time and based on the transmission time in the timestamp computes the delay. Another method was proposed by IP Performance Metrics (IPPM) group that defined A One-way Active Measurement Protocol (OWAMP) which measures unidirectional characteristics such as one-way delay [5]. Another protocol, IP Measurement Protocol (IPMP), measures delay and associated path metrics based on echo request end echo reply packet exchange [6].

Several metrics and measurement techniques are used for bandwidth estimation [7]. Bandwidth represents the transfer data rate in a network. The capacity specifies the maximum bandwidth of a path, the available bandwidth identifies the maximum unused bandwidth of a path and Bulk-Transfer-Capacity represents the achievable throughput of a bulk-transfer TCP connection. Variable Packet Size (VPS) probing and Packet/Pair train Dispersion (PPTD) probing can be used for capacity measurement. Self-Loading Periodic Streams (SLoPS) and Trains of Packet Pairs (TOPP) are used for available bandwidth measurement.

Measuring and verifying network performance is essential for improving the network design process, monitoring progress against performance indicators, or several other measurements and service level agreements, ensuring that the user's expectations are always exceeded.

### III. BENCHMARKING SYSTEM APPLICATION

The benchmarking system we propose is composed of a QoS Benchmarking Manager (QBM), QoS Benchmarking Agents (QBAs) and a QoS Benchmarking Protocol (QBP). QBM synchronizes QBAs with itself, defines and transmits to QBAs the traffic patterns and receives, computes, displays and stores the summary results. QBAs synchronize themselves with QBM, receive the traffic patterns, generate the traffic and transmit the summary results to QBM. We propose a protocol, QBP, for communication between the benchmarking entities. QBP is an application layer protocol that uses both TCP and UDP transport layer protocols. The communication between the QBM and QBAs is control type and uses TCP protocol for reliability. The communication between QBAs is data type and uses either TCP or UDP protocols, function of the traffic definition. As we said there are two main types of messages: control and data. Control messages are synchronization, traffic pattern, synchronization summary and results summary. The structure of the messages is presented below.

Synchronization:

<i>Id</i>	<i>AIP</i>
-----------	------------

Traffic pattern:

<i>Id</i>	<i>SIP</i>	<i>DIP</i>	<i>Prot</i>	<i>Port</i>	<i>Nr</i>	<i>Len</i>	<i>Gap</i>	<i>StartT</i>	<i>RezT</i>
-----------	------------	------------	-------------	-------------	-----------	------------	------------	---------------	-------------

Data:

<i>Id</i>	<i>SendT</i>	<i>No</i>	<i>Len</i>
-----------	--------------	-----------	------------

Synchronization summary:

<i>Id</i>	<i>AIP</i>
-----------	------------

Results summary:

<i>Id</i>	<i>SIP</i>	<i>DIP</i>	<i>Prot</i>	<i>Port</i>	<i>Nr</i>	<i>Len</i>	<i>SendTs</i>	<i>RecvTs</i>
-----------	------------	------------	-------------	-------------	-----------	------------	---------------	---------------

Where Id represents the message type and has the following values: for synchronization is 0, for traffic pattern is 1 if the message destination is client and 2 if the message destination is server, for data is 3, for synchronization summary is 4 and for results summary is 5. We reserved for this field 1 byte so several other message types can be defined in the future developments. AIP specifies the agent's IP address synchronized with QBM. SIP and DIP represent the source and destination IP addresses respectively. The QBA specified by DIP field will open a server socket and will receive the data traffic from the client socket of the QBA specified by SIP field. Prot and Port fields identify the transport protocol and port number used in the data traffic. Nr and Len fields specify the data messages number and length while Gap field identifies the time interval between consecutive messages. StartT field represents the time when the sending agent begins the transmission of the data traffic. This field was introduced because the data traffic should begin after all the agents are synchronized and the entire traffic pattern is set. RezT field specifies the time when the receiving agent sends the summary results to QBM. This field was introduced because summary results transmission should

be performed after the entire data traffic between all the agents is ended. SendT specifies the time value when the data message is sent. Even if this time can be computed from StartT and Gap fields SendT field is more accurate because the agent can be busy at the predefined moment and the transmission might occur later. No field specifies the data message number. This field is used to detect lost packets and packet reordering. SendTs and RecvTs fields represent the time values when the data messages were sent by the transmitter QBA and received by the receiver QBA respectively.

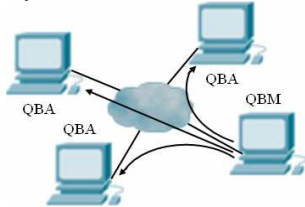


Figure 1. Synchronization and configuration step

The network performance analysis is done in three steps. In the first step, QBM synchronizes the agents with its own clock using SNTP protocol. For this, synchronization messages are transmitted to QBAs. The agents synchronize themselves and acknowledge the synchronization with synchronize summary messages to QBM. Also in this step the traffic patterns are defined and transmitted from QBM to QBAs by the means of traffic pattern messages. Another option provided by the system is to save the user's defined traffic patterns. These traffic patterns can be loaded for future analysis, so the measurements can be repeated fast and faultless in order to verify different network QoS technologies on the same traffic characteristics. This step is illustrated in Fig. 1.



Figure 2. Data traffic transmission and time values storage

In the second step (presented in Fig. 2.) data traffic pattern is transmitted and received by the agents using data messages. These traffic patterns can be generated using both TCP and UDP protocols function of the user's defined settings. At the receiving of each data message, the agent reads and stores the time so that this information can be sent to QBM in the third step.

Finally, in the third step, information collected by the agents is transmitted through results summary messages to the manager, which computes the QoS parameters,

displays and stores the results for later analysis. Fig. 3 illustrates the last step.

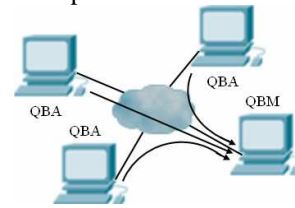


Figure 3. Results summary transmission.

The benchmarking system for QoS parameters uses a client/server architecture composed of two main applications: the QBM and the QBA. The communication between them is accomplished through a series of messages that respect the previously described QB protocol. In addition, they are used to determine the main QoS parameters that this application is monitoring.

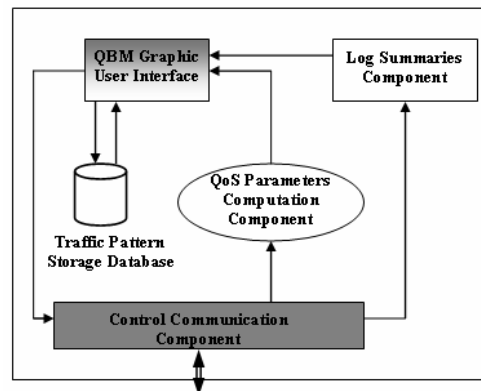


Figure 4. Benchmarking system architecture: QBM

The *QBM application*, illustrated in Fig. 4, has four primary components: Graphic user interface, control communication component, computed QoS component and the traffic pattern storage database. The GUI's responsibility (Fig. 5) is to interact with the user, through a friendly, helpful and intuitive interface, designed using .NET facilities. The user has the possibility to define and store complex traffic patterns, by specifying the source IP, the destination IP, the desired protocol (TCP/UDP) and communication port, the number of packets to be sent with a custom length, the gap between the packets, and also the starting time and the duration. Only when the duration timer expires will the agents be sending the results of their measurements to the manager. This ensures that there will be no extra traffic on the network to influence the measurements, during the benchmarking operation. Another characteristic of the QBM is the possibility to save the defined traffic patterns and to load them for several other measurements.

Furthermore, it provides several tools for traffic and data analysis to the user, by displaying the results both in numerical and graphical form.

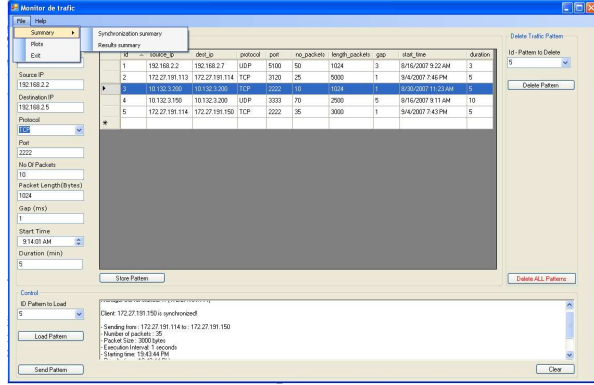


Figure 5. Benchmarking system: QBM GUI

The control communication component consists, first, of a TCP server that listens for and accepts incoming connections in an asynchronous mode, and, second, of a TCP client used to send the synchronization and traffic pattern information to the QBA applications on a predefined control port. Only synchronized QBA can participate in further actions. Another important task is to receive summaries that contain the results, to interpret and pass them to the computing component.

The third component – the computed QoS one, computes the delays, IPDVs (IP Packet Delay Variations) or jitters and bandwidths, based on the timestamps contained in the receiving packets. The formulas used for these computations are:

$$\text{Delay: } delay_i = RecvT_i - SendT_i, \quad i = \overline{1, n} \quad (1)$$

where  $RecvT_i$  - receiving times,  $SendT_i$  - sending times,  $n$  - number of received packets

$$\text{IPDV [8]: } IPDV_{i,i+1} = delay_i - delay_{i+1} \quad (2)$$

$$\text{Bandwidth: } bandwidth_i = \frac{Len}{delay_i}, \quad i = \overline{1, n} \quad (3)$$

where  $Len$  – data length,  $n$  - number of received packets

The traffic pattern storage database component is used to save the defined traffic patterns and to load them for several other measurements, thus allowing us to repeat the measurements in the same network traffic conditions. For this purpose a SQL Server 2000 connection and database was used. This represents one of the main capabilities of the system needed for the complex framework that we are designing (framework for developing QoS technologies such as traffic identification and marking, queue and congestion management, traffic shaping and policy, specific protocols).

To store the results of these computations, the log summaries component is used.

Each *QBA application* - illustrated in Fig. 6, consists of a control communication component and several pairs of

traffic servers and clients. The control communication component has two main functions: (1) to receive synchronization and traffic pattern information from the QBM computer, and (2) to send summary packets. The synchronization method is based on a SNTP (Simple Network Time Protocol) [9] client application, which requires Windows Time Service to be running on the computer. The information is interpreted from the incoming packets and the IP address of the QBM is saved internally and will be used as the time server address. The agent initializes the process, creating a new object that uses the SNTP protocol to perform the synchronization operation.

In case of receiving a control message as a source, the agent will initialize a UDP or TCP client for the transmission of a certain traffic pattern required by the QBM.

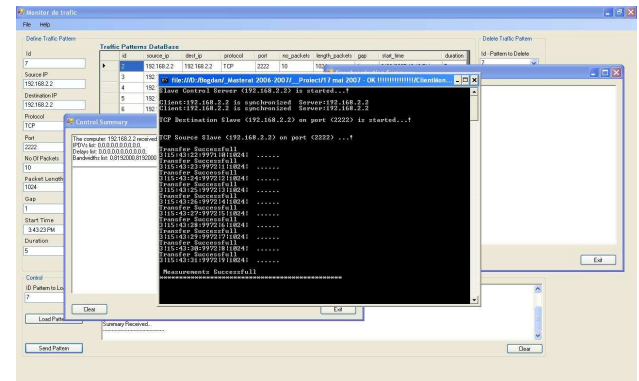


Figure 6. Benchmarking system: QBA

If the agent is considered a destination machine, it must prepare for the incoming traffic (create a UDP or TCP multithreaded server, so that several sender and receiver processes can run concurrently on every measurement PC) and the generation of the desired parameters. Regardless of the transmission protocol, the agent sends the summary to the manager server using a TCP utility, in order to ensure the safe arrival of the required data on the predefined control port.

#### IV. EXPERIMENTAL RESULTS

The benchmarking system was tested on several networks, configured with Cisco switches and the results were more than satisfactory.

In Fig. 7 and Fig. 8 are presented the results for a traffic having the following pattern: 172.27.191.114 source IP, 172.27.191.150 destination IP, TCP protocol, 2222 port, 35 packets, 3000 bytes length, 1000 ms gap. Delay and IPDV parameters are plotted in Fig. 7 and are measured in ms. It can be observed that while delay has only positive values, IPDV can have both positive and negative values. IPDV, delay and bandwidth parameters are

presented in Fig. 8 in numerical form. Bandwidth measured in bps.

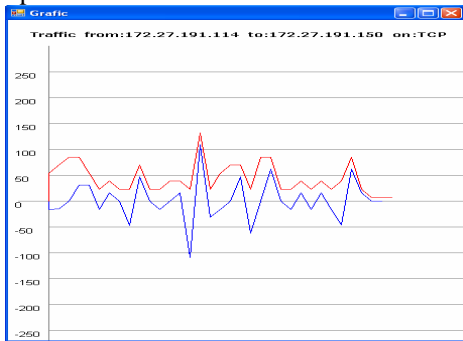


Figure 7. Delay and IPDV in graphical form

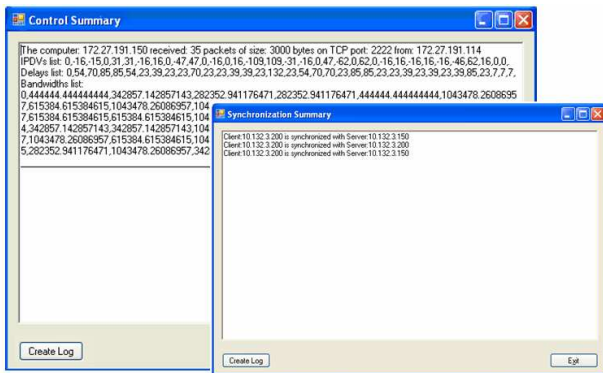


Figure 8. QoS parameters in numerical form and synchronization summary

## V. CONCLUSIONS

In this paper we described a benchmarking system for QoS parameters that represents a part of a complex framework designed for developing QoS technologies. The system measures both TCP and UDP parameters. Delay, IP delay variation (IPDV) or jitter and bandwidth are the parameters measured by the application. The system's architecture is composed of three parts: QoS Benchmarking Manager (QBM), QoS Benchmarking Agents (QBAs) and QoS Benchmarking Protocol (QBP). The QBM controls the QBAs synchronizing them with itself and also, it is used to define and store, in a SQL database, the traffic patterns. This ensures the possibility to run the same traffic patterns in order to repeat the measurements in the same network traffic conditions. Also it receives, computes, displays and stores the summary results. QBAs receive the traffic patterns, generate the traffic and transmit the summary results to QBM. We propose a protocol, QBP, for communication between the benchmarking entities. QBP is an application layer protocol that uses both TCP and UDP transport layer protocols. The system was built using a component based architecture and implemented in an evolutive incremental approach, making the application modular, reusable and

easy to understand. The application developed in C# language for .NET framework and SQL Sever 2000, presents a user friendly interface that allows for complex traffic patterns definition and detailed results display. In the future we intend to develop and incorporate new metric measurement techniques and also, to integrate the benchmarking system for QoS parameters into the framework designed for developing QoS technologies such as traffic identification and marking, queue and congestion management, traffic shaping and policy, specific protocols.

## 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] C. Aurrecoechea, A.T. Campbell, L. Hauw, "A Survey of QoS Architectures", ACM/Springer Verlag Multimedia Systems Journal, Special Issue on QoS Architecture, Vol. 6 No. 3, pg. 138-151, May 1998. [a former version can be found in: *A Review of QoS Architectures*, Proceedings of the 4. International IFIP Workshop on Quality of Service IWQoS96, Paris, March 6-8, 1996.
- [2] M. Hassan and R. Jain, *High Performance TCP/IP Networking Concepts, Issues and Solutions*, Pearson Prentice Hall, New Jersey, 2004.
- [3] J. Postel, *Internet Control Message Protocol*, RFC 792, IETF, 1981.
- [4] P. Holleczeck, R. Karch, R. Kleineisel, S. Kraft, J. Reinwand and V. Venus, "Statistical Characteristics of Active IP One Way Delay Measurements", *Proceedings of the International conference on Networking and Services (ICNS'06)*, p. 1, 2006
- [5] S. Shalunov, B. Teitelbaum, A. Karp, J. Boote, M. Zekauskas, *A One-way Active Measurement Protocol (OWAMP)*, RFC 4656, IETF, 2006.
- [6] M. J. Luckie, A. J. McGregor, and H.-W. Braun, "Towards Improving Packet Probing Techniques", in *Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*, pp. 145-150, 2001
- [7] R. S. Prasad, C. Dovrolis, M. Murray, K. Claffy, "Bandwidth estimation: metrics, measurement techniques, and tools", in *Network IEEE*, pp. 27-35, 2003
- [8] RFC 3393, "IP Packet Delay Variation Metric for IP Performance Metrics (IPPM)", <http://www.ietf.org/rfc/rfc3393.txt>
- [9] RFC 2030, "Simple Network Time Protocol (SNTP) Version 4 for IPv4, IPv6 and OSI", <http://www.ietf.org/rfc/rfc2030.txt>