FILE TRANSFER PERFORMANCE EVALUATION

R. BEURAN[*], M. IVANOVICI[**], V. BUZULOIU[***]

Am conceput şi implementat un sistem care permite măsurarea parametrilor calităţii serviciilor (QoS) în reţelele de calculatoare. Folosind acest sistem am evaluat performanţele unor componente de reţea şi am studiat în mod obiectiv cerinţele câtorva aplicaţii de reţea, astfel încât acestea să ofere o calitate acceptabilă de către utilizator. In acest articol prezentăm rezultate pentru transferul de fişiere prin FTP şi performanţa protocolului de reţea subiacent, TCP. Am definit două măsuri obiective ale calităţii la nivelul utilizatorului: debitul util şi eficacitatea temporală a transferului. Am identificat relaţia între parametrii QoS şi calitatea percepută de utilizator pentru transferul de fişiere.

We designed and implemented a system that permits the measurement of network Quality of Service (QoS) parameters. Using this system we evaluated the performance of several network devices and studied objectively the requirements of network applications for delivering user acceptable quality. In this article we present results on file transfer by FTP and the performance of the underlying network protocol, TCP. We defined two user-level metrics: goodput and transfer time performance. We identified the relationship between network QoS parameters and user-perceived quality for file transfer.

Keywords: QoS measurement, performance evaluation of network applications, user-perceived quality, file transfer performance, TCP performance evaluation.

Introduction

Network applications with real-time requirements have started to spread on a larger and larger scale over the Internet. This lead to a wider recognition of the issues related to Quality of Service (QoS). We consider QoS to be the fidelity of a system's observable behaviour to expectations: one can only assess quality by comparing the result of a measurement with the expected value for that

[*] Teaching Assistant, Applied Electronics and Information Technology Chair, "Politehnica" University Bucharest, Romania.
[**] Research Assistant, Applied Electronics and Information Technology Chair, "Politehnica" University Bucharest, Romania.
[***] Professor, Applied Electronics and Information Technology Chair, "Politehnica" University Bucharest, Romania.

measurement. Determining the performance characteristics of a network system is the first step in understanding the application-level behaviour. This must be followed by an evaluation of the user-perceived quality (UPQ) for that particular application and establishing the relationship with the measured QoS parameters.

Each network application requires a minimum QoS level in order to run according to user expectations [1], [2]. Network elements along the path cause degradation that accumulates. There is a maximum end-to-end degradation within which the network must deliver the application traffic for it to run in a satisfactory manner. The number of systems designed to correlate the quality differentiation provisioned by networks with the UPQ for specific applications is reduced. Knowing the requirements of applications, such as file transfer or Internet telephony, allows predicting whether a certain connection is valid for a certain application and what will be the perceived quality for that application.

Several projects are currently involved in studying service differentiation by QoS techniques. Mechanisms deployable in order to ensure service differentiation in networks were studied by the Quantum project [3]. TEQUILA concentrates on network service definition and traffic engineering tools built upon DiffServ in order to obtain quantitative end-to-end QoS guarantees [4], [5], [6]. Internet2 started the End-to-End Performance Initiative [7] aiming to create a predictable and well-supported network environment. While these projects focus on network QoS mechanisms, we established first the application requirements such that user expectations are fulfilled. Only subsequently QoS mechanisms may be deployed to meet these requirements.

A number of projects focus on the relationship between network conditions and application performance. A survey on QoS application needs was published by the Internet2 QoS Working Group [8], but their approach is not objective and the conclusions are vague. TF-STREAM reported on best-practice guidelines for deploying real-time multimedia applications [9]. ITU-T defined network performance objectives for IP-based services in [10]. HEAnet reviewed several aspects of perceived quantitative quality of applications [11]. Most of these approaches are qualitative, whereas we aim at creating a quantitative representation of UPQ that can be related to QoS parameters.

File transfer is one of the basic applications running over today's networks. It is largely used for the simple purpose of transferring data between two points using FTP (File Transfer Protocol), but its most important use is within Web browsing using HTTP (Hyper Text Transfer protocol). We studied FTP which is extensively used in LANs, as well as over the Internet. There is also research focusing on HTTP, see for example [12].

File transfer is an *elastic* TCP-based application. TCP tries to occupy as much of the available bandwidth as it can handle. It also adapts its transmission rate to prevailing network conditions – with high loss rates it backs off to a slower

transmission rate. It also provides reliable data transfer by means of its loss recovery mechanisms.

TCP behaviour is analysed by a multitude of researchers. Some of them take the analytical approach [13], [14], [15]. Another path is that of simulation [16], [17]. There exists also the possibility to do experimental work in real networks, to assess raw network performance [18] or to collect traffic traces [19]. Each of these methods has certain advantages and disadvantages related to their accuracy and the range of conditions that are analysed. Emulation is a hybrid performance evaluation methodology enabling controlled experimentation with real applications. We consider that this approach is the most effective; therefore it is an integral component to our approach of studying QoS.

There are two major approaches to measuring network QoS parameters: active and passive. Active measurement implies generation of traffic that is injected in the network systems under test and analysis of their response. Examples of this type are simple applications, such as *ping* and *traceroute*, or more complex ones, like Iperf [20]. Some of them, and also NetIQ Chariot, a commercial product, emulate transaction traffic from real applications and measures response time, throughput etc. A QoS testbed topology was described by the SEQUIN project [21]. For passive measurement on the other hand, one does not interfere with network functioning and monitors the traffic of real applications to compute the network QoS parameters. Hence application behaviour can be correlated with measured network conditions. This is the approach we undertook in our experiments. We built a system [1] able to measure non-intrusively the network QoS parameters and obtained a one-way delay measurement accuracy of 1 μs, for any size packets, up to loads of 100 Mbps.

In parallel with monitoring network traffic for computing QoS parameters, we quantify the perceived quality for applications, in this case file transfer by FTP, based on specifically defined metrics. Consequently we can correlate network conditions with the UPQ for these applications. For FTP this allows studying the performance of the underlying TCP mechanisms under various networks conditions.

Note that TCP can be redesigned to perform better in current best-effort networks. Initially the TCP congestion avoidance and control mechanisms were designed for networks that were relatively slow [22]. Nowadays networks have a much higher bandwidth, therefore new mechanisms could be used to improve performance [23]. However these enhancements are not widely spread, therefore our work concentrates only on standard out-of-the-box applications.

1 Measurement System

The system we designed is shown in Figure 1 in a typical test setup. A detailed description of the system is available in [1]. For a better understanding we present here the basics. We mirror the traffic on the link between two PCs that run the network application under study using FastEthernet taps. This traffic is fed into programmable Alteon UTP network cards. From each packet all the information required for the computation of the network QoS parameters is extracted and stored in the local memory as packet descriptors. The host PCs, which control the programmable NICs, periodically collect this information and store it in descriptor files. This data is then used to compute off-line the following network QoS parameters: one-way delay and jitter, packet loss and throughput. We can calculate instantaneous or average values, and various histograms.

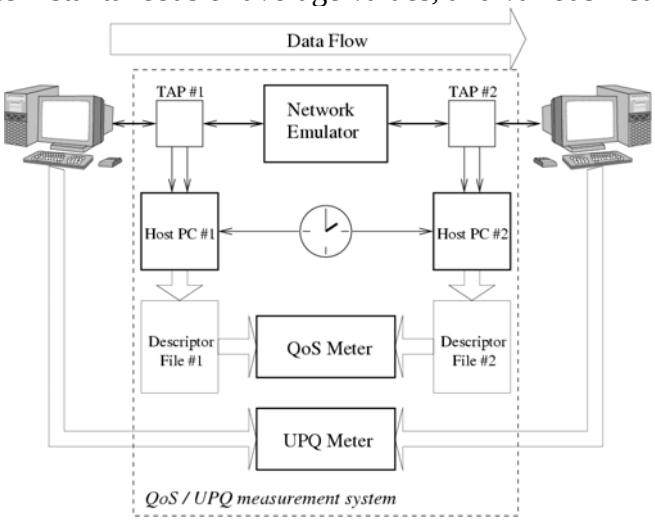

Figure 1. Measurement system setup.

The same collected data is used to assess the UPQ for FTP, using the metrics described in section 3. What follows is the most important step of our approach: correlating the network QoS parameters that have been computed for the connection with the UPQ calculated for the studied application. This correlation allows testing network connections before deploying network applications, and predicting the expected UPQ for those applications.

Our test setup makes use of a network emulator, NIST Net [24]. The emulator can degrade network QoS by introducing in a controlled way artificial delay, jitter, packet loss and throughput limitations. We have used such a solution in order to be able to analyze a wide range of controllable network conditions, while using real applications. This would not have been possible using real networks or simulators.

2 QoS metrics

Based on the data collected by the QoS measurement system we compute off-line the following QoS parameters: average one-way delay and jitter, average throughput and packet loss [25], [26]. The average one-way delay is computed by taking into account only the data frames. The average jitter is computed in three ways, only for the data frames, based on the generic formula (1):

$$J = \frac{1}{N}\sum_{i=2}^{N}\left|D_i - D_{reference}\right|, \tag{1}$$

where J is the average jitter, N is the total number of transmitted data frames, D_i is the one-way delay of each data frame, $i = \overline{1,N}$, and $D_{reference}$ is the reference delay. The reference delay can be the delay of the first frame [25], the average delay or the delay of the previous frame [26]. Since the average jitter an application would experience is influenced by the delay of the previous frame, we consider it the most relevant from an application-oriented perspective. From Figure 2 one can also observe that this is has a smoother variation with packet loss.

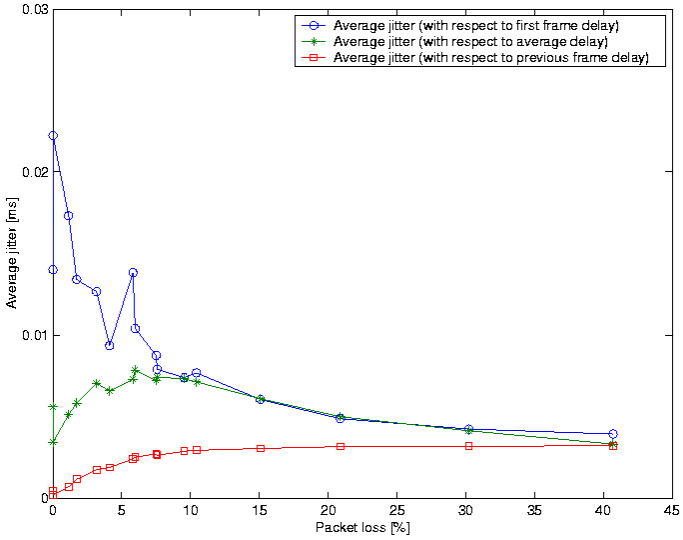

Figure 2. Average jitter computation comparison.

The average throughput is computed taking into account all transferred frames, with respect to the duration of the test. Packet loss is determined using a packet identifier associated by the monitoring system to each frame and written in the descriptors. A packet is considered lost if its identifier, which appears in the descriptor file at the first measurement point, doesn't appear in the descriptor file at the second measurement point.

3 FTP UPQ metrics

A very important aspect of our work is the definition and quantification of application specific metrics. The two UPQ metrics we propose for FTP, *goodput* and *transfer time performance*, allow the assessment of the user-perceived quality for this particular application.

Goodput (G) quantifies the network efficiency of the file transfer. It is computed as follows:

$$G = \frac{B_{\min}\,[bytes]}{B\,[bytes]}, \tag{2}$$

where $B_{\min}$ is the minimum number of bytes required for that file transfer (including protocol overhead for Ethernet, IP, TCP and FTP) and B is the count of the actually transmitted bytes.

Goodput values are on a scale from 0 to 1, where 1 means maximum efficiency of the file transfer. Goodput decreases due to packet retransmission when loss occurs. Given its definition, G doesn't depend on any time parameter related to the transfer (e.g. transfer duration, round-trip time (RTT)) but only on the amount of bytes being effectively transmitted. Therefore an additional metric is required to take this aspect into account.

Transfer time performance (*TTP*) allows the evaluation of the time efficiency for a file transfer:

$$TTP = \frac{T_{th}\,[s]}{T\,[s]} = \frac{B_{\min}[bytes]}{L\,[bps]\cdot T\,[s]}, \tag{3}$$

where T_{th} is the theoretical transfer duration and T is the measured transfer duration. The theoretical transfer duration is the ratio of the minimum number of transmitted bytes required for that transfer, $B_{\min}$, to the line speed, L (in our case 100 Mbps). T is computed as the difference between the time when the last packet from a transfer was received and the time when the first packet was sent.

TTP is also on a scale from 0 to 1, with 1 meaning the ideal, optimum performance. Packet retransmission delays make *TTP* values decrease. *TTP* depends indirectly on all parameters that influence transfer duration, such as RTT, TCP window size etc.

4 Experimental Results

In the experiments we performed, we introduced artificial packet loss using the NIST Net network emulator [24]. Packet loss was introduced in both traffic directions.

We ran our tests using the setup depicted in Figure 1, with different transferred file sizes. The conditions for our file transfer tests were the following: FTP client with Linux kernel 2.4.6 (64 kB maximum TCP window), ftp-0.17-7,

FTP server with Linux kernel 2.4.9 (64 kB maximum TCP window), wu-ftpd-2.6.1-20. In what follows, we present values obtained by averaging over 100 experiments for each intended loss rate. We ran two series of tests, one with a RTT of 0.8 ms (emulating a local network scenario) and the other with a RTT of 60 ms (emulating a wide area network).

We present first two graphs that show the moments of time packets arrive at the receiver and the throughput. The gaps correspond to the delay occasioned by one or more packets being lost, that triggers the retransmission mechanism. This leads of course to a short-term decrease of the instantaneous throughput.

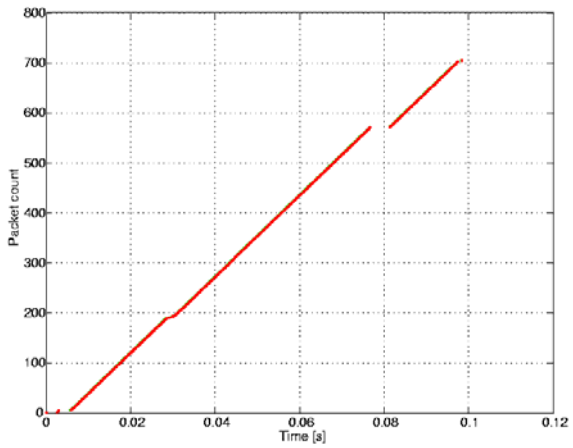

Figure 3. Packet count of received frames with respect to their reception time (file size = 1 MB, RTT = 0.8 ms, packet loss = 1%).

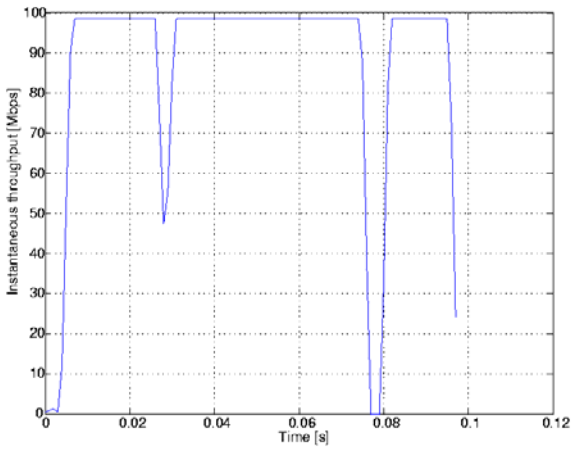

Figure 4. Instantaneous throughput with respect to time (file size = 1 MB, RTT = 0.8 ms, packet loss = 1%).

Table 1 shows the *TTP* values obtained in zero loss conditions for two different RTTs and several transferred file sizes. It can be seen that the time efficiency increases with file sizes, since the overhead of the connection establishment and termination becomes less significant compared to the file transfer time itself. The variation of *TTP* between the two RTTs is of an order of magnitude.

File size		10 kB	100 kB	1MB	10MB
TTP	0.8 ms RTT	0.0219	0.1650	0.8696	0.8919
	60 ms RTT	0.0029	0.0141	0.0559	0.0791

Table 1. Transfer time performance depending on file size and RTT (packet loss = 0%).

TCP window size is an important parameter regarding TCP performance. The optimal window size, $W_{optimal}$, is given by the bandwidth-delay product:

$$W_{optimal} = BW \cdot RTT, \quad (4)$$

where BW is the bottleneck bandwidth of the connection (100 Mbps in our case).

Considering the 0.8 ms RTT we obtain $W_{0.8}$ = 10 kB. For the 60 ms RTT we get W_{60} = 750 kB. Given that the default maximum window size was 64 kB, this doesn't represent a limitation for the 0.8 ms RTT, but it limits the traffic for 60 ms RTT, and the performance is one order of magnitude lower, exactly as observed in Table 1.

The results presented below were obtained for a 10 kB file, which is the typical file size for Internet traffic [27]. For larger file sizes, the graphs of goodput and *TTP* have a similar shape. *TTP* values approach 1 for large files and small RTTs (see Table 1), which shows that it is more efficient to send the same amount of data in one large transfer than in multiple short ones. For these tests, intended packet loss rates ranged from 0% to 25%.

Goodput (see Figure 5) decreases almost linearly with packet loss, showing the diminution of link utilization efficiency. As expected RTT doesn't have any influence on goodput, since *G* is not time dependent. Therefore goodput is not a stand-alone indicator of file transfer UPQ and must be correlated with *TTP*.

Transfer time performance (see Figure 6) shows the significant dependency of transfer time on packet loss. The maximum value of *TTP* equals 0.0219 due to the additional durations of connection establishment and termination, which represent approximately 96% of the transfer time for 10 kB files.

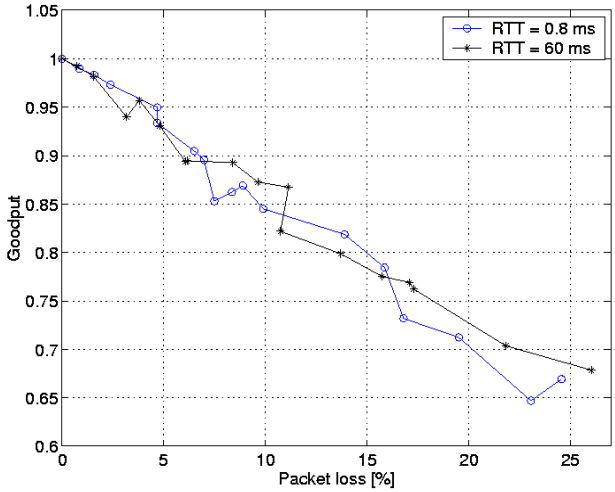

Figure 5. Goodput versus packet loss for file transfer tests (10 kB file).

Figure 6 shows that for 0.8 ms RTT, *TTP* value decreases 20 times for packet loss rates of 5% compared to the value obtained at zero loss. This is equivalent with an increase of 20 times of the transfer duration, which means a significant degradation of the UPQ. For loss rates of 10% and higher, performance degrades hundreds of times. For 60 ms RTT *TTP* is smaller than for 0.8 ms RTT and loss has a less dramatic influence on it.

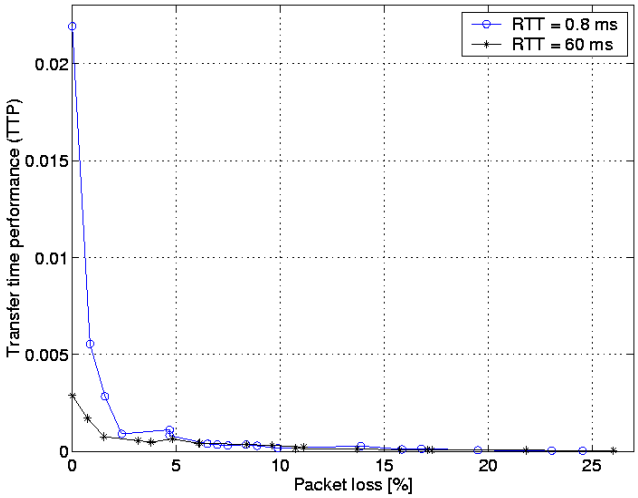

Figure 6. Transfer time performance versus packet loss for file transfer tests (10 kB file).

The influence of packet loss on TCP performance depends on the type of the lost packets: losing a data packet is easily hidden by the retransmission mechanism, whereas losing a TCP connection establishment or termination packet has a more important effect due to the relatively large timeouts. For 10 kB files, transfer duration has increased by an order of magnitude in such cases.

The two figures below show, for comparison. the goodput and TTP for three file sizes (the RTT was 0.8 ms, intended packet loss ranged from 0 to 40%). Note the similarity between the goodput graphs (Figure 7) and the performance improvement with transferred file size emphasized by the TTP graph (Figure 8).

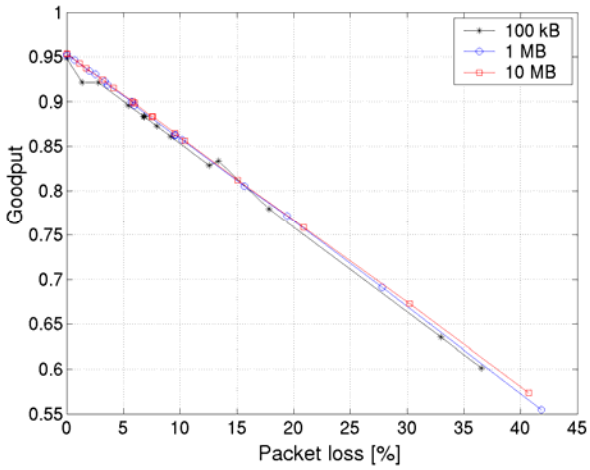

Figure 7. Goodput for three file sizes (RTT = 0.8 ms).

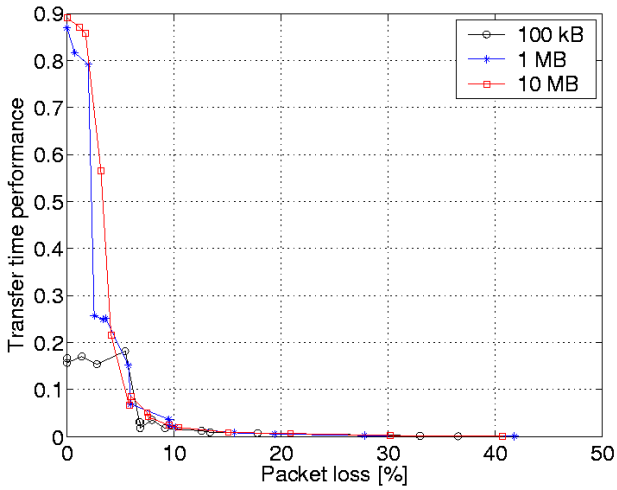

Figure 8. Transfer time performance for three file sizes (RTT = 0.8 ms).

Conclusions

The novelty of our work is that we are able to both accurately measure network QoS parameters and objectively assess application UPQ in parallel. This allowed us to quantify the relationship between QoS parameters and UPQ for file transfer and identify its QoS requirements in a standard configuration.

Goodput diminishes as expected with packet loss. The dependency is linear and goodput decrease is not very large in the range of 0 to 5% packet loss.

Setting the value of 0.9 as the threshold of acceptability for network utilization efficiency, we determine that packet loss should not exceed 5%. For loss rates above 20%, goodput indicates a transfer efficiency lower than 0.7. This approaches 0.5 for loss rates close to 40%.

The transfer time performance graph has a negative exponential shape, showing that the time needed to transfer a file increases significantly with packet loss. For loss rates around 5% and low RTTs, the *TTP* is one order of magnitude smaller than the value obtained at zero packet loss. The degradation observed is less significant for the 60 ms RTT than for the 0.8 ms RTT. At 25% loss rate, the time to transfer has become several hundred times larger than in the case the loss rate is smaller than 5%. This renders the connection practically unusable for file transfer.

We conclude that file transfer applications require packet loss not to exceed 5% in order to keep the network utilisation efficiency above 0.9 and not have an increase of the transfer time larger by more than an order of magnitude with respect to no loss conditions. Good performance requires even tighter bounds: packet loss should not exceed 1% in order to obtain a network utilization efficiency around 0.99 and a transfer time not larger than three times with respect to no loss conditions.

Using our results it is possible to predict an application UPQ based on the corresponding measured network QoS parameters and understand the reasons of possible application failure. One can also determine the end-to-end network QoS requirements for an application to run with a desired UPQ level. Mapping high-level user requirements to network QoS conditions is also a key issue in Service Level Agreement contracts.

REFERENCES

[1] *R. Beuran, M. Ivanovici, B. Dobinson, N. Davies, P. Thompson*, "Network Quality of Service Measurement System for Application Requirements Evaluation", International Symposium on Performance Evaluation of Computer and Telecommunication Systems, SPECTS'03, Montreal, Canada, July 20-24, 2003, pp. 380-387.

[2] *R. Beuran, M. Ivanovici*, "User-Perceived Quality Assessment for VoIP Applications", technical report, CERN-OPEN-2004-007, January 2004.

[3] *T. Ferrari, S. Leinen, J. Novak, S. Nybroe, H. Prigent, V. Reijs, R. Sabatino, R. Stoy*, "Report on Results of the Quantum Test Programme", Quantum Project, June 2000.

[4] *D. Goderis* (editor), "Functional Architecture Definition and Top Level Design", TEQUILA Project, September 2000.

[5] *D. Griffin* (editor), "Selection of Simulators, Network Elements and Development Environment and Specification of Enhancements", TEQUILA Project, May 2000.

[6] *D. Manikis* (editor), "Overview of the TEQUILA Reference Testbeds", TEQUILA Project, February 2001.

[7] Internet2 End-to-End Performance Initiative, http://e2epi.internet2.edu/.

[8] *D. Miras*, "A Survey on Network QoS Needs of Advanced Internet Applications", working document, Internet2 QoS Working Group, December 2002.
[9] *V. Cavalli, E. Verharen*, "TF-STREAM Real Time Multimedia Applications", TERENA Technical Report, March 2002.
[10] ITU-T Recommendation Y.1541. "Network Performance Objectives for IP-Based Services", ITU, draft, October 2001.
[11] *V. Reijs*, "Perceived Quantitative Quality of Applications", http://www.heanet.ie/Heanet/projects/nat_infrastruct/perceived.html.
[12] *J. Padhye, S. Floyd*, "Identifying the TCP Behavior of Web Servers", SIGCOMM 2001, August 2001.
[13] *M. Mathis, J. Semske, J. Mahdavi, T. Ott*, "The macroscopic behavior of the TCP congestion avoidance algorithm", Computer Communication Review, **27**(3), July 1997.
[14] *J. Padhye, V. Firoiu, D. Towsley, J. Kurose*, "Modeling TCP throughput: a simple model and its empirical validation", Computer Communication Review, **28**(4), pp. 303-314, 1998.
[15] *J. Padhye, V. Firoiu, D. F. Towsley, J. Kurose*, "Modeling TCP Reno Performance: A Simple Model and Its Empirical Validation", IEEE/ACM Transactions on Networking, **8**(2), pp.133-45, April 2000.
[16] *K. Fall, S. Floyd*, "Simulation-based Comparisons of Tahoe, Reno, and SACK TCP", Computer Communication Review, **26**(3), July 1996, pp. 5-21.
[17] *L. Breslau, D. Estrin, K. Fall, S. Floyd, J. Heidemann, A. Helmy, P. Huang, S. McCanne, K. Varadhan, Y. Xu, H. Yu*, "Advances in Network Simulation", IEEE Computer, **33** (5), pp. 59-67, May 2000.
[18] *K. Korcyl, G. Sladowski, R. Beuran, R. W. Dobinson, C. Meirosu, M. Ivanovici, M. L. Maia*. "Network performance measurements as part of feasibility studies on moving part of the ATLAS Event Filter to off-site Institutes", Lecture Notes in Computer Science, Springer-Verlag Heidelberg, Vol. **2970**, Grid Computing: First European Across Grids Conference, Santiago de Compostela, Spain, February 13-14, 2004, pp. 206 - 213.
[19] The Internet Traffic Archive, http://ita.ee.lbl.gov/index.html.
[20] *A. Tirumala, F. Qin, J. Dugan, J. Ferguson, K. Gibbs*, "Iperf:", http://dast.nlanr.net/Projects/Iperf/.
[21] *M. Campanella, M. Carboni, P. Chivalier, S. Leinen, J. Rauschenbach, R. Sabatino, N. Simar*. "Definition of Quality of Service Testbed", SEQUIN Project, April 2002.
[22] *V. Jacobson*, "Congestion Avoidance and Control", ACM Computer Communication Review, SIGCOMM '88, Stanford, CA, USA, August 1988.
[23] California Institute of Technology, "FAST Kernel for Large Data Transfers", http://netlab.caltech.edu/FAST/, November 2002.
[24] National Institute of Standards and Technology, NIST Net, http://snad.ncsl.nist.gov/itg/nistnet/.
[25] ITU-T Recommendation I.380, "Internet Protocol (IP) Data Communication Service - IP Packet Transfer and Availability Performance Parameters", ITU, February 1999.
[26] *C. Demichelis, P. Chimento*, "Instantaneous Packet Delay Variation Metric for IPPM", IETF RFC 3393, November 2002.
[27] *M. F. Arlitt, C. L. Williamson*, "Web Server Workload Characterization: The Search for Invariants", Proc. SIGMETRICS, Philadelphia, PA, USA, April 1996.