Evaluation of Real Applications and Services Using the Heterogeneous Network QoS Evaluation Tool

Pedro A. L. Mindlin, Fabrício J. Barth, Edison Spina, Denis Gabos
Polytechnic School – University of São Paulo
Av. Prof. Luciano Gualberto, trav. 3, no. 158.
São Paulo, SP - Brasil
{fabricio.barth,pedro.mindlin,edison.spina}@poli.usp.br,dgabos@larc.usp.br

Abstract: - This paper presents the Heterogeneous Network QoS Evaluation Tool. This tool should help to manage technical influences on the system and allow the consumer/user to evaluate real applications and services from a qualitative point of view. It simulates the behaviour of a real network with the transmission of real data like video streams. We tested our tool using Mpeg2 and Mpeg4 video transmissions over Real Time Protocol. We also propose a method for validating this tool by measuring the distance between its results and real networks results.

Key-Words: - **Simulation, Heterogeneous Network, QoS**

1 Introduction

The rapid advances recently in mobile devices, wireless networking, and messaging technologies have given mobile users an excess of choices to access service contents [9]. Unfortunately, all these devices and protocols, such as Palm PDAs, cell phones with Wireless Application Protocol (WAP) or Short Message Service (SMS) and Digital Video Broadcast (DVB) do not communicate with each other easily. For example, the nature of wireless Internet, or the traditional Internet is very different from Digital Video Broadcast.

In other words, new communication systems proposed for heterogeneous environments use many of the existing network communication systems to perform convergent services over heterogeneous networks.

Heterogeneous networks project management has to coordinate multiple technologies, interfaces, vendors and materials. These characteristics make the infrastructure and business model design more difficult. Quality of Service (QoS) validation is also very hard to perform [8]. In most cases, mock-ups (a usually full-sized scale model of a structure, used for demonstration, study, or testing) do not present real applications (audio and video) and network performance requirements (delay, jitter, bandwidth) to network, application or business analysts [7].

In this case, we need a tool that will allow the evaluation of QoS parameters of the various networks and services in the mobile environment [5,2,1]. This tool should help to manage technical influences on the system, including other system parameters, equipment, models, terminals and middleware. When used with mock-ups of the services, it allows the consumer/user to evaluate real applications and services. The tool also provides guidelines about user behaviour, expectations and acceptability of mobile applications.

However, we need to validate this tool. We should measure the distance between the simulation and the real network. This paper presents a method for measuring this distance.

This paper is organized as follows. In the next section we describe the environment where we focus our work; in section 3 we examine the issues about evaluating end-to-end convergent services over heterogeneous networks; in section 4 we describe the project decisions; in section 5 we present the Evaluation Tool; in section 6 we present the experiments with video transmissions using the Evaluation Tool; in section 7 we describe the performance measurements and in section 8 we describe a method for measuring the simulation distance. Finally, in the last section, we draw some considerations.

2 Environment

The main downstream path of convergent heterogeneous services, mainly video streaming with a high content of data, meta-data and interactivity possibilities, comes through Digital TV system DVB, named DVB-T (Terrestrial) and DVB-H (for handhelds) [6]. The reason for this choice is its huge capability for transmitting digital content downstream. There is, of course, an IP-HEADEND to provide IP content to be transmitted over the

DVB channel. The receiver itself has some capabilities for reception of various sources. Hybrid receivers will be receiving data from the DTV channel but can, if it is not available, receive data from the mobile network. In this case, enough bandwidth must be granted but, above all, the transition from one channel to the other must occur without notice, or with minor consequences for the consumer. When moving the mobile receiver there is a channel switch, named handover, which is called horizontal if done from one cell to another of the same technology and vertical if from one technology to another; the handover must happen without the participation of the consumer, for whom it must be transparent.

When watching a TV program or gaming, the consumer would interact with the service and there should be a return channel, a channel for the interaction information. This return channel, usually created over the mobile network because of its unique UNICAST capability, will play an important role when the services need quick answers from the service provider. The connection capability, the time to connect and the service provider reach ability over the various networks through which the information needs to travel and, on the other hand, the time that the consumer feels that the action has taken, are some of the parameters of quality of service - QoS.

Each network will be playing its own role on the overall service delivery. The Internet will be used many times in this path, from any service provider, service creator, video streaming, return channel and so on. A model for this environment is shown in figure 1.

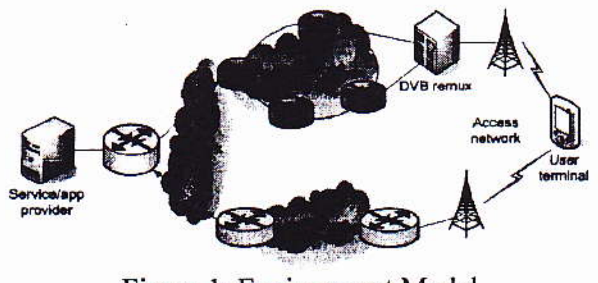


Figure 1: Environment Model

A new era of communication has begun, not only for the communication processes themselves but also mainly for the new consumers that the market brings to business every day. This new environment is creating solutions for problems not even imaginable a couple of years ago.

These new services and their potential consumers are not well known, or not known at all, to the potentials vendors, service providers, etc. The sets of stakeholders [4], infrastructure, equipments, possible contents and so on are too complex and too expensive to be experimented in laboratories.

In this environment, with so many variables, there must be a way to learn before doing any mistakes. The way one can find is to simulate, to emulate, to create a way to learn what the consumer will like and, if possible, how much he would pay for a set of services with a given quality.

The environment where various networks serve as the infrastructure for streaming data, news information, video and TV content, interactivity and mobility need to be integrated with the end-to-end quality perception. End-to-end in its real meaning, not from one border to another of a single network; the complete set of internal network QoS parameters have to be integrated and the consumer must be the main end.

The end that will pay the bill will decide when a service is worth it. This quality perception has to be measured against that promised when the service was sold. There is an agreement - the Service Level Agreement (SLA) - where quality in its many ways will be confronted to the expectations of the client. The human factor study is, in this sense, the better returning investment in the project. Whoever knows better what to sell to who will be in business for a much longer time.

An Evaluation tool that can bring some "reality" to the test environment will bring light to the consumer needs and expectations on whatever service one could imagine. Prototyping the service in a local area network is far cheaper than the real implementation, even in a restricted area or lab. Using mobile equipment, the consumer will feel like the service is real providing that the evaluation tool can bring even the technical QoS problems to the service. The Evaluation Tool concept is in figure 2.

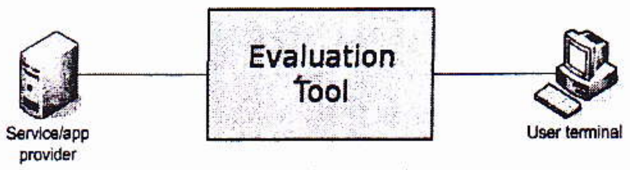


Figure 2: Evaluation Tool Concept.

The concept is to give to the Evaluation Tool the real, measured or simulated, QoS parameters and get, at the consumer end, the quality results to be evaluated by the consumer point of view.

3 Service Design and Quality Assessment Service Design

All stakeholders in such a heterogeneous environment have their own QoS parameters and SLAs or contracts. The main problem is that the consumer does not need nor want to know who does what, who is the responsible for any lack of quality. A lack of quality in any part of the chain is equally responsible for the lack of quality of the overall service. The only SLA that really matters is the one seen by the consumer. Internal QoS parameters or any other problem with technical equipment, traffic problems, billing, etc are not to be addressed by or with the consumer.

The design of any new service must comply with the existing infrastructure and must be evaluated using, if possible, the real environment. Such type of evaluation is often not feasible or too expensive to be implemented. Considering this, the mere existence of an evaluation tool would be a necessity.

Using such tool, a consumer can try a complex service using a mock-up with the added quality problem that the given network infrastructure will have; this network infrastructure could even be one that does not exist or is still in planning. This way the consumer would have a "real feeling" of the services. It would be inexpensive to know the consumer in advance.

4 Design decisions

The idea of the Evaluation Tool is to route Instinct traffic [3], primarily MPEG2 streams, from a service provider machine to a user terminal, emulating on the process the QoS parameters (such as packet loss, delay etc) of the real networks it is substituting, in a way that such traffic appears to the end user the same way it would appear if it were routed through all the real networks.

This idea requires a network emulation engine as the core of the tool. This engine would be controlled by a graphical user interface (GUI) that would provide extra services like time-based variation of parameters and advanced configuration options on top of it.

The first network emulation engine considered was OpNet [10]. OpNet is a powerful network simulator that was already being used by the Instinct Project [3] to design and simulate complex internetworking scenarios. OpNet has the advantage of presenting pre-designed components that model virtually any networking equipment. It is very powerful and highly recommended, though very expensive also. The problem is that, although capable of calculating complex networking models, OpNet is not capable of doing real-time simulations.

This can be explained by understanding the simulation process. A network simulator calculates all operations that happen inside network elements such as routers and determines by these calculations, the time at which each packet arrives at its destination. In order to do a real simulation for a complex network in real time, it would be necessary to have the computing power of all elements of the network together. As this is far too expensive and not realistic, simulators slow down time and compute all operations for this simulated time. Thus, each simulated second can take minutes to calculate, depending on the processing power of the host machine.

Our tool, however, has to receive real video packets in one network interface, delay them for the amount of time that a real set of inter-networks would and send them out on another network interface. The amount of time available for processing a single packet, considering high speed networks, is not enough for a complex simulator like OpNet to complete all necessary calculations, so OpNet simply does not implement such feature.

Another important network simulator, especially in the scientific community, is NS2 [12]. NS2 is, as its website says, "a discrete event simulator targeted at networking research". It is freely available and also much used in networking research. Its models are not so comprehensive, and it is more complex to learn and use. As a simulator, NS2 works in a similar way as OpNet, but it has one advantage: there is a real-time mode. NS2 can communicate with the host machine's network interfaces and exchange packets to and from the simulation environment. In order to emulate a network in this way, NS2 has to perform all calculations in real-time. This is done by simplifying packet treatment, not reading inside too many headers, etc, but even so it cannot cope with high speed data rates. Tests report the use of this scheme to emulate networks with rates at the order of kilobits per second, and our needs go way beyond that to tens of megabits per second. This made it clear for us that using simulators to emulate networks would not be possible, and we had to search for pure network emulating with pre-calculated parameters.

Among the network emulators available on the Internet, NistNet [11] is probably the best known. It runs on top of a Linux machine allowing it to act over passing traffic in order to emulate several different networking conditions. This is done by adding delays to packets, dropping packets, duplicating packets, etc, as commanded by the user through a GUI. It is possible to add different parameters to different flows of data, determined by source and destination addresses. Having pre-calculated network QoS parameters, we would be able to input them into NistNet and emulate that particular network.

Unfortunately, NistNet sources have been unmaintained for the last three years at least, so it takes a lot of effort to get to the end of a complete installation. There are many undocumented problems with it that had to be figured out one by one, with the aid of user email lists, especially the need to have many Linux source packages installed on the system. There were also kernel version compatibility problems.

One possible solution for the problem would be to have different instances of the kernel, each one doing its own routing, with its own network interfaces. This is possible through the UML Project. UML stands for User-Mode Linux - a Linux virtual machine that runs as a user process on top of a real machine [14].

Among the many projects built on top of UML, one of particular interest for us is VNUML - Virtual Network User-Mode Linux [15], built by the Technical University of Madrid. This project makes use of UML virtual machines in order to create a virtual network topology described by an XML file. It creates Ethernet bridges between the virtual machines, that appear to them as real network segments.

Being able to boot the virtual machines and test connectivity between them and the host (the real machine that is running the virtual ones), our first challenge was to route packets through the virtual machines. The first idea was to simply add a route to some destination through one of the virtual machines' network interface. This would not work, however, because the destination network of the packets was a locally connected network, so kernel would route the packets directly to the external interface connected to it. In order to solve this problem, we still needed more than one routing table in the kernel.

That is when we came across Linux Advanced Routing, also known as iproute2 [16]. This is a complete and very powerful networking architecture that has been implemented in Linux kernels versions 2.2 and up, but is rarely used. Linux users usually type *route*, *arp* and *ifconfig* commands, which have been maintained for backwards compatibility, but are only wrappers for the new *ip* suite of commands. This architecture provides means to create separate routing tables and rules (via *ip rule*) to tell the kernel when to consult them. We used it to create rules telling the kernel to consult a certain table only when a packet comes from a certain interface. In this table we would tell the kernel to route packets through the virtual machines. As it came out of the virtual network, the packet would be rerouted by the kernel, only that now it would consult the local routing table, that tells it to route the packet through one of the external network interfaces.

During our research to solve this problem, however, we came across a new piece of information: the Linux kernel had already incorporated, as part of the iproute2 architecture, a queueing discipline called *netem* [13], which adds network emulation capabilities to outgoing queues on network interfaces. If the Linux kernel has this capability, there is definitely no need to use external software like NistNet, especially if it is unmaintained, so we decided to use it with VNUML.

In figure 3 we can see the macro UML Component Diagram of the Evaluation Tool. In this component we have the User Interface component, the Control component and the Scripts component. The User Interface Component contains all the user interface classes. These classes are responsible for the interaction with the users.

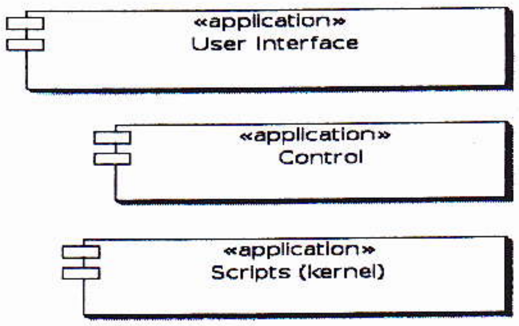


Figure 3: UML Component Diagram

The Control Component contains all the classes responsible for the scripts manager. These classes are responsible for manager the network information and control the scripts that run the virtual networks.

The Scripts component have all the scripts that create the virtual networks and configure all parameters. These scripts are responsible for configure, start and stop the virtual networks developed using *netem* and VNUML.

5 Evaluation Tool

The Evaluation Tool provides the user with a testing environment composed of ten virtual networks (uml1,...,uml5) (figure 4). Each one of these virtual networks can represent one of the real networks in the data path the user wants to study. The data path is divided in two groups: forward path and return path, each of them containing five virtual networks. The forward and return paths are serial sequences of networks that represent how the real networks are distributed between a specific server and a specific client.

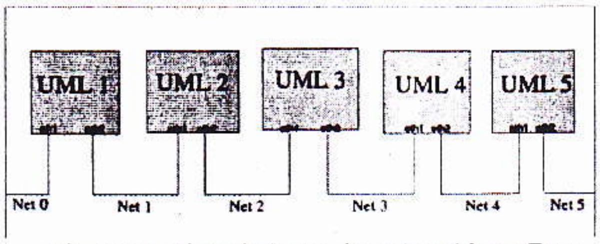


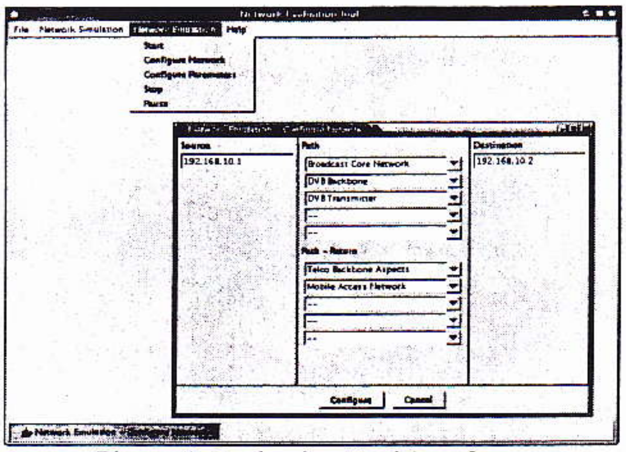
Figure 4: Virtual Networks: Host Linux Box

In order to configure the virtual networks, the tool provides five different types of network configurations. Each network configuration has a name (such as Broadcast Core Network, DVB Backbone, DVB Transmitter, Mobile Access Network and Telco Backbone Aspects) and an array of QoS parameters, and can be applied to any of the ten virtual network interfaces. As of now, the available QoS parameters are: packet delay (ms); packet delay jitter (ms); packet drop (%); packet duplication (%), and bandwidth.

When using the tool, the user has also to tell it who the two end machines are. The tool will have all - and only - the packets flowing between these two addresses routed through the virtual networks. These packets will then be affected by the parameters specified for each virtual network interface, representing the conditions of a real network. As a result, the user is able to see the effects of these networks on real traffic. This means that, instead of analyzing network simulation delay charts, the user can just play a video stream through the Evaluation Tool and see the effects on playback quality.

In order to completely configure the tool for usage, the user has to follow these steps: (i) define source and destination machines; (ii) configure parameters in all network configurations that will be used; (iii) apply network configurations to desired interfaces along the forward and return paths. The Evaluation Tool interface can be seen in figure 5.

It is easy to see that configuring all five parameters for all five possible network configurations, and after that applying these configurations to all ten network interfaces can be a tedious job. For this reason, we have included an XML configuration file import facility. This file can be easily generated from network simulation environments, allowing for integration between QoS parameters generation (simulators) and their usage (Evaluation Tool). For example, if a network design team wishes to test the performance of a new planned network, it generally uses a lot of simulation before deploying it. The data generated by simulation, however, is purely technical, composed of traffic delay numbers at certain nodes, queue sizes etc. For an end-to-end point of view with real data, the team could assemble an XML file with the technical data and upload it to the Evaluation Tool, which would show them the effects of that particular simulation run with real data passing through.



Source: 192.168.10.1

Path:
Broadcast Core Network
DVB Backbone
DVB Transmitter
--
--

Path - Return:
Telco Backbone Aspects
Mobile Access Network
--
--
--

Destination: 192.168.10.2

Figure 5: Evaluation Tool Interface.

Another important feature of the Evaluation Tool is temporal variation. As this tool was designed for mobile environments testing, it has to be able simulate mobility features such as signal strength variation, losses of connectivity etc. These features can be translated into TCP/IP QoS parameters if their values can shift according to a predefined function over time.

6 Experiments

We tested our tool using Mpeg2 and Mpeg4 video transmissions over Real Time Protocol (RTP) without any cache or buffering. We did not use any buffer at these tests because we really want to understand very well how the buffer size will affect this early results in QoS for video jitter and, on the other hand, how it will affect the time response for continued interaction with the consumer. In figure 6 we can see the result of an Mpeg4 video transmission where the network packet drop was 0.1%, the delay was 1.5 ms and the jitter was 0.025 ms.

In figure 7 we can see an Mpeg2 video transmission result with the same drop, delay and jitter values.

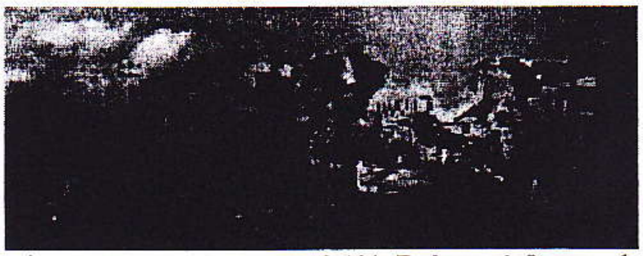


Figure 6: Mpeg4, Drop = 0.1%, Delay = 1.5 ms and Jitter = 0.025 ms

Figure 7: Mpeg2, Drop = 0.1%, Delay = 1.5 ms and Jitter = 0.025 ms

7 Performance Measurements

During our experiments, we noted that sometimes the results were not as clean (video quality) as they should be with all parameters set to zero. This could be due to the introduction of several virtual networks along the data path. In an experiment to measure this interference, we sent approximately 4000 ICMP echo request (ping) packets to a neighbour host, measuring the total round-trip time until the ICMP echo reply came back. The experiment was repeated on four different setups: no virtual machine (the standalone Linux kernel), two, four and six virtual machines in the data path. The results are in Figure 8.

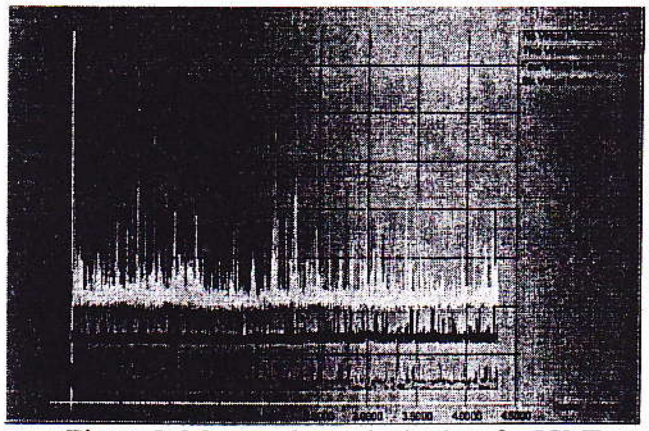


Figure 8: Measured round-trip time for ICMP echo packets

From figure 8 we can see that there is no significant difference between delays of the two first setups, with zero or two virtual machines. There is, however, a significant increase in delay with four and six virtual machines, as long as an increase in delay variation, which can be interpreted as delay jitter. The consolidated data can be seen in fgure 9.

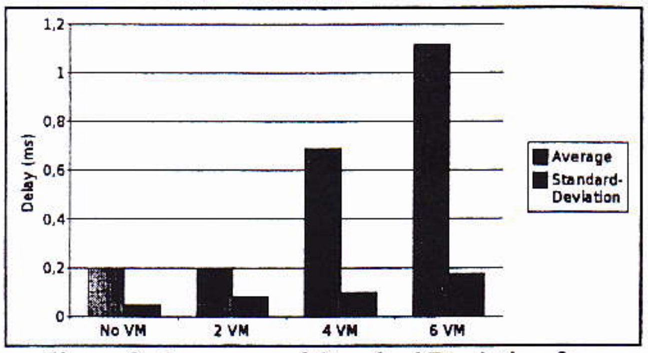


Figure 9: Average and Standard Deviation for Measured Packet Delay

8 Simulation Distance

The purpose of the Evaluation Tool is to produce qualitatively the same results as a real network in terms of Quality of Service. As we are mainly focused on the user perception of Quality of Service, this means that the results produced by the Evaluation Tool must be the same data that would have traveled through a whole network, such as an audio or video stream, for example. It is important, however, to know if and how much these results are reliable.

The process of measuring the Evaluation Tool's simulation distance from a real network is depicted in Figure 10. We can see that this distance is of a qualitative type and is obtained by comparison of the video traces generated by experiments on the real network and using the Evaluation Tool. This comparison is made over a qualitative scale such as (*much worse, worse, same, better, much better*).

In order to obtain the video (or audio) trace, however, the Evaluation Tool needs as input the QoS parameters of the network it is (qualitatively) simulating. These parameters can be obtained from the real network itself. If the real network is not available, however, the QoS parameters can be obtained from a network simulator. The network simulator takes as input a network model and a network load, producing quantitative data that represent the network behavior. It is possible to compare the QoS results obtained from a network simulator with those obtained from a real network, producing a quantitative distance that characterizes the simulator. The QoS parameters are then fed to the Evaluation Tool together with the original video trace in order to obtain the results.

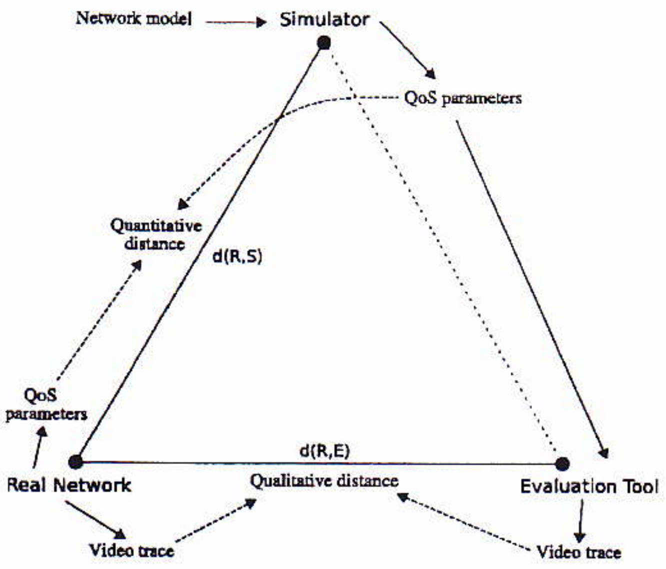


Figure 10: Process of measuring the Evaluation Tool´s simulation distance

We now have the qualitative distance between the real network and the network simulator, and also the quantitative distance between the real network and the Evaluation Tool. By combining these two distances, we obtain the simulating distance between the real network and the simulator/Evaluation Tool set. If it is an acceptable distance, we can now use this combined tool to obtain a video trace (qualitative result) directly from a network model, without needing the presence of a real physical network.

The method used takes the following steps:

1. an experiment (video streaming) is performed on the real network. The video trace and QoS parameters are recorded;

2. a model of the same network, along with a traffic load profile, is used in the simulator to obtain other set of QoS parameters;

3. the parameters obtained from the network simulator are used as inputs to the Evaluation Tool. Another video streaming is performed over the Evaluation Tool and the video trace is recorded.

4. the quantitative distance d(R,S) is obtained by the formula in (1).

$$d(R,S) = (R_{delay} - S_{delay})^2 + (R_{drop} - S_{drop})^2 + (R_{jitter} - S_{jitter})^2 \quad (1)$$

where all values are normalized.

5. the qualitative distance d(R,E) is obtained by first comparing both video traces on the scale presented above and then attributing numeric values for the categories as follows (table 1):

Qualitative value	d(R,E)
much worse	1
Worse	0.5
Same	0
Better	0.5
much better	1

Table 1 - Conversion of quantitative values

6. the final distance is obtained by combining both qualitative and quantitative distances in the formula below:

$$d(R,SE) = \sqrt{d(R,S)^2 + d(R,E)^2} \quad (2)$$

The qualitative part of this method should be repeated many times with different users in order to obtain smooth and statistically significant values.

Another way to measure the distance between the Real Network results and the Evaluation Tool results is to capture the quantitative information about delay, drop and jitter in both sides (figure 11).

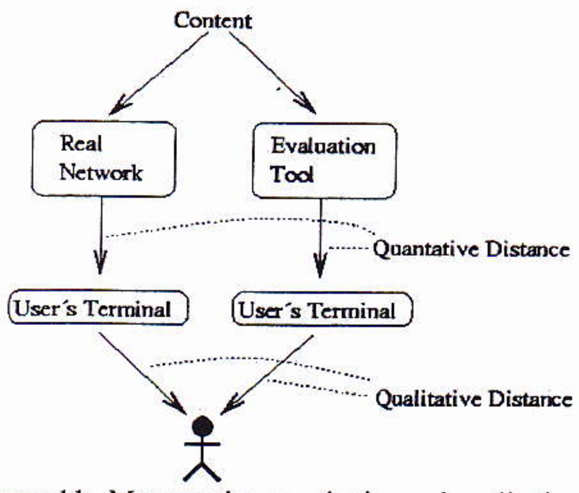


Figure 11: Measure the quantitative and qualitative distance

Delay, drop and jitter information is obtained before the user terminal with a software that traces the content reception.

9 Considerations

This paper described the Evaluation Tool and a method for measuring the distance between the Evaluation Tool results and real networks behavior.

The Evaluation Tool allows the evaluation of QoS parameters of various networks and allows the consumer to evaluate real applications and services from a qualitative point of view.

The simulation distance method is necessary to validate the Evaluation Tool results, check if the Evaluation Tool results match with real network results and return a quantitative information that represents this difference.

This paper describes an ongoing work. We are applying the method described here in experiments with real users in order to assess the Evaluation Tool's usefulness for the Instinct Project [3] partners.

References

[1] A. Diniz. Um serviço de alocação dinâmica de banda passante em redes ATM para suporte a aplicações multimídia. PhD thesis, Universidade Federal de Minas Gerais, 1998.

[2] C. Genilson. Especial Tercerização: consolidação dos servicos de TI. http://www.sucesues.org.br/documentos, Access on April, 26 2005.

[3] Instinct Project, http://www.ist-instinct.org/. Instinct Project Home Page, Access on May, 04 2005.

[4] PMI, Maryland, USA. A Guide to the Project Management Body of Knowledge, 2000.

[5] J. Saltzer, D. Reed, and D. Clark. End-to-end arguments in system design. ACM Transactions on Computer Systems, 2(4),277:288, 1984.

[6] E. M. Schwalb. iTV Handbook: Technologies and Standards. Prentice Hall PTR, July 2003.

[7] Usability NET: A European Union project, www.usabilitynet.org/tools/13407stds.htm. Human centred design processes for interactive systems, Access on May, 04 2005.

[8] A. Vogel, B. Kerherve, G. von Bochmann, and J. Gecsei. Distributed multimedia and qos: A survey. IEEE MultiMedia, 2(2),10:19, 1995.

[9] B. Zheng and D. L. Lee. Information dissemination via wireless broadcast. Commun. ACM, 48(5),105:110, 2005.

[10] OpNet, http//www.opnet.com. OpNet Software. Access on June, 2005.

[11] NistNet, http://www-x.antd.nist.gov/nistnet/. NIST Net Home Page. Access on June, 2005.

[12] NS2, http://www.isi.edu/nsnam/ns/. NS2 Network Simulator. Access on June, 2005.

[13] Netem, developer.osdl.org/shemminger/netem/. Netem – Network Emulator. Access on June, 2005.

[14] UML Project, http://user-mode-linux.sourceforge.net/. The User-Mode Linux Kernel Home Page. Access on June, 2005.

[15] VNUML Project, http://jungla.dit.upm.es/~vnuml/. The Virtual Networks User-Mode Linux Project. Access on June, 2005.

[16] Linux Advanced Routing. http://lartc.org/. Linux Advanced Routing & Traffic Control. Access on June, 2005.

Acknowledgments
This work has been carried out partially through the financial support of the European Commission IST FP6 Integrated Project INSTINCT (IST-1-507014-IP-0e).

WSEAS TRANSACTIONS on INFORMATION SCIENCE and APPLICATIONS

Issue 6, Volume 2, June 2005

ISSN 1790-0832 http://www.wseas.org

ISSN 1790-0832