

Design and Implementation of a Test Bed for QoS Trials

Giorgio Calarco¹, Roberto Maccaferri¹, Giovanni Pau², and Carla Raffaelli¹

¹ D.E.I.S. - University of Bologna
Viale Risorgimento, 2 - 40136 Bologna, Italy
{gcalarco,rmaccaferri,craffaelli}@deis.unibo.it,
WWW home page: <http://www-tlc.deis.unibo.it>

² Department of Computer Science
The Henry Samueli School of Engineering and Applied Science
Campus Box 951596, UCLA, Los Angeles, California 90095-1596, USA
gpau@cs.ucla.edu

Abstract. This paper describes the design, implementation and testing of a test bed supporting flow-based classification functions for multi-service traffic. Protocol and statistical analysis of application flows is performed in the edge routers to provide EF treatment to multimedia traffic without any user signaling. These functions take advantage of the Linux Traffic Control environment and implement SLA management and traffic statistics collection. Sample measurement performed on the test bed shows the effectiveness and feasibility of the proposed solution.

1 Introduction

Quality of Service (QoS) is the capability of a network to forward packets in different ways by grouping them into traffic categories called classes. Several different solutions to the QoS problem have been devised: ATM (Asynchronous Transfer Mode), RSVP (Resource ReSerVation Protocol) [1], the Integrated Services [2] and the Differentiated Services [3] architectures are examples of complementary and interoperable approaches addressing different needs. The Differentiated Services architecture, that is considered here, supports a scalable solution to QoS in IP networks being it based on few fundamental concepts and components: the identification of the packet QoS class through a code point and the differentiated treatment of that packet within a diffserv node as Per Hop Behaviour (PHB). Two main PHBs have been standardised so far: the Expedited Forwarding PHB [4] - for the support of services requiring time guarantees - and the Assured Forwarding PHB [5] - for packet treatment according to three types of drop precedence. PHBs are identified through a 6 bits label, called Differentiated Services Code Point (DSCP) which is placed into the Diffserv Field of the IP header.

In order to permit the end-to-end QoS management a hierarchical, two-tier [6] architecture was also proposed. This model defines the inter- and intra-domain resource allocation, needed to achieve the end-to-end QoS support. The approach

requires the interaction between the RSVP signalling in the stub networks and the bandwidth brokers within the diffserv domains [7]. So the user application should be RSVP capable in order to take benefit from traffic differentiation.

In this paper a new approach to end-to-end QoS is proposed to allow QoS unaware users to access network QoS capabilities in a "plug and play" fashion. The basic idea behind it is a DSCP marking of the traffic based on a content-oriented micro flow classification. The flow classification is made by the edge routers by looking at the traffic aggregate generated within the stub network. The proposed classification process is developed for real time traffic and considers both protocol and statistical analysis of stub network traffic. It is implemented in a Linux environment using Traffic Control utilities.

The paper is organized as follows. In section 2 interactive multimedia traffic characteristics are analysed; section 3 describes the test bed characteristics; section 4 introduces the real time classifier architecture; section 5 focuses on real time classifier functionality and performance evaluation. The paper is concluded with section 6 that summarizes the main achievements and focuses on some ideas for future work.

2 Interactive Multimedia Traffic

From the overall performance point of view interactive multimedia applications are more resistant to packet loss than to high end-to-end delay or jitter when they are transmitted across IP networks. TCP flow control mechanisms assure the correctness of TCP streams but the delay introduced by the retransmission of lost packets creates a bigger damage than the loss itself, if this is reasonable small (i.e. 10%). [8]. Typically these applications are based on the UDP protocol [9]. The impossibility for current best effort IP networks to assure a better service to real-time applications has lead to the development of special protocols. The main contributors on this direction are the IETF and ITU. To address the previous problems, the IETF Audio-Video Transport and Multiparty Multimedia Session Control working groups have developed RTP/RTCP [10] protocols for the transport of real-time content, and RTSP [11] protocol optimised for multimedia streaming. The benefits introduced by these protocols, together with the need for a common standard base, has given RTP the role of standard protocol for the transport of real-time contents over the Internet. The RTP protocol is being used by the most common interactive multimedia applications covering both the commercial and the scientific community as shown in table 1. This means that the use of RTP by an application is a sufficient condition to classify the transmitted data as real-time data. The key concept, behind our classification and marking scheme, is to try to recognise RTP as the protocol used above layer four, typically above UDP.

As regards specific delay requirements, the ITU [12] studies the transmission delay constraints for PSTN. Three different classes of delay that satisfy most of the applications have been identified for connections with adequately controlled echo [12]. In order to keep the end-to-end delay as low as possible, it is better

to transmit the audio stream as a bigger number of small packets, instead of a smaller number of big packets [13]. There are different reasons that justify this choice. Smaller packets are more unlikely to be fragmented or dropped due to buffer management problems and moreover the loss of one packet introduces a very limited source of noise at the receiver side.

Table 1. Most commonly used interactive multimedia tools

	RTP/RTCP	RTSP	RVSP	Audio Video
Netmeeting	Yes	No	Yes	AV
Vic	Yes	No	Yes	V
Rat	Yes	No	No	A
Real Server	Yes(live)	Yes	No	AV

The packet size depends also on audio and protocol aspects. The audio part depends in its turn on the codec frame size and bandwidth while the protocol one is related to the use of different headers (i.e. IP, UDP, RTP). In the case of very limited transmission bandwidth (i.e. analog dial-up modems) the transmission of different audio frame within the same IP packet is required in order to limit the protocol overhead [14]. Given a set of codecs it is possible to estimate the typical mean packet frequency and size for audio conference applications over IP networks in order to keep the end-to-end delay in the range of few hundreds of milliseconds as specified in ITU-T G.114 to assure acceptable quality to delay-aware users [12]. The lower bound for this packet frequency can be considered about 10 packets per second. This value can be drawn from table 2 where the values of packet size and frequency are shown for G.723.1 codec, assuming 24 byte audio frames generated every 20 ms.

Table 2. Rate and packet sizes for G.723.1 codec with 24 byte frames every 20 ms

IP Packets per second	Coding Delay (msec)	Audio payload size (byte)
1	1000	1200
5	200	240
10	100	120

These considerations about delay, with minor differences, can be applied also for interactive video conference applications and videophone connections. Both packet size and frequency values are considered in the classification process.

3 Test Bed Layout and Configuration

In order to perform quality of service trials, a local test bed was designed to emulate functions of a real network environment that offers real time services with quality of service guarantees. Figure 1 shows the test bed layout, which consists of five Intel i810-board systems connected through a Layer2-switch and equipped with a 1Ghz- Pentium III processor and a 256MB bank of RAM. An Internet link is also provided for geographical connection and testing. The edge router (called Alfa) is based on the popular Linux operating system. In detail, a 2.2.19 version of this kernel was used as a developing platform and partially modified to satisfy our aims. It represents the core of the test bed, since it performs the quality of service functions. To this end, it takes advantage of classification, SLA, and bandwidth management utilities, eventually by the interaction with a bandwidth broker (called DeisBB), connected through the switch. 100 Mbit/s Ethernet cards are used. Nevertheless the output links of the edge router are forced to work at 10 Mbit/s to be more easily saturated by test traffic.

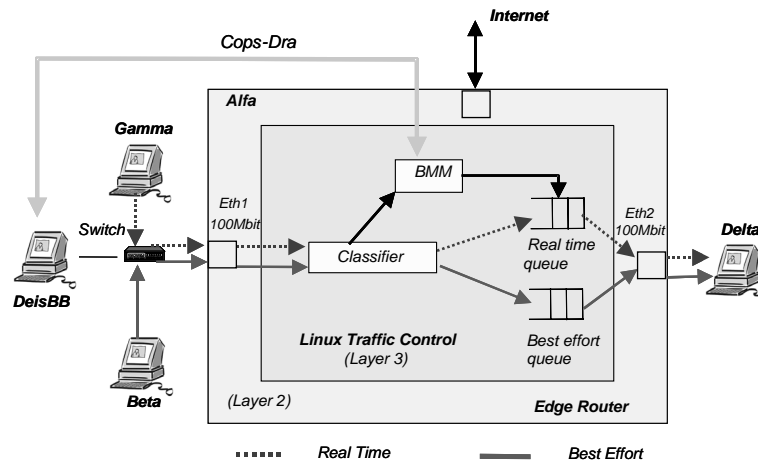


Fig. 1. Functional diagram of the test layout

The other three computers (called Beta, Gamma and Delta) are dual-boot systems having Linux and Windows 2000 installed; they are exclusively utilised for traffic generation and analysis. In particular, Rude, a traffic generator [17] was installed on Beta and Gamma and used for injecting three distinct flows of traffic into the input port of the router. Specifically, these are a real time flow, a

non real time flow (both at 64 Kbit/s), and a best effort flow (at 16 Mbit/s, thus sufficient to saturate alone the output port of the router). Access to the router by Beta and Gamma is obtained through the 10/100 Mbit/s Ethernet switch. A traffic receiver, Crude [17], is set up on Delta: it collects information about the packets coming from the output interface of the router, helping us to verify if the real time flow had been correctly treated. Other applications were also useful for generating the real time flow and evaluating how the system can significantly improve the quality of the communication under a human perspective. Examples of these are the popular Microsoft "NetMeeting", GnomeMeeting and RAT. The traffic control can be configured via the "tc" command, a user-space application which interacts with the Linux kernel to create various objects as queues, classes, SLAs, etc and to initialise them. A graphical front-end interface was also released for easiness of use. The Linux Traffic Control queuing discipline chosen for service differentiation is here based on the CBQ (Class Based Queuing) algorithm. Its configuration assumes two classes, with 1 Mbit/s and 9 Mbit/s rates, respectively, each with a 100 packet-long FIFO buffer attached. Figure 2 shows the basic software architecture of the system. The Bandwidth Management Module (BMM), is a Unix bash script and interacts with both the Linux Traffic Control and the Bandwidth Broker through the COPS client/server protocol (Common Open Policy Service) [18]. Figure 2 shows the architecture of software for bandwidth management. By measuring the real time traffic flows, the BMM decides if the utilised bandwidth is adequate or not. If a bandwidth increase is necessary, it interacts with the bandwidth broker trying to obtain additional bandwidth. The BB keeps the value of the residual bandwidth continuously updated.

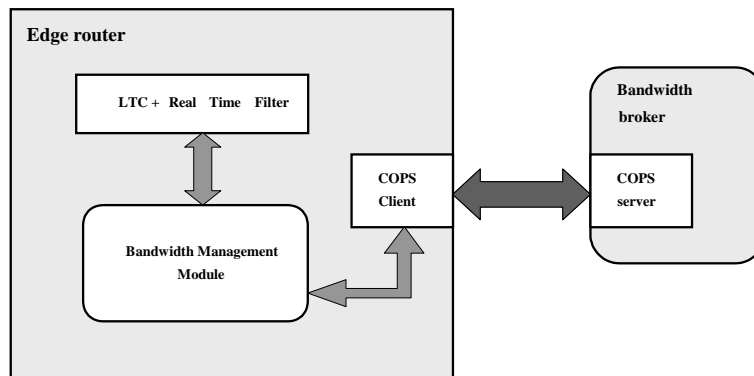


Fig. 2. Software architecture

4 The Real Time Classifier

In this section the approach to allow QoS unaware users of multimedia tools to take benefit of network QoS is described. The new functionality is introduced into the edge router in relation to the network scenario of figure 3, although it can be even introduced within user equipments. The new feature consists in a classifier that, according to a given set of SLAs, performs both protocol and statistical analysis on the traffic incoming from the stub network. The new functionality and its prototype implementation are called Real Time Classifier (RTC). RTC is designed for interactive multimedia applications and, at this moment, it is able to recognize and mark that kind of traffic. In terms of diffserv PHB, RTC marks the traffic recognized as belonging to real-time multimedia streams as EF, setting the IP packet DSCP field. The number of packets necessary used for classification can be chosen independently for each classification algorithm with the aim to optimise the classification delay and failure rate trade-off. RTC

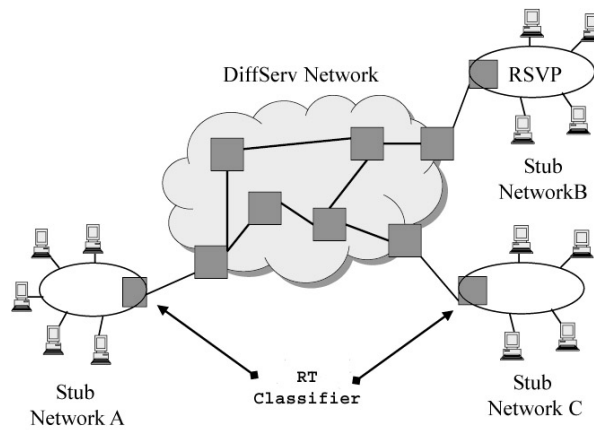


Fig. 3. RTC classifier functionality on the DiffServ Architecture

is actually composed by the following four logical units:

- Classifier: performs traffic classification on protocol and statistical basis;
- Marker: marks the packets according to the classifier policy;
- Meter: meters the incoming traffic;
- Control Unit: manages the SLAs and performs supervision of other units actions.

The control unit has in charge the management of SLA's and the supervision of the whole classification process. For our purposes a 7th-tuple as shown in the table 3 defines a SLA.

Table 3. SLA format

ID	IP	Mask	BW	Shared	DSCP	Policy
----	----	------	----	--------	------	--------

The ID parameter is the unique SLA identifier. The fields IP and MASK are used to identify the host/network belonging to the SLA. The BW parameter is the bandwidth allowed for the considered SLA. The policy parameter can take different values according with the policy adopted for the out-profile traffic of the considered SLA. In particular at the moment the following values are allowed: OK when the out-profile traffic is forwarded as in-profile and no actions is taken; DROP when the out-profile traffic is discarded.

Finally, "shared" is used to specify the degree of fairness to among flows belonging to the same SLA. The value of the field ranges between 0 and 100, and represents the percentage of the bandwidth used on a FCFS basis, with zero meaning that all the bandwidth is equally split between the flows and 100 meaning all bandwidth used on a FCFS basis. RTC control unit has in charge the supervision of whole classification, marking and policing process as a filter between the input interface and the scheduler. Let consider two user A, and B, using a videoconferencing tool (i.e. Netmeeting) across a DiffServ capable network. Suppose to have a RTC capable router as Edge Router. For each Packet coming from stub network RTC control unit performs the following algorithm:

```

when (packet from stub network) is received
{
  If (exists SLA entry for Sender)
  {
    if (packet flow is already classified)
      mark(Flow_DSCP);
    else
      Classify;

    if (Meter(packet,SLA)==out-profile)
      Policy(SLA policy);
  }
  else{
    if (source==untrusted) mark(Best Effort);
  }
  Forward(scheduler); }

```

The "classify" procedure tries a classification of the incoming packet using both application protocol header and statistical information. Once sufficient information has been collected, the flow is classified and its socket tuple is recorded in the hashing table of classified flows. All its subsequent packets are recognized and marked accordingly by matching the value of the tuple.

The RTC classifier is the first functional unit encountered by the traffic entering the Differentiated Services domain from a stub network. RTC is integrated in Linux Kernel QoS Mechanisms named Linux Traffic Control, so once classified and marked a packet can be forwarded using a Linux QoS scheduler.

RTC classifies traffic using both statistical and protocol analysis. The protocol analysis is based on the RTP header characteristics; in particular there are a few parameters within the RTP header keeping constant their values during the whole session. The RTP header has a minimum length of 96 bits and 42 of them remain constants during the whole session.

The protocol-based classification algorithm can be tuned changing the size of the population used to classify. This tuning affects both the precision of the results and the time needed to obtain them. The presence of the RTP protocol in the analysed flow is considered a sufficient, but not necessary, condition to classify the flow as an interactive multimedia stream.

The statistical classification algorithm adds classification capability in the case of real time applications that are not RTP compliant. It takes into account the flow rate and the packet size as main parameters for the classification process. As described in section II, the flow rate depends on the codec used and bandwidth. Typically each IP packet contains one single audio frame. On the other hand when the introduced overhead becomes an issue two or more audio frames are grouped in one IP packet. The number of audio frames per packet is kept as small as possible in relation to the line bandwidth. For example, the codec G.723.1 [15] produces audio frames of 30 ms (33 audio frames per second) and they are generally transmitted one per IP packet if the link bandwidth is enough, but considering a 14.4 kbs modem, they are grouped in three audio frames per IP packet in order to satisfy the bandwidth constraints (table 4).

Table 4. Flow rate and coding delay

	Modem 14.4	LAN
Packet size (Bytes)	100 (3 audio frames)	52 (1 audio frame)
Packet/s	11	33
Bit/s	8800	13728
Overhead	28%	53%
Coding Delay	90 ms	30 ms

Taking into account all the parameters a flow is supposed to be interactive multimedia flow if the number of packet per second will be greater than 10 packets per second. This is the lower bound to keep the delay in the constraints defined by G.114 [15]. In this scenario, in order to obtain an acceptable delay for end users, an application has to encode the audio using more than 10 packets per second. Finally RTC marker unit writes the DSCP value in the DSFIELD of the classified packets according to the classification results. RTC meter unit, in conjunction with the policy unit, performs the enforcement of the traffic profiles.

5 Experimental Evaluation

The RTC tool has been implemented hacking a Linux kernel (release 2.2.19) and the command TC included in the package iproute2 [16] in order to realize a new filter (called RTC) of the Linux Traffic Control. Several tests have been done in order to evaluate classification effectiveness and the amount of resources, in terms of memory and time per classified flow, required for the classification process. The classification process is tested using both the RTP-based and statistical approaches. Three different traffic flows have been generated by Rude [17] in order to saturate the 10 Mbit/s router output link: a real time flow and a non real time flow, both at 64 Kbit/s, and a best effort flow at 16 Mbit/s. Access to the router by Beta and Gamma is obtained through a 100 Mbit/s Ethernet switch.

The Linux Traffic Control queuing discipline for service differentiation is here based on the CBQ (Class Based Queuing). Its configuration assumes two classes, with 1 Mbit/s and 9 Mbit/s rates, respectively, each with 100 packet FIFO queues attached. The main performance figures of interest are the bandwidth used by each flow, the packet loss rate and the time jitter. Jitter is defined with reference to figure 4 as $J_{tx} - J_{rx}$.

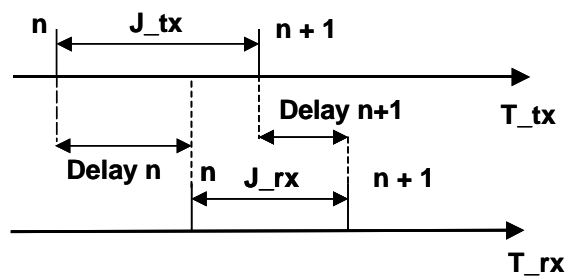


Fig. 4. Main quantities for jitter evaluation

Figure 5 shows the bandwidth usage for real time and non real time traffic during congestion. It is evident that the real time traffic after the classification process has recognized this kind of traffic, obtains the required bandwidth of 64 Kbit/s even if saturation is present. The bandwidth used by non-real time

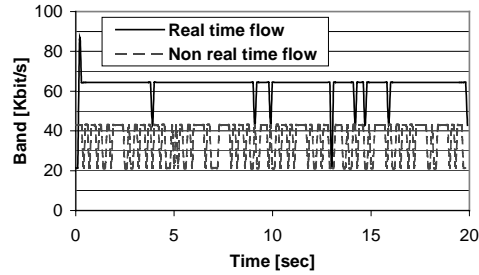


Fig. 5. Link bandwidth usage as a function of time for real time and non real time 64 Kbit/s flows in the presence of a 16 Mbit/s best effort flow over a 10 Mbit/s link.

traffic is on the other hand not stable. Some losses are present for real time traffic due to the layer 2 transmission buffer overflow, where both EF and BE traffics are considered in the same way. These losses can be eliminated by reducing the upstream CBQ queue size for the BE class, thus limiting the BE traffic offered to the transmission queue. The packet loss in time are represented in figure 6 where is evident the different behaviour of the two flows. The packet loss rate

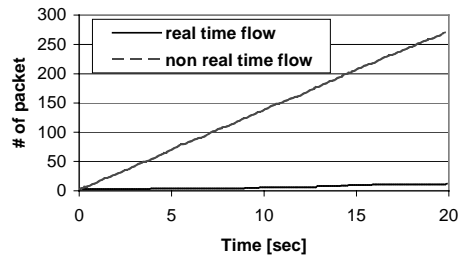


Fig. 6. Packets lost during test for real time and non real time traffic, both at 64 Kbit/s, in the presence of a 16 Mbit/s best effort flow over a 10 Mbit/s link.

for real time traffic has been calculated to be lower than 2 %. The analysis of transmission delay is presented in figure 7 and 8. Figure 7 shows the percentage of packets at different delay for the to 64 Kbit/s flows. The real time flow has a delay limited at 5 ms, much lower than the delay for the non real time flow. Figure 8 evidences the effect of the RTC after the time necessary for recognizing

the real time flow. At the beginning the two flows are dealt in the same way, then, after less than 10 ms, when the classification procedure is completed, the different behaviour in terms of delay is evident. Figure 9 shows the temporal jitter as previously defined for real time and non real time traffic. The values of the jitter for real time traffic are acceptable being within 5 ms after the flow has been classified. As regards the jitter for non real time traffic, it is limited only because almost all non real time traffic is lost and the small percentage of packets that enters the queue typically finds it full.

A temporal analysis focused only on real time traffic has been also performed in the presence of best effort traffic at 16 Mbit/s with the insertion at a given time of the RTC function. In figure 10 the time behaviour for the bandwidth used by real time traffic is considered, showing the transition from a situation with not guaranteed bandwidth to a stable one when the RTC function is active.

6 Conclusions

In this paper the design, implementation and testing of a flow based real time classifier called RTC have been described. RTC uses different methodology to perform its function, based on protocol analysis and traffic patterns. Being it flow-oriented, it can perform functions such as SLAs management and bandwidth usage measurement. This can be fruitfully used in dynamic bandwidth management (i.e. bandwidth broker support). The results show effectiveness of RTC in recognizing real time flows and in guaranteeing bandwidth as limited delays for these kinds of applications.

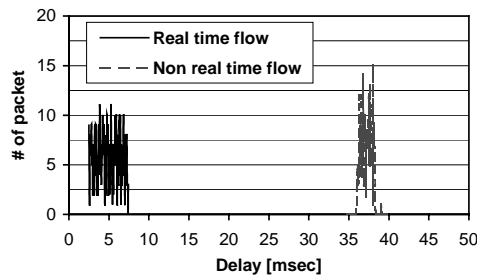


Fig. 7. Distribution of delay for real time and non real time traffic.

7 Acknowledgements

The authors wish to thank Dr. Christian Benvenuti for the technical support in the development of the classification procedure.

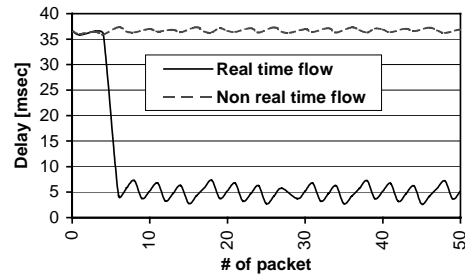


Fig. 8. Packet delay during test for real time and non real time traffic.

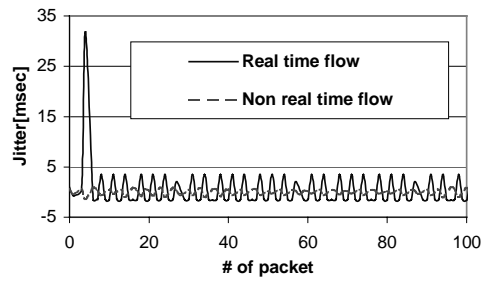


Fig. 9. Time Jitter for real time and non real time traffic.

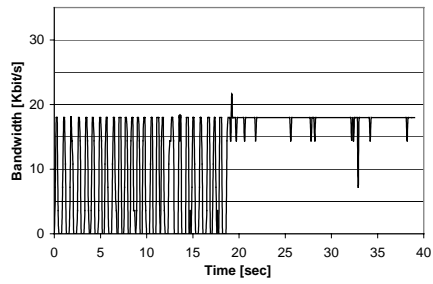


Fig. 10. Effect of the introduction of the RTC function on the behaviour of real time traffic.

References

1. R. Braden Ed. and L. Zhang and S. Berson and S. Herzog and S. Jamin. Resource ReSerVation Protocol (RSVP) Version 1 Functional Specification. Request For Comment 2205, IETF, 1997.
2. Opens H.323 group. Codec Bandwidth and Latency Calculations. Url-<http://www.openh323.org/bandwidth.html>, June 2000. R. Braden, and D. Clark, and S. Shenker, Integrated Services in the Internet Architecture: an Overview. Request for Comments 1633, Internet Engineering Task Force, June 1994.
3. S. Blake and D. Black and M. Carlson and E. Davies and Z. Wang and W. Weiss. An Architecture for Differentiated Service. Request For Comment 2475, IETF, Dec. 1998.
4. B. Davie et al. An Expedited Forwarding PHB. Request for Comments 3246, Internet Engineering Task Force, March 2002.
5. J. Heinanen, F. Baker, W. Weiss, J. Wroclawski. Assured Forwarding PHB Group. Request for Comments 2597, Internet Engineering Task Force, June 1999.
6. K. Nichols, V. Jacobson, L. Zhang. A Two-bit Differentiated Services Architecture for the Internet. Request for Comments 2638, Internet Engineering Task Force., July 1999.
7. Andreas Terzis and Jun Ogawa and Sonia Tsui and Lan Wang and Lixia Zhang. A Prototype Implementation of the Two-Tier Architecture for Differentiated Services. In RTAS99, Vancouver, Canada, 1999.
8. D. Su and J. Srivastava, and Jey-Hsin Yao. Investigating factors influencing QoS of Internet phone. In Proc. of IEEE International Conference on Multimedia Computing and System, pages 308-313, June 1999.
9. J. Postel. User Datagram Protocol. Request for Comments 768, IETF, Aug 1980.
10. H. Schulzrinne, and S. Casner, and R. Frederick, and V. Jacobson. RTP:a Transport Protocol for Real-Time Applications. Request for Comments 1889, Internet Engineering Task Force, Jan. 1996.
11. H. Schulzrinne, and A. Rao, and R. Lanphier. Real Time Streaming Protocol (RTSP). Request for Comments 2326, IETF, Apr. 1998.
12. International Telecommunication Union (ITU). Transmission Systems and Media, General Recommendation on the Transmission Quality for an Entire International Telephone Connection; One-Way Transmission Time. Recommendation G.114, Telecommunication Standardization Sector of ITU, Geneva, Switzerland, Mar. 1993
13. Bolot, Jean-Chrysostome and Garcia, Andres Vega. Control Mechanisms for Packet Audio in the Internet. In INFOCOM, San Fransisco, California, Mar. 1996.
14. Opens H.323 group. Codec Bandwidth and Latency Calculations. Url-<http://www.openh323.org/bandwidth.html>, June 2000.
15. <http://www.itu.int>
16. <ftp://ftp.sunet.se/pub/Linux/ip-routing/>.
17. <http://www.atm.tut.fi/rude/>.
18. R. Mameli, S. Salsano, "Use of COPS for Intserv Operations over Diffserv: Architectural Issues, Protocol Design and Test-bed implementation", ICC 2001, Helsingky.