World Academy of Science, Engineering and Technology
International Journal of Electrical and Computer Engineering
Vol:1, No:12, 2007

QoS Expectations in IP Networks: A Practical View

S. Arrizabalaga, A. Salterain, M. Domínguez, and I. Alvaro

***Abstract*—**Traditionally, Internet has provided best-effort service to every user regardless of its requirements. However, as Internet becomes universally available, users demand more bandwidth and applications require more and more resources, and interest has developed in having the Internet provide some degree of Quality of Service. Although QoS is an important issue, the question of how it will be brought into the Internet has not been solved yet. Researches, due to the rapid advances in technology are proposing new and more desirable capabilities for the next generation of IP infrastructures. But neither all applications demand the same amount of resources, nor all users are service providers. In this way, this paper is the first of a series of papers that presents an architecture as a first step to the optimization of QoS in the Internet environment as a solution to a SMSE´s problem whose objective is to provide public service to internet with certain Quality of Service expectations. The service provides new business opportunities, but also presents new challenges. We have designed and implemented a scalable service framework that supports adaptive bandwidth based on user demands, and the billing based on usage and on QoS. The developed application has been evaluated and the results show that traffic limiting works at optimum and so it does exceeding bandwidth distribution. However, some considerations are done and currently research is under way in two basic areas: (i) development and testing new transfer protocols, and (ii) developing new strategies for traffic improvements based on service differentiation.

***Keywords*—**Differentiated Services, Linux, Quality of Service, queueing disciplines, web application.

I. Introduction

INTERNET provides new business opportunities to service providers, application developers, infrastructure builders and researchers in general, because it allows mobility, sharing information, etc. Therefore the concept of always being connected exists and is demanded more and more by all users so internet is becoming a basic necessity and compel industry and governments to offer Internet services as a mandatory rule [1], [2].

This work was supported in part by the Basque Government, the SMSEs IKUSI and Donewtech Solutions, S.A.

S. Arrizabalaga is with the *Department of Automatic and Control Engineering,* CEIT and Tecnun (University of Navarra), Manuel de Lardizabal 15, San Sebastian 20018 Spain (phone: (+34) 943-212800; fax: (+34) 943-213076; e-mail: sarrizabalaga@ceit.es).

A. Salterain is with the *Department of Automatic and Control Engineering,* CEIT and Tecnun (University of Navarra), Manuel de Lardizabal 15, San Sebastian 20018 Spain (e-mail: asalterain@ceit.es).

M. Dominguez is with the *DIFE Division*, IKUSI-Angel Iglesias S.A., Paseo Miramón 170, San Sebastian 20009, Spain (e-mail: dominguez.m@ikusi.es).

I. Alvaro is with the *DIFE Division*, IKUSI-Angel Iglesias S.A., Paseo Miramón 170, San Sebastian 20009, Spain (e-mail: alvaro.i@ikusi.es).

However, offering internet connection is not enough. Nowadays users, on the one hand, expect a good service, being unacceptable that a user connected in a P2P mode for example, consumes nearly all the bandwidth, leaving the others with no connection at all when all have paid for a connection. On the other hand, developers create new and more powerful applications demanding new and different resources.

So in a scenario where a user demands a service, several questions arise; how services with some QoS management are served? [3] What is the economical model used to pricing?

Regarding the first question, several approaches could be followed: one technique could be to assure QoS based on the type of transferred data, where objectives would be:

1) Keep low latency for interactive traffic, where downloading or uploading files should neither hinder SSH nor Telnet.
2) Allow "navigation" to a reasonable speed while information is being transferred.
3) Assure that sending data does not disturb downloading data, and vice versa.

Another method could be to guarantee a specific bandwidth to each user, without taking into account the type of services it uses. It could also be possible to mix both approaches: to guarantee a specific bandwidth to some users and to manage the remaining bandwidth based on the services they use (QoS assuring strategy). But in either case, the link is used by different types of services at the same time, so, a service differentiation must be made. The Internet Engineering Task Force (IETF) has defined the Differentiated Services (DiffServ) framework [4], [5] as a simple mechanism to provide Quality of Service to traffic aggregates. Services classes and their forwarding treatment are defined by a Service Level Agreement (SLA) and a Traffic Conditioning Agreement (TCA). The TCA is part of the SLA and specifies the practical details of the service parameters for traffic profiles.

Each DiffServ node from a logical point of view has two functionalities: packet classification and traffic conditioning.

The ***packet classification*** policy (Fig.1) identifies the subset of traffic which may receive a differentiated service. Packet classifiers select packets in a traffic stream based on the content of some fields of the packet header.

A ***traffic conditioner*** (Fig.1) may contain the following elements: meter, marker, shaper and dropper. A traffic stream is selected by a classifier (based on the content of some fields of the packet header), which steers the packets to a logical

instance of a traffic conditioner. A meter is used (where appropriate) to measure the traffic stream against a traffic profile (previously determined in TCA). The state of the meter with respect to a particular packet may be used to affect a marking, dropping, or shaping action. Shapers delay some or all of the packets in a traffic stream in order to bring the stream into compliance with a traffic profile.

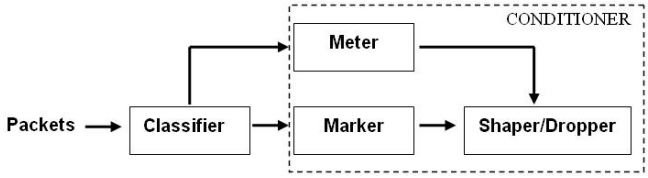

Fig. 1 Logical view of a packet classifier and traffic conditioner

Once several classes have been defined and their traffic profile specified (a kind of SLA agreement) the developed architecture follows the Differentiated Service philosophy in terms of classifying and marking packets to differentiate services. As a component of the link sharing, the packet scheduler performs the central task of selecting a packet to transmit when the outgoing link is ready and also performs the task of shaping, by delaying or dropping packets to make a traffic flow conform to the configured traffic profile.

But what about the facilities that are offered by today's operating systems for its implementation? As explained in [6] traffic control capabilities have been available in the Linux kernel since the 2.2 series. Shaping configurations are implemented using a variety of available packet schedulers and shapers which can be configured using the ***tc*** binary included in the ***iproute2*** package of tools [7]. ***Netfilter*** [8], [9] framework also aids the ***tc*** and ***iproute2*** systems used to build sophisticated ***QoS*** and policy routers. ***Qdisc*** is the term used to refer to these schedulers under Linux. Apart from ***qdiscs,*** there are more traffic control components in Linux, such as classes (used for shaping functionalities) or filters (capable of classifying packets), both also configurable by the ***tc*** binary. For further details, Linux network traffic control is explained in depth in [10], and how Differentiated Services are supported in Linux in [11].

Therefore **qdisc** also called a queuing discipline is the major building block which all Linux traffic control is being built on. Basically there are two types of **qdiscs**:

The **classless queueing disciplines**: contain neither classes, nor is possible to attach filters to it. Since a classless **qdisc** contains no children of any kind, there is no utility to classify packets. This means that no filter can be attached to a classless **qdisc**. Examples:

1) The simple **drop-tail FIFO**: a FIFO stores and sends packets in the order in which they arrive ("First-In First-Out"). A drop-tail FIFO drops newly arriving packets when it has reached its maximum size.
2) A **Random Early Detection** (RED) FIFO [12]: RED starts dropping packets before reaching the maximum queue size, so that congestion-controlled protocols like TCP can slow down in time.
3) The **Token Bucket Filter** (TBF), a shaper that emits packets at a fixed rate.
4) **Stochastic Fair Queueing** (SFQ) [13], one of the several implementations of a set of algorithms called Fair Queueing (FQ) [14], [15]. The basic idea of the SQF is to provide to each flow with a fair share of resources.

The **classful queueing disciplines:** contain classes, and provide a handle which attach filters to. There is no prohibition on using a classful **qdisc** without child classes, although this will usually consume system resources for no benefit. Important queueing disciplines include:

1) PRIO (Priority scheduler): in classic Priority Queueing, packets are first classified by the system and then placed into different priority queues. Packets are scheduled from the head of a given queue only if all queues of higher priority are empty. Within each of the priority queues, packets are scheduled in FIFO order [16].
2) Schedulers implementing the model called "Hierarchical Link-Sharing" described in [17]. First, "Class Based Queueing" (CBQ) was created, but a better implementation of the model and easier in its configuration called "Hierarchical Token Bucket" (HTB) [18] is recommended to be used nowadays. A key feature is that the model gives us some tools to avoid starvation of a low priority class, as the framework has the ability to share bandwidth between classes with different priorities. The model also allows a packet scheduler to be assigned to each class, which determines which packet is selected when the class is allowed to transmit a packet.

Concerning the second question, two facts must be considered: a pricing model and a simple mechanism for payment. From the pricing point of view, again several alternatives: logging time, amount of transferred data, type of service, etc. From the payment point of view the user should have as many possibilities as possible to make the payment. A new bonus system could be designed so that the user could freely use services or simply pay at the end of the connection.

II. Problem and Implemented Solution

A. Problem

In our case the problem to be solved is posted as: there exists a SMSE that wants to offer public access to Internet with certain Quality of Service expectations to both administrative staff and general domains, the solution must be cost effective and easy to use and to install so that it provides new business opportunity.

B. Problem Solution

The design and the implementation of the solution to our problem are based on the answers to the two questions above described: How services with some **QoS** management are served? What is the economical model used for pricing?

To "how services with some **QoS** management are served?" we propose a topology based on three networks, a single link

connection shared by both domains and guaranteeing no interference between them, Linux as the Operating System and open source software products (Fig. 2). In this scenario to have exclusive internet connections for each domain may not be the best economical solution. Therefore we believe that sharing a connection is a good alternative.

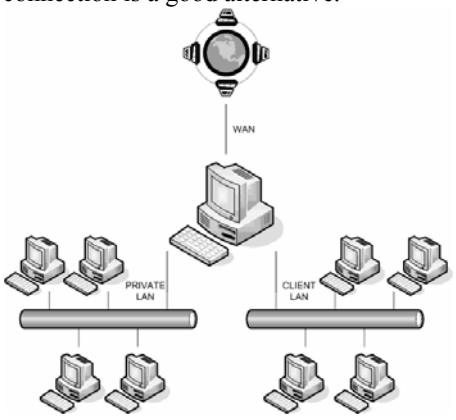

Fig. 2 Basic architecture

Fig. 2 shows the basic connection structure of the design:

1) *WAN*: denotes the connection to Internet.
2) *Private LAN*: This *LAN* represents the network which management staff and other SMSE´s personnel are connected to. This *LAN* network has not any restriction to access to Internet.
3) *Client LAN*: This *LAN* stands for the network where a general customer is connected to. The administrator may force to this network's users into authenticating before accessing to Internet, or may restrict internet connectivity based either on user's navigation time or on the amount of transferred data (sent or received) by the user.

To the question of "what is the economical model used for pricing?" a new bonus model that offers different connection modalities has been design to facilitate the payment:

1) Prepaid bonus for different amount of bytes (500 kB, 1 GB, for example)
2) Prepaid bonus for different time-steps (half an hour, 2 hours, 10 hours, for example)
3) Consumed data bonus (payment based on the amount of data being transferred)
4) Consumed time bonus (payment based on logging connection time)

We are of the opinion that the utilization of bonuses is very important, firstly, because they are sharable, that is, a bonus can be used by several customers simultaneously allowing, organizers of meetings, lecture classes etc., to use just one bonus, and secondly, because bonuses can be assigned to special users. Definitely, this type of pricing represents a clear business opportunity to the SMSE.

C. Implementation

A web-based user-friendly interface with a high level of abstraction has been developed which allows the SMSE's administrator to keep up to date the system configuration without knowing how it really works.

There are several parameters which can be configured by the administrator, however, based on Differentiated Services philosophy, the administrator simply defines, specifies and groups services into classes or levels which will have different treatment. Selected configurations are saved and retrieved to/from a database and scripts are generated and executed internally and transparently.

Based on DiffServ philosophy, and concerning the **packet classification**, in each of the interfaces, traffic is classified based on the services and classes defined for that interface. For each defined service, an ***iptables*** rule [9] is introduced, to classify and to mark all traffic that follows the pattern identifying that service. All services grouped in the same class are marked with the same specific number. This number will be used by traffic conditioning module to give packets from each class different treatment. Concerning **traffic conditioning**, a Class-Based-Queueing (HTB queueing discipline) packet scheduler [18] has been selected because its hierarchical approach is well suited for setups where a fixed amount of bandwidth is divided for different purposes, giving each purpose a guaranteed bandwidth, with the possibility of specifying how much bandwidth can be borrowed. Moreover, it can be combined with SFQ queues, trying to prevent any single client or flow from dominating the network usage. Summing up, traffic is selected and marked by iptables (with same number for each class). Based on this identifying number, filters attached to the root **qdisc** are responsible for assigning packets to the associated HTB class, which represents the differentiated service treatment for each class. To end, Stochastic Fair Queueing scheduler is used within each class to achieve a fair sharing of resources within each class (Fig. 3).

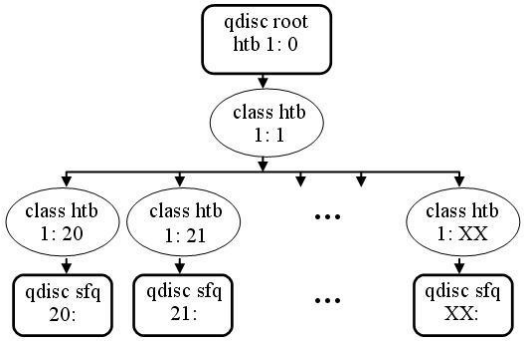

Fig. 3 Implemented scheme

Traffic classification: The administrator defines the services that need to be differentiated. By default the most common services are already defined in the web application, but the administrator can modify, delete and introduce new services. A service is defined by a set of parameters, basically:

1) *Name*: identifies the service; for example, HTTP, SMTP.
2) *Protocol*: "ICMP", "TCP", "UDP" or "TCP/UDP"
3) *Type* (enabled when protocol is TCP or UDP): "port" or "packet's length".
4) *Value* (enabled when type is defined): numeric value. In

case *type* has been defined as a "port", single ports or port ranges can be defined. In case *type* is defined as "packet's length", an integer defining packet's length is introduced. It is recommended to classify short packets, ACKs, etc.

This first step does not associate services and classes. This association will be done in the following steps.

Service classes and differentiation: Firstly, the administrator must specify the amount of bandwidth that is available (download/upload from/to Internet), and decide how to divide it between both domains: *Client LAN* and *Private LAN*. This information is obtained by requesting:

1) Download rate (kb/s)
2) Upload rate (kb/s)
3) Percentage of available bandwidth for Client LAN (%)

Since, the developed application supports service differentiation in **outgoing** traffic in the network interfaces it is necessary to know the amount of bandwidth that is actually available in each interface. These approximations are made:

1) *WAN* interface outgoing traffic is interpreted as upload traffic
2) *Client LAN* interface outgoing traffic is interpreted as download traffic in *Client LAN*
3) *Private LAN* interface outgoing traffic is interpreted as download traffic in *Private LAN*

It should be clear that, these approximations do not lead to error since outgoing traffic in both *Client LAN* and *Private LAN* comes from *WAN* interface (traffic between *Client LAN* and *Private LAN* is not generally allowed, if it were permitted this traffic would be minimum due to the functionality of the architecture). Consequently, available bandwidth in each interface is obtained after applying a penalization factor (0.90) to the theoretical values as shown below:

1) *WAN* interface: 0.90 * *upload rate*. *(kb/s)*
2) *Client LAN* interface: 0.90 * *download rate* * *percentage* / 100 *(kb/s)*
3) *Private LAN* interface: 0.90 * *download rate* * (100 – *percentage*) / 100 *(kb/s)*

Secondly, it is necessary to group services defined in the previous step into service classes and to define traffic conditioning rules for each traffic class as if they were bandwidth rules for the administrator. Bandwidth rules are defined in each one of the network interfaces. To define a rule, it is necessary to set the following parameters:

1) *Priority*: "Low", "Medium" or "High".
2) *By default*: "Yes" or "No". Only one of the rules of each network interface must be defined as default. This is necessary to define which rule will be applied to the services that have not been specifically defined.
3) *Minimum BW (%)*: Minimum BW guaranteed.
4) Maximum BW (%): If BW is available, maximum BW will be used.
5) Type: "Services" or "User". In case that "services" is selected it is necessary to specify which ones will be included among the services that were defined in previous step (to help the administrator, a list box will be shown with all defined services). In case that "user" is selected, this rule will be reserved for a user that asks for a guaranteed bandwidth no matter what services are used.

These rules automatically generate an identification number for ordering and identification purposes. Each of these rules is a service class, where applications are included, having all of them the same treatment, defined by the HTB class assigned to them.

Pricing and counting: To price and to count *Client LAN*'s users' consumption, their accesses are controlled through a captive portal functionality. To allow user navigate correctly after authentication, and once user's ip address has been obtained by the servlet attending user's authentication petition, new iptables rules are introduced. These specific rules are responsible for:

1) Skipping http redirection rules
2) Data transfer counting rules
3) Packet marking rules (in case the bonus has got associated a guaranteed bandwidth)

When a user closes its session, session's consumption is saved into a database and rules are deleted.

This operational mode indicates that, the administrator must define all the modalities that will be offered to the users. These will be the patterns of the individual bonus that will be created when client requests them. These patterns specify how long the bonus will be valid, if it is prepaid or not, if it is based on time or transferred data, how much it costs (if it is prepaid) or how much the defined unit on time (hour, minutes, etc) or on data quantity (bytes, kbytes, etc) costs.

III. Testing

To evaluate the performance of the developed application a thorough testing has been performed to one interface. For that a PC acting as traffic generator is placed in the *Private LAN,* and another PC is positioned in the *Client LAN* to collect the generated traffic. *WAN* network has not been used since total control on the networks is needed. To carry out this testing, communication between *Private LAN* and *Client LAN* has been enabled. A software tool called IPERF [19] is used for traffic generation (iperf acting as a client) as well as for generated traffic collection (iperf acting as a server). For monitoring use of bandwidth, a perl script (*monitor_tc.pl*) [20] is used. It monitors the HTB classes by using tc utility periodically and writes down bandwidth consumption information in a log file.

In our case four flows are generated at different time steps. In all cases, source is Client A, connecting to Server B in a specific port.

1) *Flow 1*: port 5001, from t=1 minute to t=5 minutes.
2) *Flow 2*: port 5002, from t=2 minutes to t=6 minutes.
3) *Flow 3*: port 5003, from t=3 minutes to t=7 minutes.
4) *Flow 4:* port 5004, from t=4 minutes to t=8 minutes.

And four services have been created to classify these flows. Following the description given in *traffic classification*, services' *names* are "Flow 1", "Flow 2", "Flow 3" and "Flow 4", all of them are characterized by "TCP" *protocol* and "port"

Type. The number of the port is the previously mentioned for each one of the flows.

In *Client LAN*, four rules are created, one for each service (*F1*, *F2*, *F3* and *F4*). Traffic generation process is the same in all of the tests, as well as the services associated to each rule, but rules' characteristics are modified as described.

Client LAN's download bandwidth (after considering a penalizing factor) is 1620 kb/s, and the rate parameter for all rules is 405 kb/s (25% of the available bandwidth (*Minimum BW*)). In Test 1 and Test 2, *Maximum BW* is 100%, whereas in Test 3 and Test 4 it is 50%. In Test 1 and Test 3, all rules have the same *priority*, whereas in Test 2 and Test 4, *F1* has high *priority*, *F2* and *F3* have medium *priority* and *F1* low *priority*.

Tests are done from t = 0 seconds to t = 9 minutes. Fig. 4 shows the results graphically and in Table I some comments about them are made. Traffic limiting works at optimum and so it does exceeding bandwidth distribution.

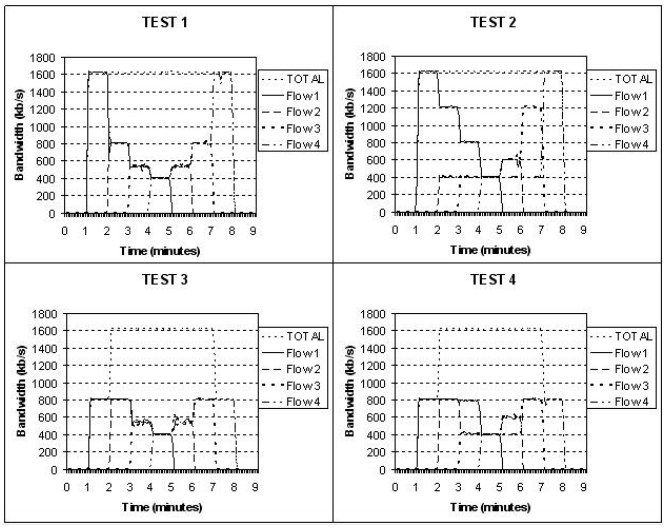

Fig. 4. Results for tests.

TABLE I
COMMENTS ABOUT TESTS

Time interval (min)	Existing flows' number	TEST 1 • Max BW 100% • = priority	TEST 2 • Max BW 100% • ≠ priority	TEST 3 • Max BW 50% • = priority	TEST 4 • Max BW 50% • ≠ priority
1-2	*F1*	Assured 25% for F1. All remaining BW is used by F1 as there is no more traffic and F1 can use up to 100 %.	Same as TEST 1.	Assured 25% for F1. Remaining BW: 75%. As there is no more traffic, F1 uses its maximum quantity of BW, 50%.	Same as TEST 3.
2-3	*F1-F2*	Assured 25% for both F1 and F2. Remaining BW is 50%, which is shared by F1 and F2 (25% for each) as they have the same priority.	Assured 25% for F1 and 25% for F2. Remaining BW is 50%, which is used only by F1 as it has higher priority than F2. So 25% for F2 and 75% for F1.	Same as TEST 1.	Assured 25% for F1 and 25% for F2.Remaining BW is 50%, which is firstly used only by F1 as it has higher priority until it arrives at its maximum (50%). At this point, rest of BW (25%) is used by F2. So, 50% for F1 and 50% for F2.
3-4	*F1-F2-F3*	Assured 25% for F1, 25% for F2 and 25% for F3. Remaining BW is 25%, which is shared between F1, F2 and F3 as they have the same priority.	Assured 25% for F1, 25% for F2 and 25% for F3. Remaining BW is 25%, which is used by F1, as it has the highest priority and can use up to 100% of the total BW.	Same as TEST 1.	Assured 25% for F1, 25% for F2 and 25% for F3. Remaining BW is 25%, which is used by F1, as it has the highest priority and can use up to 50% of the total BW.
4-5	*F1-F2-F3-F4*	Assured 25% for each flow. There is no remaining BW	Same as TEST 1	Same as TEST 1	Same as TEST 1
5-6	*F2-F3-F4*	Assured 25% for F2, 25% for F3 and 25% for F4. Remaining BW is 25%, which is shared between F2, F3 and F4 as they have the same priority.	Assured 25% for F2, 25% for F3 and 25% for F4. Remaining BW is 25%, which is shared between F2 and F3, as they have the same priority (and higher than F4's).	Same as TEST 1.	Assured 25% for F2, 25% for F3 and 25% for F4. Remaining BW is 25%, which is shared between F2 and F3, as they have the same priority (and higher than F4's).
6-7	*F3-F4*	Assured 25% for both F3 and F4. Remaining BW is 50%, which is shared by F3 and F4 (25% for each) as they have the same priority.	Assured 25% for F3 and 25% for F4. Remaining BW is 50%, which is used only by F3 as it has a higher priority than F4. So 25% for F4 and 75% for F3.	Same as TEST 1.	Assured 25% for both F3 and F4. Remaining BW is 50%, which is used firstly by F3 as it has a higher priority, but when it arises up to its maximum (50%) the rest of the BW is consumed by F4.
7-8	*F4*	Assured 25% for F4. All remaining BW is used by F4 as there is no more traffic and F4 can use up to 100 %.	Same as TEST 1	Assured 25% for F4. Remaining BW: 75%. As there is no more traffic, F4 uses its maximum quantity of BW, 50%.	Same as TEST 3

IV. Concluding Remarks

A solution to a SMSE´s problem as a first step to the optimization of QoS in the Internet environment has been presented and thoroughly tested. The obtained profiles show that the main objectives: simplicity, easily configurable, pricing modalities based both on login time and/or quantity of data and a relatively complete bandwidth management solution are satisfied.

The profiles (Fig. 4) and comments (Table I) show that, exceeding bandwidth distribution works as desired and assured bandwidth is always achieved. Bandwidth limiting works also at optimun.

However, some considerations must be done:

The approximation of considering Client LAN interface's outgoing traffic as download bandwidth for users, Private LAN interface's outgoing traffic as downloading bandwidth for management staff and WAN interface's outgoing traffic as upload bandwidth, imposes some limitations:

--Download bandwidth separation between Client LAN and Private LAN is static: one LAN's unused bandwidth is not available for the other LAN.

--There is no separation between upload traffic from Client LAN and upload traffic from Private LAN.

Yet, improvements could be made by using IMQ (Intermediate Queueing Device) [21], as it allows to shape over multiple interfaces.

b) For this first implementation, some restrictions have been assumed:

--Configuration rules are static. Bandwidth parameters need to be assigned statically.

--Traffic classification's filters are limited to ICMP protocol, tcp/udp ports or packet length.

--All rules are converted to HTB classes which have SFQ queues attached, however, for interactive traffic SFQ queues are not the best solution as SFQ perturb causes packet reordering.

c) Traffic classification could be improved by several ways:

--L7 filters [22] (filters that are able to classify application level packets) and ipp2p filters [23] (filters that are able to detect peer to peer traffic). This is an area that is being improved continuously.

--Flag detection in TCP packets: SYN, FIN, RST.

Currently research is under way in two basic areas: (i) development and testing new transfer protocols, and (ii) developing new strategies for traffic improvements leading to the optimization of network level QoS parameters [24] in IP networks and also for new ways of pricing based on service differentiation.

References

[1] Behrouz and A. Forouzan, *Data Communications and Networking*, Ed. New York: McGraw-Hill, 2003.

[2] Alberto Leon-Garcia and Indra Widjaja, *Communication Networks*, Ed. New York: McGraw-Hill, 2004.

[3] Ajmone Marsan et al. (Eds), *Quality of Service in Multiservice IP Networks*, QoS-IP 2005. Springer-Verlag, 2005.

[4] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, *RFC2475 – An architecture for differentiated service*. Technical report, RFC Editor, December 1998.

[5] D. Grossman, *RFC3260 – New terminology and clarifications for diffserv*. Technical report, RFC Editor, April 2002.

[6] Jason Boxman. (2005, November 15) *A practical Guide to Linux Traffic Control.* [Online]. Available: http://edseek.com/~jasonb/articles/traffic_shaping

[7] Bert Hubert, Thomas Graf, Greg Maxwell, Remco van Mook, Martijn van Oosterhout, Paul B Schroeder, Jasper Spaans, and Pedro Larroy. (2002, March). *Linux Advanced Routing & Traffic Control*. [Online]. Available: http://lartc.org/howto

[8] Harald Welte, et al. (2005). *Netfilter/iptables Project*. [Online]. Available: http://www.netfilter.org

[9] Oscar Andreasson. (2005, July 20). *Iptables Tutorial 1.2.0*. [Online]. Available: http://iptables-tutorial.frozentux.net/iptables-tutorial.html

[10] Werner Almesberger. "Linux Network Traffic Control - Implementation Overview". *Proceedings of 5th Annual Linux Expo*, Raleigh, NC, pp. 153-164, May 1999. Available: http://www.almesberger.net/cv/papers/tcio8.pdf

[11] Werner Almesberger, Jamal Hadi Salim and Alexey Kuznetsov. "Differentiated Services on Linux". *Proceedings of Globecom '99*, vol.1, pp. 831-836, December 1999. Available: http://www.almesberger.net/cv/papers/18270721.pdf

[12] Sally Floyd, Van Jacobson. "Random Early Detection Gateways for Congestion Avoidance", *IEEE/ACM Transactions on Networking*, August 1993.

[13] Paul E. McKenney. "Stochastic fairness queueing". *Interworking: Research and Experience*, Vol.2, January 1991. Available: http://citeseer.ist.psu.edu/mckenney91stochastic.html

[14] A. Demers, S. Keshav, and S. Shenker. "Analysis and simulation of a fair queueing algorithm". In SIGCOMM '89*: Symposium proceedings on Communications architectures & Protocols*, pp. 1-12. ACM Press, 1989. ISBN 0-89791-332-9.

[15] Srinivasan Keshav. "On the efficient implementation of fair queueing". *Journal of Internetworking Research and Experience*, 1991.

[16] Chuck Semeria., *"Supporting Differentiated Service Classes: Queue Scheduling Disciplines"*, Juniper Networks, December 2001.

[17] Sally Floid and Van Jacobson. "Link-sharing and resource management models for packet networks". *IEEE/ACM Transaction on Networking*, 3(4):365-386, 1995. Available: http://citeseer.ist.psu.edu/article/floyd95linksharing.html.

[18] Martin Devera. (2002, May 5). *Hierarchical token bucket theory*. [Online]. Available: http://luxik.cdi.cz/~devik/qos/htb/manual/theory.htm

[19] Ajay Tirumala, Feng Qin, Jon Dugan, Jim Ferguson and Kevin Gibbs. (2003 March) *IPERF: Tool for measuring UDP and TCP bandwidth performance*. [Online]. Available: http://dast.nlanr.net/Projects/Iperf

[20] Stef Coene. *Bandwidth monitoring*. [Online]. Available: http://www.docum.org/docum.org/monitor/

[21] *Linux IMQ: Intermediate Queueing Device*. (2005, August). [Online]. Available: http://www.linuximq.net

[22] *Application Layer Packet Classifier for Linux*. (2005, August). [Online]. Available: http://l7filter.sourceforge.net/

[23] *Official IPP2P homepage*. [Online]. Available: http://www.ipp2p.org

[24] Jha, Sanjay. *Engineering Internet QoS*. Norwood, MA, USA: Artech House, Incorporated, 2002.