QoS in GNU/Linux: its application on free Internet infrastructure

Javier Charne, Diego De La Riva, Hugo Ramón, Adrián Jaszczyszyn

{javier,delariva,hugoramon,adrianjasz}@unnoba.edu.ar
Instituto de Investigación y Transferencia en Tecnología – IITT
Escuela de Tecnología
UNNOBA
Calle Jorge Newbery y Sarmiento – (6000) Junin, Bs As., Argentina

ABSTRACT

With the advent of VoIP transport and videoconferencing/telepresence equipment at relative low costs, companies started to use a single provider for data, telephony and video, thus causing three types of transport –with different characteristics and requirements- to converge in a single hard link. The current converging networks carry packages related to three different kinds of services (data, voice and video) and one of the usual classification criteria draws a distinction between traffic with strict time requirements and all other services. As bandwidth is always limited, it is necessary to identify and give priority treatment to this kind of traffic over others sharing the same medium. This paper discusses the assessment and application of different QoS tools on the free internet infrastructure for public spaces in the city of Junín, Buenos Aires, Argentina.

Keywords: VoIP, Videoconferencing, Quality of Service, Real Time, Links, Free Internet, Linux

1 Introduction

Before the term "converging networks" first appeared in authoritative literature and that they started to be offered by different providers, companies used different media to carry each kind of traffic. Telephony was transferred through PSTN (Public Switched Telephone Network) and another separate service, typically Frame-Relay was hired to transport data.

The video service required a high-cost satellite service. With the advent of the VoIP transport and videoconferencing/telepresence equipment at low costs, companies started to use a single provider for data, telephone and video, thus causing three types of transport –with different characteristics and requirements- to converge in a single hard link.

Under [1], traffic is classified into two main categories: elastic and non-elastic traffic

Elastic traffic can be adjusted to the changes in delays and performance of a given network, without failing to meet its application needs. This is the type of traffic supported by TCP/IP (Transmission Control Protocol/Internet Protocol) networks, the traffic in individual connections gets adapted so that data speed delivery to the network is reduced.

Internet applications, such as file transfer, e-mails, remote connections, network management and web access are elastic traffic applications even though there are some differences in their requirements.

With elastic traffic, having a network service that distinguishes these two kinds of flows (with QoS) is advantageous. Without it, routers manage IP packages blindly, regardless of the kind of application causing or expecting it, or whether the package is part of a small or big transfer. If there is a congestion, it is unlikely that resources will be allocated in a manner that would meet the needs of all the applications. The more non-elastic traffic you add, the less satisfactory the experience.

Non-elastic traffic does not adapt itself to changes in delays and performance of the interconnected networks. Real-time traffic, such as voice and video, is the main example of this situation. The requirements of non-elastic traffic are as follows: minimum ***performance*** value. Many non-elastic applications require minimum consistent performance. Packages should not be ***accelerated*** or ***delayed***. Real-time applications differ in the number of ***lost packages*** they admit.

These are difficult to meet in a congested network, with varying delays in the queue and package loss. Elastic transfer introduces two requirements in the network interconnection architecture. In the first place, some mechanism is needed to give the most demanding applications a preferential treatment, either in advance, with some kind of service request function or on-the-fly, using the header fields in IP packages and allowing the network to anticipate the demand and reject new requests if

the requested resources are not available and a when a congestion is possible. This approach involves the use of some kind of protocol to reserve resources.

Another requirement to support non-elastic traffic on an Internet architecture is that elastic traffic should take into account non-elastic traffic. Whenever there is a congestion, TCP controls flow, UDP does not. It does not reduce its demand and it does not even detect the situation. A protocol should be implemented to reserve resources and control the situation, thus rejecting any new transmission request when there is no bandwidth available for elastic traffic.

2 QoS mechanisms

Different QoS mechanisms together make it possible to implement it inside the IP network and should be carried out in a certain order as if it were an industrial production chain: Classification and marking, policing and shaping, and congestion management.

All pieces of communication equipment have buffers in order to save all the data that enters through an interface in a frame format. These buffers are usually independent memories, one for each interface. And as the processing time of the incoming frames can be longer than the entry rate, they are stored in such memory to be processed later.

Outgoing frames use these buffers in a different way. Those frames that are "ready to go" through the network will be located in the buffers. They will once again be physically independent memories but they will not be allocated to each interface on a one-to-one basis, but rather each physical interface can have several memories, also known as queues.

Classifying traffic will simply mean identifying certain types of flows. Most IOS (Internetwork Operating Systems) [2] which implement QoS mechanisms allow for classifications to be made using the marks in the CoS (class of service) and ToS (type of service) fields. Then any action that does not affect traffic is a mark that can be done in any structure of the operating system or in the CoS or ToS fields.

To manage bandwidth, after having a package marked, the decision is whether such package should be sent, delayed or discarded. *Policing* does not take into account traffic behavior whereas *shaping* does [3].

Congestion Management: Once the packages are classified and marked and a decision as to what to do with them is made, they need to be reprocessed and - in the case of outgoing traffic - be dispatched through the chosen interface, paying special attention at this time not to cause a congestion on the link. An overrun link is one that has reached its physical limit of transmission or of the memory allocated to such interface.

Dispatched packages are taken from the outgoing queues. The selection criterion is known as qdisc (queueing disciplines). It is important to carry out the QoS process on the device where the real queue is located.

3 Junin I/L

At first, the Junín-IL (Free Internte) Project appeared as a simple request: "*Internet is required in the city squares"*. When the working team met for the first time, the following questions were raised:

- Which squares?
- What area should we cover?
- Should we undertake this project ourselves or should we outsource a turn-key solution?
- How do we make the network reach the squares?
- Should we wire through the lightning posts and then set an AP in each square?
- Should we place a wifi router in a nearby building and then wire up from the NOC to the building?
- Which solution is the most scalable one?
- What quality of service do we want the client to experience?
- What will the total outgoing bandwidth be?
- How many recurring clients are we expecting to have?
- Will navigation be free or will we restrict some sites?

The public areas involved in the project were: the main city square, the bus terminal and its adjacent squares and the railway station square (See Fig. 1). Navigation was to be free but the speed at which each user could download data was to be limited, giving priority to web navigation over any other service.

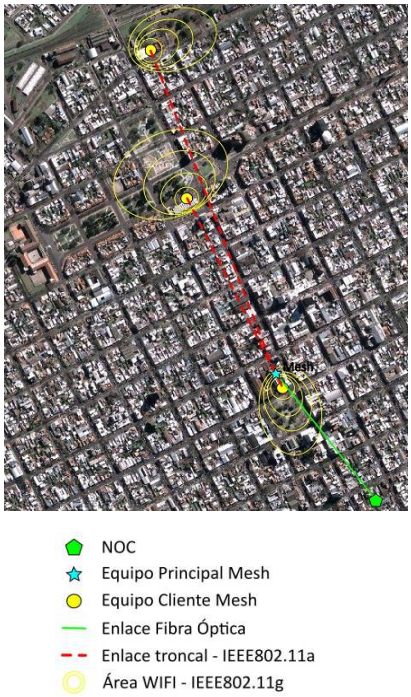

Fig. 1

Several solutions, such as UNLP [16] and San Luis [17] were studied while in production.

We asked for quotations and specifications for the WLC solutions for wireless outdoor Aironet from Cisco [18] [19] [20] and also Wavion [21]. Finally, we decided to set up the installation with Planet Mesh Network MAP-2000 and MAP-2000R [22] both for the wireless network trunk as well as for the Internet distribution for clients. In addition, the optic fiber was laid from the NOC to the main equipment on the mesh and a server was set up to carry out the routing tasks between the LAN and the WAN link, firewall, proxy, DNS, QoS and monitoring.

The wireless equipment communicate through the 5.8 Mhz frequency redistributing the signal at local level at 2.4 Ghz. They were placed considering the area they were supposed to serve. These pieces of equipment were installed in towers of at least 25 meters on each location.

3.1 Implementation Details

Fig. 2 shows the topology. Since the default configuration for the client equipment has a preset range of 10.0.0.0 in its 802.11a interface, such a range was chosen for the links against the central equipment in order to make installation and the initial management of future enhancements easier.

Each client equipment (City Hall, Bus Terminal and UNNOBA University) routes traffic from the clients to the central equipment and a DHCP server is set to distribute to a sub-network of 172.20.0.0/16 among its clients. The NAT options were disabled on every client, so that the IP of the final user would reach the server, where the QoS policies are specified.

The server was set up with the GNU/Linux Debian 6.0 "Squeeze" operating system, where the following packages were set up to provide the required services:

- Domain name system, DNS **bind9**, version 1:9.7.2.dfsg.P3-1.1
- Secure Shell, SSH: **openssh-server**, version 1:5.5p1-6
- Proxy web: **squid3**, version 3.1.6-1.2
- Web server: **apache2**, version 2.2.16-6
- Time server: **ntp**, version 1:4.2.6.p2+dfsg-1+b1
- SNMP service: **snmpd,** version 5.4.3~dfsg-2
- Servidor VPN: **openvpn,** version 2.1.3-2

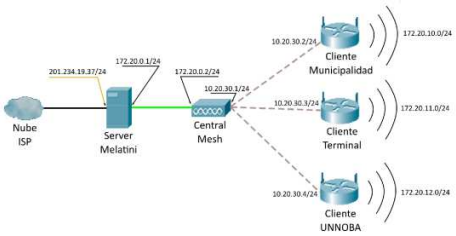

Fig. 2

In addition, we had: **munin,** to monitor server status and **sarg**, to generate proxy use report. Most of the current GNU/Linux distributions have a wide range of network traffic management, control, and monitoring tools in their repositories [4]. This study is based on the GNU/Linux Debian [5] distribution, and we will also discuss in further detail some of the tools that, when used jointly, enable the system to manage traffic efficiently and provide quality services at the Junin/IL project. The tools used are iptables [6], iproute [7], squid [8] and those to monitor heavy traffic such as snmpd, mrtg [9], RRDTool [10], wireshark [11], iptraf [12], somokeping [13], flowscan [14], icinga [15].

3.2 Configurations

To provide the required service quality and restrict bandwidth use per user to 256KBPS, we used Squid proxy delay pools. This allows queues to be set per client (identified by its IP) or per sub-

network. Then the queue is set to allow a maximum bandwidth. When a client reaches this limit, packages are delayed to adjust traffic to the specified rate.

Web traffic coming from the wifi network is redirected so that it can go through the proxy:

```
iptables -t nat -A PREROUTING -i
${LAN_WIFI} -p tcp --dport 80 -j
REDIRECT --to-port 3128
```

There are two important values: The first one sets the rate to which traffic should adapt; the second one sets the initial rate per client. Since web navigation is usually in blasts, it is good for the client to be able to download more data (1MB) initially and then limit to the indicated rate. Limits are specified in bytes, therefore, we specified 32000B for 256Kbps.

There are two queues: an unlimited one (for clients connected through VPN to carry out management tasks) and another one with the required limits. Details concerning the types of delay pools in Squid, preset performance and configuration fall beyond the scope of this study. The proxy configuration is in file /etc/squid3/squid.conf, and part of the delay pools are listed below:

```
acl lan_wifi src 172.20.0.0/16
acl vpn src 192.168.138.0/24
…
delay_pools 2
delay_initial_bucket_level 50
delay_class 1 1
delay_parameters 1 -1/-1
delay_class 2 3
delay_parameters 2 -1/-1 -1/-1 32000/128000
delay_access 1 allow vpn
delay_access 1 deny all
delay_access 2 allow lan_wifi
delay_access 2 deny all
```

Any traffic that does not go through the proxy creates queues (qdisc[24]). In the following script, INT_IF is the internal interface and EXT_IF is the interface connected with the WAN link.

An **htb** queue is set (a hierarchy of classes that can make up traffic at the specified rates) in the internal interface and an **sfq** queue (Stochastic Fairness Queueing) which reorders traffic on the queue so that each session can send a package in turn) for the interface connected to the Internet. Due to performance reasons [27], traffic in the wireless network travelling at more than 20mbit is not desirable.

```
#!/bin/bash
INT_IF=<interfaz con la LAN>
EXT_IF=<interfaz con Internet>
local TCQ="tc qdisc add dev ${INT_IF} "
local TCC="tc class add dev ${INT_IF} "
local TCF="tc filter add dev ${INT_IF} "
local AB="20mbit"
tc qdisc add root dev ${EXT_IF} sfq \
perturb 10
${TCQ} root handle 1: htb default 30 r2q 20
${TCC} parent 1: classid 1:1 htb rate ${AB}\
ceil ${AB} burst 2k
${TCC} parent 1:1 classid 1:10 htb rate 7mbit\
ceil 10mbit burst 2k
${TCC} parent 1:1 classid 1:20 htb rate 1mbit\
ceil 2mbit burst 2k
${TCC} parent 1:1 classid 1:30 htb rate \
512kbit ceil 1mbit burst 2k
${TCQ} parent 1:10 handle 10: sfq perturb 10
${TCQ} parent 1:20 handle 20: sfq perturb 10
${TCQ} parent 1:30 handle 30: sfq perturb 10
${TCF} protocol ip parent 1:0 prio 1 handle 1\
fw flowid 1:10
${TCF} protocol ip parent 1:0 prio 1 handle 2\
fw flowid 1:20
```

To label traffic:

```
iptables -A OUTPUT -t mangle -o ${INT_IF} \
-p tcp --match multiport --sports \
3128, 443, 53,123 -j MARK --set-mark 0x1
iptables -A OUTPUT -t mangle -o ${INT_IF} -p \
tcp --match multiport --sports \
110,143,993,995 -j MARK --set-mark 0x2
```

According to these commands, traffic coming from the Proxy, or belonging to the https, dns or ntp protocols, is labeled with a “1” and sent to the qdisc 1:10, which has 7mbits and a level above 10mbits. Most of this kind of traffic will come from the Proxy.

The ideal thing would be to estimate accurately how much of the web traffic used comes out through the proxy (hit) and how much was not in the proxy and had to be required (miss) based on the monitoring. If the queue is set with the “exact” bandwidth of the wan link, it is guaranteed that there will be no congestion problems on the outgoing interface but the additional bandwidth provided by the proxy when it hit is lost. It such figure is overestimated, the QoS configuration is no longer useful because the packages are in a bottleneck situation and they would not go into the queue of our QoS server but

they will go into the Internet link that is behind the server outgoing interface.

With the second rule of iptables, if traffic comes from pop3 services, imap, pop3s or imap, it is marked with a “2” and placed in the 1:20 queue with a reserved 1mbit.

The remaining traffic goes to the 1:30 queue with 512K and it can climb to 1 mega, when link is idle.

All this aims at preventing clients connected to the wifi network to abuse the service by means of p2p applications since they will only be able to download 1mbit only if no-one else across the whole network is using a similar protocol.

4 Reports

The ***mrtg*** demon collects periodic statistics of the use of links via SNMP (Fig. 3) and builds graphs indicating the total incoming and outgoing traffic, without segregating it by protocol type. Graphs are generated dynamically through a cgi on the web server. These graphs show where the service is being used, the bandwidth consumed and the use peaks, as well as how the traffic on each terminal is added on the mesh central equipment.

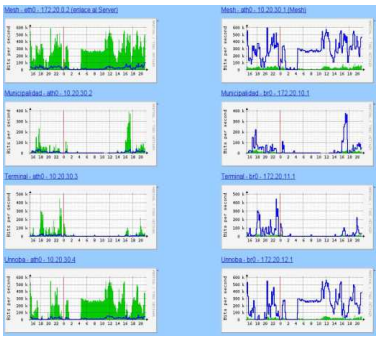

Fig. 3

The set of ***flowscan*** applications (netflow) collect information on the central server concerning the number of flows routed, recording its type and size (Fig. 4 and 5).

When there is a congestion, or there are problems on the links, latency goes up. To monitor this variable, we use ***smokeping***, which sends ICMP packages periodically, recording the reply times. The ***sarg*** tool prepares daily, weekly and monthly reports of the proxy use, recording users (IP), destinations, times and flow sizes. The ***icinga*** (Nagios fork) shows the current state of the links and server and it can send e-mails and sms if it detects that a node is down. The ***munin*** tool delivers information concerning the server's status, such as the following: memory use and server CPU load, number of processes and accesses to the web server, I/O on disc, latency, % of use, temperature, number of connections through the firewall, errors on the interfaces, traffic rate on the network interfaces, connections managed, proxy cache status, number of clients and traffic managed.

5 Conclusions

Our aim was to analyze the problem of the increasing traffic on data networks, the increasing requirements on bandwidth and the congestion and loss of service quality issues; tools to deal with these problems were introduced through the implementation of traffic shaping and traffic policing on GNU/Linux in a real scenario where free Internet access was provided at public areas in the city of Junín, including the main square, the squares near the bus terminal and the square in front of the railway station using free software tools available to implement QoS management.

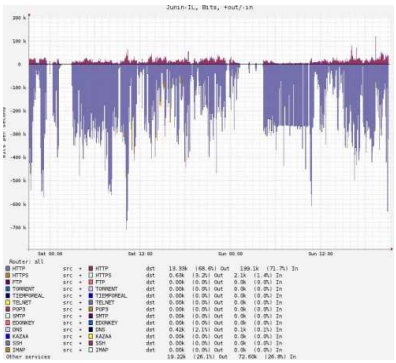

Fig. 4

During this implementation we were able to:

Analyze the issue of increasing traffic in data networks and the need to identify and give priority to a certain kind of traffic.

Study the existing solutions to achieve traffic control and quality of service.

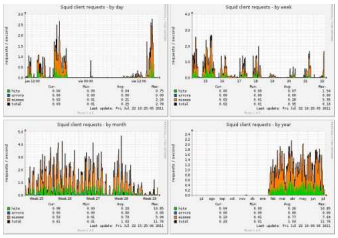

Fig. 5

Analyze the implementation of queues in depth in order to achieve QoS and compare the different classes andvarieties developed.

Analyze the tools provided by GNU/Linux to manage traffic queues.

Study the performance of HTB queues on laboratory and real scenarios.

The tools provided by GNU/Linux are mature, documented and have been widely proved in the past few years.

The initial implementation of this kind of solutions is relatively simple. The real job is to adjust them to meet the objectives for which it was developed. If the outgoing bandwidth is 5 megabytes and the proxy's success/failure rate is around 50%, it is necessary to adjust the preestimated values regarding what the total traffic bandwidth to be consumed by the wireless network should be. Supposing that 50% of the network's request are solved by the proxy, then the QoS queues are set at 10 Mbps. If the success rate falls, this means that out of the 10 Mbps that reach the proxy, more than 5 Mbps of data need to be searched from the Internet. If the link is 5 Mbps, there is a bottleneck and all the queuing infrastructure (placed before the bottleneck) becomes inoperative. Then the bandwidth of the WAN link is extended or the highest limit of traffic is reduced. This is why it is necessary to monitor and adjust configurations on a regular basis in these implementations.

The proxy configuration manages bandwidth per client efficiently and the HTB queues do the same with the rest of the traffic. For a medium level of traffic, the solution discussed above is effective, economical and has been working properly since December 2010.

6 Bibliography

1. Stallings, William. *"Redes e Internet de Alta Velocidad. Rendimiento y Calidad de Servicio"*, (High-speed Networks and Internet. Performace and Quality of Service) pág 16. ISBN: 978-84-205-3921-8, Prentice Hall.
2. Internetwork Operating System, Cisco Systems.
3. "Architecture for Differenciated Services http://tools.ietf.org/rfc/rfc2475.txt
4. Busybox (http://www.busybox.net/about.html),
5. http://www.debian.org
6. http://www.netfilter.org/
7. http://www.linux-foundation.org/en/Net:Iproute2
8. http://www.squid-cache.org/
9. http://oss.oetiker.ch/mrtg/
10. http://oss.oetiker.ch/rrdtool/
11. http://www.wireshark.org/
12. http://iptraf.seul.org/
13. http://oss.oetiker.ch/smokeping/
14. http://www.caida.org/tools/utilities/flowscan/
15. http://www.icinga.org
16. http://riutec.riu.edu.ar/lib/exe/fetch.php?media=wlc-slides.pdf
17. http://www.aui.edu.ar/
18. http://www.cisco.com/en/US/prod/collateral/wireless/ps5679/ps8368/data_sheet_c78-532987.html
19. http://www.cisco.com/en/US/products/ps7221/index.html, con software de control WCS:
20. http://www.cisco.com/en/US/products/ps6305/index.html
21. http://www.wavionnetworks.com/
22. http://www.planet.com.tw/en/product/product_ov.php?id=5774
23. http://wiki.squid-cache.org/Features/DelayPools
24. See tc manual for qdisc at http://linux.die.net/man/8/tc

http://es.wikipedia.org/wiki/IEEE_802.11#802.11a