Volume 56, Number 3, 2015

**ACTA TECHNICA NAPOCENSIS**
*Electronics and Telecommunications*

# IPV6 EXTENSION OF HTB-TOOLS FOR LINUX TRAFFIC CONTROL BASED ON HTB

Doru Gabriel BALAN, Alin Dan POTORAC, Adrian GRAUR
*Stefan cel Mare University of Suceava, 720229, Romania*
*doru.balan@usv.ro*

**Abstract: It is well known that traffic control is a permanent challenge for network engineers. The present paper proposes a novel method for Linux QoS (Quality of Service) solution used in network management. The research is focused on the classful queueing disciplines and specific HTB (Hierarchy Token Bucket) Linux implementations. This paper proposes a new practical improvement of bandwidth management software, named HTB-tools. These tools are generally used for implementing QoS policies based on the HTB algorithm, under a common Linux environment. The proposed improvement of the HTB-tools packages updates the software components with features that allow the implementation of QoS solutions for computer networks running the latest Internet Protocol version, IPv6. Our solution is extending the applicability domain of the HTB-tools QoS instrument to the addressing space of the IPv6, deploying an IPv6-ready rule parser component.**

***Keywords:*** *communication system traffic control, quality of service, queueing analysis, IP networks.*

## I. INTRODUCTION

Since, in recent years the number of network equipment has increased exponentially, the public address space provided by the basic network protocol, IP (Internet Protocol) version 4 [1], was found to be insufficient to cover all connection requests. A first solution to address this problem was the IP translation (NAT - Network Address Translation) [2]. The specific NAT mechanisms provide a mapping of addresses from private [3] to public IP addressing space. This solution is not fully satisfactory, since there are incompatibility problems with the security communication protocol IPSec (Internet Protocol Security) [4] or conflicts of addressing whether the two networks with the same private address space are interconnected via virtual networks (VPN - Virtual private Networks) [5]. On the other hand, the manner in which IPv4 address assignment was carried out led to the existence of a large number of routes in the routing tables of the routers that manage the internet infrastructure.

These drawbacks and the continuous need for a better service quality in data networks have given rise to the transition to a new version of the network protocol, IPv6 [6].

## II. LINUX QOS ELEMENTS

Every existing Linux-based operating system provides a set of tools for managing and manipulating the transmission of packets [8]. The main components of Linux traffic control mechanisms are represented by a series of actions applied to data flows, such as:

- Shaping (delay packets to meet a desired rate);
- Scheduling (organize and/or rearrange packets);
- Classifying (sort or separate traffic into queues);
- Policing (measure and limit traffic in a queue);
- Dropping (discards an entire packet or data flow);
- Marking (modify packet metadata).

In a generic Linux mechanism that is designed to provide quality of service, the data handling process involves three types of objects through which data are processed:

- Schedulers or queueing discipline (qdiscs);
- Traffic classes;
- Traffic filters.

The queueing disciplines that can be used in the data flow scheduler mechanisms are generally classified into two major categories:

- Classless qdiscs (like: [plb]fifo pfifo_fast, RED - Random Early Detection, SFQ - Stochastic Fairness Queueing, TBF - Token Bucket Filter)
- Classful queuing disciplines (such as: CBQ - Class Based Queueing, HFSC - Hierarchical Fair Service Curve, PRIO - priority scheduler, and HTB - Hierarchy Token Bucket.).

### II.A. HTB as a QoS method

The Hierarchy Token Bucket (HTB) [9] is a packet scheduler algorithm that implements a link sharing hierarchy of data traffic classes. HTB facilitates bandwidth allocations to traffic classes, while also allowing specification of upper limits for inter-class sharing. Taking into account its features, the HTB mechanism is generally used to provide shaping functions, based on TBF specific principles [10], identified by the token bucket concept. At the same time, HTB is able to prioritize classes of data traffic in a complex and performant class-based system [11] that allows quality of service specific control over the network traffic.

A powerful characteristic feature of HTB, that lead to many QoS success implementations on Linux based platforms, is related to the fact that since HTB shapes traffic

*Manuscript received September 3, 2015; revised September 8, 2015*

based on the TBF algorithm, it does not depend on network interface characteristics and so is not taking concerns about the underlying bandwidth of the outgoing interface. HTB is used to control the usage of the outbound bandwidth on a given link, using one physical link to simulate several slower links and to send different kinds of traffic on different simulated links [12].

The system administrator role is to specify how to divide the physical link into simulated links and to decide which simulated link to be used for sending a given packet.

In practical implementations, within the Linux based routers/servers, using Linux distributions such as RedHat Linux or Fedora Linux, CentOS and others, HTB can be found as a packet scheduler and is currently included in the latest Linux kernels, under /net/sched/sch_htb.c kernel source tree [13].

### II.B. Linux traffic control instrument

Modern Linux-like operating systems provide traffic control through the *Iproute2* component, which is a collection of utilities for controlling TCP/IP networking parameters and data traffic flows in Linux [14]. Among the kernel included tools found in the *iproute2* package there are two very important instruments:

- ***ip*** command, provides the necessary access to network control and configuration operations for IP packets;
- ***tc*** command, with powerful traffic control features used to achieve QoS implementation at the Linux kernel level.

Since the ***tc*** (traffic control) tool from *iproute2* can be used to show or manipulate traffic control settings in a Linux router environment, this instrument will be used to configure the QoS mechanism that involves those three types of objects through which the data flows are processed: qdiscs (queuing disciplines), classes and filters, as it can be seen in Figure 1 [15].

### II.C. HTB-Tools

There are known several implementations of the HTB algorithm over Linux-based platforms [16, 17].

The quality assurance solutions that are using HTB components cover the full scale of methods for QoS implementation:

- the oldest and most common method, the CLI (Command Line Interface), as it can be used with *Iproute2* components (e.g. *tc*);
- the configuration of a Linux network service (service htb or /etc/init.d/htb), using the text configuration files (e.g. /etc/htb/eth0-qos.cfg, contains the QoS rules for the eth0 network interface, generated by the *HTB-tools*);
- the most recent method, the Web interface.(e.g. *WebHTB* or *T-HTB WEB manager*).

Among all QoS solutions that are based on the HTB algorithm, the most interesting one was considered the one named *HTB-tools*.

The attraction points of this QoS solution are given by the fact that it is easy to implement, it is offering a fast deployment and configuration, and is a low resource consuming application.

The HTB-tools, bandwidth management software package, that can be found online in different versions, [18-20] offers a software suite with several tools that help the system administrator to simplify the complex process of bandwidth allocation. Having the Linux kernel's HTB facility as core structure, it can be used for both upload and download traffic management. The HTB-tools components can generate and check the configuration files and also provide a real time traffic overview for each separate client.

One of the most attractive features of HTB-tools is the fact that the configuration manner is very simple and intuitive, in contrast with the Linux standard traffic control (*tc*) instrument.

The setting procedure of the QoS service based on the HTB-tools is accomplished through the configuration files that can be easily understood and written, having intuitive directives. The simplicity of the configuration modality results from the syntax in which the directives for classifying data traffic and allocating network resources are written.

The initial version of the HTB-tools, as a QoS instrument, allows *src* and *dst* descriptors to identify only IPv4 network addresses as the source and the destination of a data packet, schematically represented in Figure 2. In the following section we will propose a solution to upgrade the QoS instrument with features that will allow IPv6 addresses processing also. Having no IPv6 components, the initial version of the HTB-tools package is unable to process the IPv6 traffic.

This is a major disadvantage because all IPv6 data traffic arrived will not fit in any defined traffic class, so this traffic with no match will be redirected to the *default class,* as it can be seen in Fig. 2. Generically, the default traffic class is a class of traffic that collects the unidentified data traffic and which allocates low QoS resources.

```
[root@router]# tc
Usage: tc [ OPTIONS ] OBJECT { COMMAND | help }
  where:
         OBJECT := { qdisc | class | filter }
         OPTIONS := { -s[tatistics] | -d[etails] | -r[aw] }
```

*Figure 1. Traffic control (tc) help usage.*

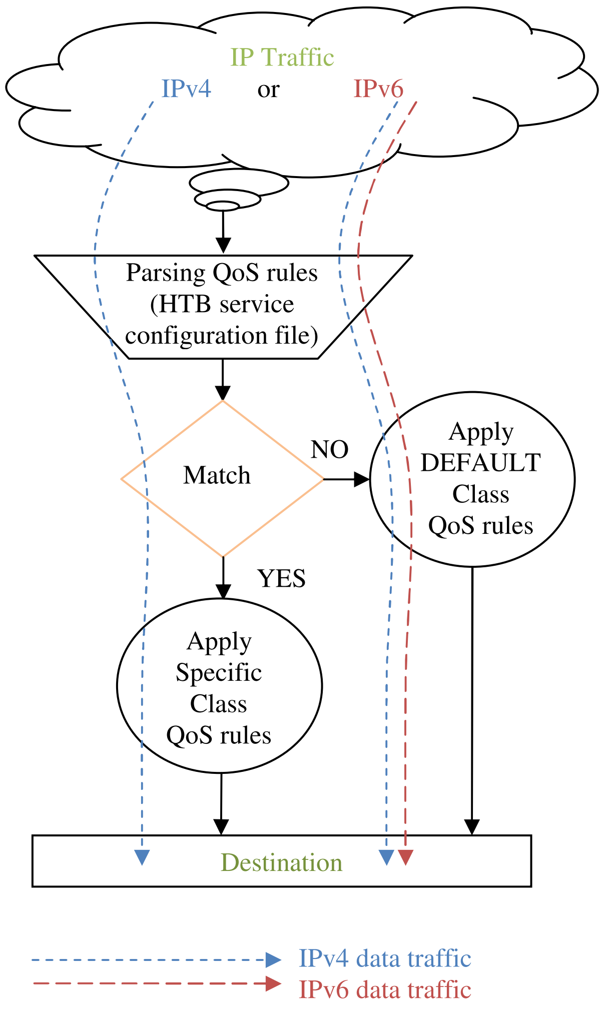

IP Traffic
IPv4
or
IPv6
Parsing QoS rules (HTB service configuration file)
Match
NO
Apply DEFAULT Class QoS rules
YES
Apply Specific Class QoS rules
Destination
IPv4 data traffic
IPv6 data traffic

*Figure 2. Possible traffic paths, without IPv6 QoS component.*

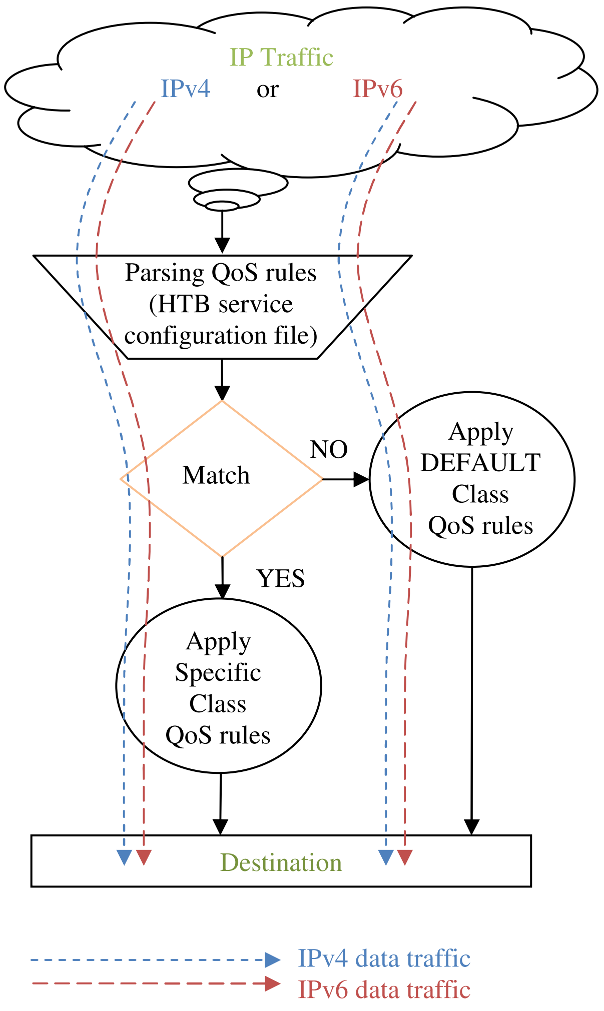

IP Traffic
IPv4
or
IPv6
Parsing QoS rules (HTB service configuration file)
Match
NO
Apply DEFAULT Class QoS rules
YES
Apply Specific Class QoS rules
Destination
IPv4 data traffic
IPv6 data traffic

*Figure 3. Possible data pathway, after IPv6 QoS components inclusion.*

## III. PROPOSED IMPROVEMENTS

The objective pursued in this article is to provide an improved version of the HTB-tools instruments by extending the applicability domain to the IPv6 addressing space.

The contributions found in the following sections aim to solve a practical situation where there is a need to enforce QoS rules within a computer network administered via a common Linux-based router, when an IPv6 network protocol specific addressing space it was introduced in usage, along with the previously used IPv4 addresses.

### III.A. HTB-tools limitations

The major drawback of the HTB-tools is related to the fact that it is limited to the IPv4 addressing space, being developed before the IPv6 mass implementations. Since the current IPv4 address space tends to be depleted due to the increased number of users who need public addresses from this address space, we are proposing an upgraded version of HTB-tool, a version that will be able to process the entire IP addressing space.

### III.B. IPv6 Extension for HTB-tools

As mentioned earlier, the QoS applications found in original HTB-tools package has been developed for the specific network address space of IPv4 network layer protocol. Our research was motivated by the need of simple and performant quality of service instrument for Linux-based routers.

The IPv6 networking addressing scheme is solving the IPv4 deficient addressing space problem. After migrating to IPv6-based networking or in mixed IPv4 and IPv6 networks, there is a permanent need of QoS mechanisms to provide quality communication level to each user or network component.

We are proposing to improve the parsing component of the QoS instrument with IPv6 elements. With such features, the improved QoS mechanism will be able to filter and classify IPv6 traffic, as represented in Figure 3. In this way, the IP data traffic can be completely processed in all running IP versions, and QoS rules can be applied to satisfy the network administration and performance requirements.

```
((([0-9A-Fa-f]{1,4}:){7}(([0-9A-Fa-f]{1,4})|:))|
(([0-9A-Fa-f]{1,4}:){6}(:|((25[0-5]|2[0-4]|[01]?{1,2})(.(25[0-5]|2[0-4]|[01]?{1,2})){3})|(:[0-9A-Fa-f]{1,4})))|
(([0-9A-Fa-f]{1,4}:){5}((:((25[0-5]|2[0-4]|[01]?{1,2})(.(25[0-5]|2[0-4]|[01]?{1,2})){3})?)|((:[0-9A-Fa-f]{1,4}){1,2})))|
(([0-9A-Fa-f]{1,4}:){4}(:[0-9A-Fa-f]{1,4}){0,1}((:((25[0-5]|2[0-4]|[01]?{1,2})(.(25[0-5]|2[0-4]|[01]?{1,2})){3})?)|
((:[0-9A-Fa-f]{1,4}){1,2})))|
(([0-9A-Fa-f]{1,4}:){3}(:[0-9A-Fa-f]{1,4}){0,2}((:((25[0-5]|2[0-4]|[01]?{1,2})(.(25[0-5]|2[0-4]|[01]?{1,2})){3})?)|
((:[0-9A-Fa-f]{1,4}){1,2})))|
(([0-9A-Fa-f]{1,4}:){2}(:[0-9A-Fa-f]{1,4}){0,3}((:((25[0-5]|2[0-4]|[01]?{1,2})(.(25[0-5]|2[0-4]|[01]?{1,2})){3})?)|
((:[0-9A-Fa-f]{1,4}){1,2})))|
(([0-9A-Fa-f]{1,4}:)(:[0-9A-Fa-f]{1,4}){0,4}((:((25[0-5]|2[0-4]|[01]?{1,2})(.(25[0-5]|2[0-4]|[01]?{1,2})){3})?)|
((:[0-9A-Fa-f]{1,4}){1,2})))|
(:(:[0-9A-Fa-f]{1,4}){0,5}((:((25[0-5]|2[0-4]|[01]?{1,2})(.(25[0-5]|2[0-4]|[01]?{1,2})){3})?)|((:[0-9A-Fa-f]{1,4})
{1,2})))|(((25[0-5]|2[0-4]|[01]?{1,2})(.(25[0-5]|2[0-4]|[01]?{1,2})){3})))(%.+)?
```

*Figure 4. IPv6 lexical analysis rule [22].*

Having both IP addressing classes identified the quality assurance mechanism will be able to apply the QoS rules to all IP traffic, including originating from IPv6 sources, or intended to IPv6 network destinations.

The improvement proposes a contribution to the HTB-tools QoS application package by extending the applicability domain of the HTB-tools solution to the addressing space of the IPv6 version of network protocol, which we have identified generic as: the IPv6 Extension of HTB-tools.

### III.C. HTB-tools improvements

The most important contributions to the HTB-tools package can be found in the application that is designed to browse and interpret directives from the configuration file of the HTB service system. During the development process, we have identified and made the required improvements to a HTB-tools component, the instrument called *q_parser*.

We are proposing an upgraded version of the QoS rules parser that allows IP addresses interpretation of the IPv6 addressing space. For this reason have to insert in the source code of the *q_parser* component a directive that defines the traffic control filter for IPv6 addresses, as follows:

*#define U32v6 "U32v6=\"filter add dev $DEV protocol ipv6 parent 1:0 prio 2 u32\"".*

To make IPv6 addressing space usable by the HTB-tools, first we had to include a lexical analysis rule for IPv6, as presented in Figure 4.

The rule is described using the FLEX programing language, which is the specific language dedicated to generating lexical analysis. The lexical analysis rule was included in *parse_cfg* file. The proposed lexical syntax analysis overlaps the specific format of IPv6 addresses (i.e.: 2001: b30: 4c02: 1: 219: 66ff: fe73: ddc5).

From Figures 5 and 6, which contain HTB service configuration items, it can be seen a difference between the previous version of HTB-tools and the proposed version. The improved version adds the IPv6 features to the HTB-tools, providing a more powerful Linux QoS instrument.

```
#################
### eth0-qos.cfg ###
#################
class class_ipv4 {
    bandwidth 1024;
    limit 2048;
    burst 2;
    priority 1;
    que htb;

    client client1 {
      bandwidth 48;
      limit 64;
      burst 2;
      priority 1;
      mark 20;
      dst {
        80.96.120.48/32;
          };
    };
    client client2 {
      bandwidth 148;
      limit 264;
      burst 2;
      priority 3;
      mark 20;
      dst {
        80.96.120.122/24;
          };
    };

}; # end of specifc traffic class

class default { bandwidth 8;
}; #end of the default class
```

*Figure 5. HTB service configuration file with IPv4 rules*

*only.*

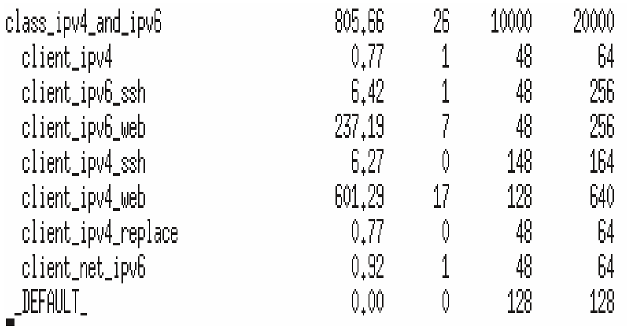
```
#################
### eth0-qos.cfg ###
#################
class class_ipv4_and_ipv6 {
        bandwidth 1024;
        limit 2048;
        burst 2;
        priority 1;
        que htb;

    client client1_ipv4 {
        bandwidth 48;
        limit 64;
        burst 2;
        priority 1;
        mark 20;
        dst {80.96.120.48/32;};
        };

    client client2_ipv4 {
        bandwidth 148;
        limit 264;
        burst 2;
        priority 3;
        mark 20;
        dst {80.96.120.122/24;};
        };

    client client3_ipv6_ssh {
        bandwidth 48;
        limit 64;
        burst 2;
        priority 1;
        mark 20;
        dst {
    2001: b30: 4c02: 1: 219: 66ff: fe73:ed61/128 22;
            };
    };

    client client4_ipv6_web {
        bandwidth 148;
        limit 264;
        burst 2;
        priority 3;
        mark 20;
        dst {
    2001: b30: 4c02: 1: 219: 66ff: fe73:ed61/128 80;
            };
    };}; # end of speciffic traffic class
class default { bandwidth 8;
}; # end of the default class
```

*Figure 6. HTB service sample configuration file with IPv6 components.*

The second major improvement to the HTB-tools is found in *q_parser,* where the rules for IPv6 traffic control were added, to be performed by Linux traffic control (*tc*) utility. The source code has been improved by adding code to process the communications streams between IPv6 addresses. This section also includes traffic control calls for IPv6 traffic prioritization enforcement, creating traffic classes and traffic filtering procedures for IPv6 addresses.

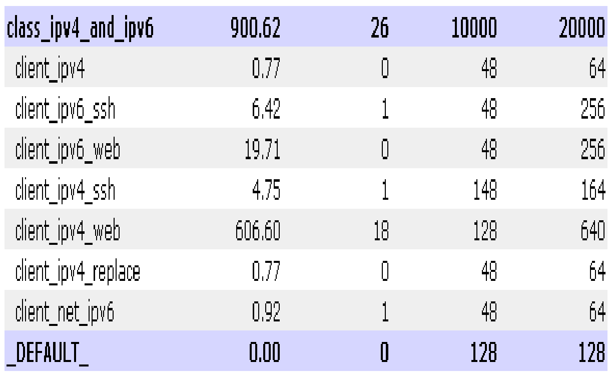
After updating the IP-parsing component (*q_parser*), the QoS directives can be acquired from the configuration files. Having IPv6 elements active in the parser components, in configuration files can be now included rules that have IPv6 components in source and destination descriptors.

HTB-based QoS service monitoring can be done in two ways: from console (using the *q_show* console utility) and from a web page (placing the *q_show* web component under a web accessible structure). Figure 7 and Figure 8 present the interfaces of monitoring utilities for HTB service, from console and the web interface, respectively, that are introduced to demonstrate the functionality of the proposed solution.



```
class_ipv4_and_ipv6          805.66   26   10000   20000
  client_ipv4                  0.77    1      48      64
  client_ipv6_ssh              6.42    1      48     256
  client_ipv6_web            237.19    7      48     256
  client_ipv4_ssh              6.27    0     148     164
  client_ipv4_web            601.29   17     128     640
  client_ipv4_replace          0.77    0      48      64
  client_net_ipv6              0.92    1      48      64
 _DEFAULT_                     0.00    0     128     128
```

*Figure 7. HTB-tools with IPv6 extension console.*



| class_ipv4_and_ipv6 | 900.62 | 26 | 10000 | 20000 |
|---|---|---|---|---|
| client_ipv4 | 0.77 | 0 | 48 | 64 |
| client_ipv6_ssh | 6.42 | 1 | 48 | 256 |
| client_ipv6_web | 19.71 | 0 | 48 | 256 |
| client_ipv4_ssh | 4.75 | 1 | 148 | 164 |
| client_ipv4_web | 606.60 | 18 | 128 | 640 |
| client_ipv4_replace | 0.77 | 0 | 48 | 64 |
| client_net_ipv6 | 0.92 | 1 | 48 | 64 |
| _DEFAULT_ | 0.00 | 0 | 128 | 128 |

*Figure 6. HTB-tools with IPv6 extension web component.*

As it can be seen in the sample configuration file, Fig. 6, or from the monitoring applications, Fig. 7 (console) and Fig. 8 (web interface), the improved QoS tools provide inputs for networking traffic classes with clients (hosts or networks) that have full range of network addresses.

In previous versions of HTB-tools, all connections from hosts or networks based on IPv6 addresses were directed to the *default* traffic class. This was a great inconvenient because in previous versions of HTB-tools it was impossible to identify IPv6 data flows and implicitly it was difficult to allocate QoS resources to those data streams, hosts, networks, or services, the default traffic class having, most of the time, a limited level of associated network resources (less bandwidth and low priority).

## IV. CONCLUSIONS

This article presents the enhancements proposed over the HTB-tools software packet, which is a successful Linux QoS solution. With our contribution, we have expanded the scope of the original applications set to the latest version of network protocol, IPv6. The result is a more powerful tool that can be used to ensure the quality of service to the computer networks, having both, IPv4 and IPv6 resources.

Compared with the previous solutions, the IPv6 Extension of HTB-tools proposed by the authors has the advantages of providing the ability to work with both versions of IP network protocol, IPv6 networking traffic breakdown into traffic classes, clients or services. Thereby, after introducing rules for the processing of IPv6 addresses, the improved QoS instrument can be easily used to associate QoS rules for each network address, defined network spaces, or specified network services, whether they are using IPv4 or IPv6.

The easy administration of the QoS service is a decisive factor in using HTB-tools as QoS tool. Whit HTB-tools service components, the QoS rules management is performed through a very intuitive configuration files, without requiring advanced knowledge of networking, Linux or QoS principles.

Our proposed solution presents a real interest as QoS solution on Linux-based packet routing equipment, because it can be used to manage QoS specific rules over computer networks having both IPv4 and IPv6 protocol stacks available, without involving high cost levels. There are QoS implementations that have already been successfully tested in production environments, academia and private companies, which are using our proposed solution of improving HTB-tools by the IPv6 extension for Linux traffic control based on the HTB packet scheduler.

## ACKNOWLEDGEMENTS

This paper has been financially supported within the project entitled „**SOCERT. Knowledge society, dynamism through research**", contract number POSDRU/159/1.5/S/132406. This project is co-financed by European Social Fund through Sectoral Operational Programme for Human Resources Development 2007-2013. **Investing in people!**

## REFERENCES

[1] Postel, J., "Internet Protocol", RFC 791, September 1981.
[2] P. Srisuresh, K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)", RFC 3022, January. 2001.
[3] Rekhter, Y., Moskowitz, R., Karrenberg, D., Groot, G., and E. Lear, "Address Allocation for Private Internets", RFC 1918, 1996.
[4] Aboba, B. and W. Dixon, "IPsec-Network Address Translation (NAT) Compatibility Requirements", RFC 3715, March 2004.
[5] P. Srisuresh,and B. Ford, "Unintended Consequences of NAT Deployments with Overlapping Address Space", RFC 5684, 2010
[6] S. Deering, R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, 1998.
[7] I. van Beijnum, "IPv6 Internals," The Internet Protocol Journal, vol. 9, no. 3, pp. 16-29, 2006.
[8] M. A. Brown, Traffic Control HOWTO, http://linux-ip.net/articles/Traffic-Control-HOWTO, 2006.
[9] M. Devera, HTB , http://luxik.cdi.cz/~devik/qos/htb/, 2003.
[10] Martin Devera, Hierarchical token bucket theory, http://luxik.cdi.cz/~devik/qos/htb/manual/theory.htm, May 2002.
[11] Ivancic, D.; Hadjina, N.; Basch, D., "Analysis of precision of the HTB packet scheduler," Applied Electromagnetics and Communications, 2005. ICECom 2005. 18th International Conference on , vol., no., pp.1,4, 2005.
[12] M. Devera, Bert Hubert, Linux 2.6 - man page for tc-htb, http://www.unix.com/man-page/linux/8/tc-htb/, 2002.
[13] Linus Torvalds, Linux kernel source tree, git online: https://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/tree/net/sched/sch_htb.c?id=refs/tags/v4.0, 2015.
[14] A. Kuznetsov, S. Hemminger, "NET: Iproute2", http://www.linuxfoundation.org/en/Net:Iproute, 2002.
[15] Martin A. Brown, Traffic Control HOWTO, http://linux-ip.net/articles/Traffic-Control-HOWTO/software.html, 2002.
[16] Balan, D.G.; Potorac, D.A., "Linux HTB queuing discipline implementations," Networked Digital Technologies, 2009. NDT '09. First International Conference on, pp.122-126, 2009.
[17] D. Balan, A. Potorac, "Extended Linux HTB Queuing Discipline Implementations", International Journal of Information Studies, Vol 2, No 2, ISSN 1911-8414, pp. 122-131, 2010.
[18] HTB-tools, http://sourceforge.net/projects/htb-tools.
[19] HTB-tools, http://freecode.com/projects/htb-tools.
[20]HTB-tools, https://aur.archlinux.org/packages/htb-tools.
[21] Bert Hubert, Linux Advanced Routing & Traffic Control HOWTO - The u32 classifier, http://www.tldp.org/HOWTO/Adv-Routing-HOWTO/lartc.adv-filter.u32.html, 2002.
[22] Check IPv4 and IPv6 addresses regular expression http://www.cprogramdevelop.com/1820821/.