AUTOMATIC QOS SYSTEM IN LINUX

Oldřich Plchot
Doctoral Degree Programme (4), FIT BUT
E-mail: iplchot@fit.vutbr.cz

Supervised by: Tomáš Kašpárek
E-mail: kasparek@fit.vutbr.cz

ABSTRACT

This paper deals with the possibilities how to guarantee the quality of service in the area of computer networks using the GNU/Linux operating system. The goal of this work is to discuss the advantages and disadvantages of the tools available in GNU/Linux and to describe the designed system which handles quality of service automatically. Designed system uses a heuristics, which allows the user to set up the QoS without studying specific properties of communication protocols on the network or application layer.

1 INTRODUCTION

We cannot imagine today's world without possibility of easy to use and fast communication, internet and multimedia. The number of small and large computer networks being connected to the internet is still increasing and therefore the need of protecting these networks and ensuring its users' comfortable and effective work is increasing as well. Most of the small and middle networks which are being connected to the internet are dealing with a problem of bottleneck in the capacity of its connection to the international network.

In the area of computer networks, the term Quality of Service (QoS) means ability of the network to achieve required parameters for different types of network applications over the line of a given bandwidth. Different solutions of Quality of Service must deal not only with the simple division of bandwidth, but also with the errors and problems which depends on the actual state of the network, we are unable to influence, and it is impossible to forecast when they happen. In many cases, appropriate setting of the system for quality of service at the access point to the internet can save time and money of connected users.

In this paper we describe a system, which can cooperate with the traditional tools available in the Linux operating system and which can the functionality to automatically adjust the QoS according to the current conditions of the network.

2 EXISTING PROBLEMS WITH QOS

Most of existing solutions dealing with QoS are based on a principle of classification of individual packets and then assigning them to different priority queues. These systems work usually on the network or application layer.

If the classification is done on the network layer, then the only information available for the classification process are the data stored in the packet's header. If we classify according to this data, it is possible to successfully distinguish only limited number of services which have fixed port number. However, some types of applications use random port numbers and in this case it is impossible to correctly classify them, which means that they end up in the same class regardless of an application which needs priority service (like VoIP) or an application which does not need priority service (like P2P download). Somebody can object, that the field Type of Service in the packet header can used, but most applications just ignore the setting of the correct ToS value and on the top of it, this value can be changed by every router the packet passes through and it is never guaranteed that the value really denotes real demand of the service.
Another possibility is an inspection of packet's data on the application layer, when we compare them against some known typical patterns. This approach not only is computationally expensive, but it cannot correctly classify an encrypted traffic. The problem can also arise when some new application or a protocol is created.

3 REALIZING QOS USING CLASSIFICATION OF DATA FLOWS

The result of these facts is that we cannot ensure correct classification of packets and subsequent assignment of priorities, not even by combination of both methods mentioned above. But we can look at this problem from the other side. If we are observing the traffic which is generated by particular applications, we are able to guess into which class does this traffic belongs to, just by observing the bandwidth characteristics it generates.
Every network application creates a connection, which is unambiguously identifiable by the quadruple of data: source and destination IP address and port. Using these data, we are able to class packets into particular flows and therefore we are able to observe the bandwidth characteristics of a data flow they generates. The main information we will need to guess is what type of application belongs to the data flow which has particular function of transfer speed and time. These characteristics can even vary in time, so that interactive application enters the non-interactive mode, which means it will be possible to even re-classify once classified data flow if it changes it's behavior. Another advantage of this approach is, that it works with the encrypted traffic, because we do not need to look inside the packet.
On the figure 1 is depicted a situation when the user is running a graphical application remotely and he is normally working. This is a type of data flow which requires higher priority, so that the user can work interactively and effectively. On the other hand, on the figure 2 is depicted a situation when user starts transmission of a large file after some time of interactive work like browsing directories and typing some commands. When user starts transmitting a large file, the connection does no longer need high priority, because user is waiting for the end of the file transmission. The change of bandwidth of a given data flow is obvious from these two graphs.
As it was shown on the figures we can guess what type of data flow we are dealing with even from the characteristic of one direction of the connection. But we can also use the fact, that in most cases we need to transmit data in both directions, so that we can look at two characteristics simultaneously and therefore increase the amount of information we can obtain. This approach will allow us to classify flows more accurately and into more classes.

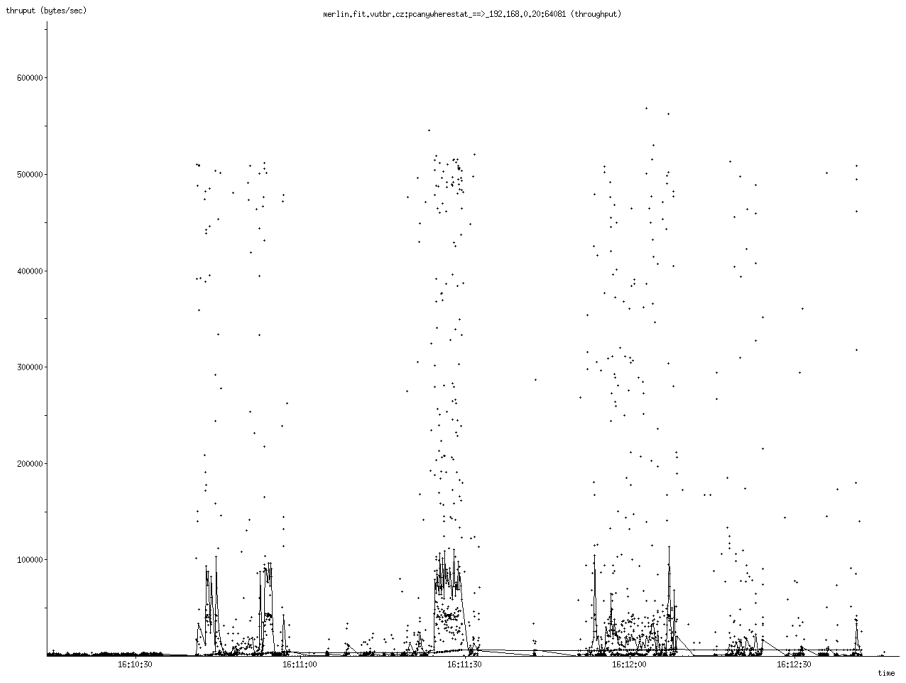

Figure 1: Interactive traffic over ssh

4 DESIGN OF THE SYSTEM

In order to analyze individual data flows on a interface (Ethernet), we have to capture all of the packets passing through this interface. This is the work of a module called *packet sniffer*. We use a multi-plattform library *pcap*, which provides simple and effective interface for capturing packets.

When we obtain packets, we have to analyze them using the data stored in the IP header of each packet. On the basis of these data, we can class the packets into the particular data flows. As an appropriate data structure for storing all the data flows a combination of a hash table and a circular buffer was chosen. We will store particular packets in the circular buffer and the key into the hash table will be created using a data structure unambiguously identifying each data flow.

Another module of the system works with data structure containing individual data flows and it periodically evaluates the transfer speed in time and creates *feature vector* containing information which will be used to classify flows into particular classes representing different types of traffic. Values of this feature vector are normalized into the range $(0, 1)$ according to the given maximum available bandwidth.

We use a neural network (Fast Artificial Neural Network Library - FANN was used) in another module *flow classifier* to classify feature vectors into the different classes. A fully connected feed forward neural network with one hidden layer and bias neurons was chosen. This neural network has thirty input neurons, which is also the size of an input layer, eighty neurons in a hidden layer and 6 output neurons which is also the number of a target classes.

Once we have a data flow classified, the information from the classification and identification of a particular data flow is input for another module, which uses a Linux packet filtering tool *iptables* to set up the correct priority both in the input and output direction. For output direction, the created iptables rule can look like this:

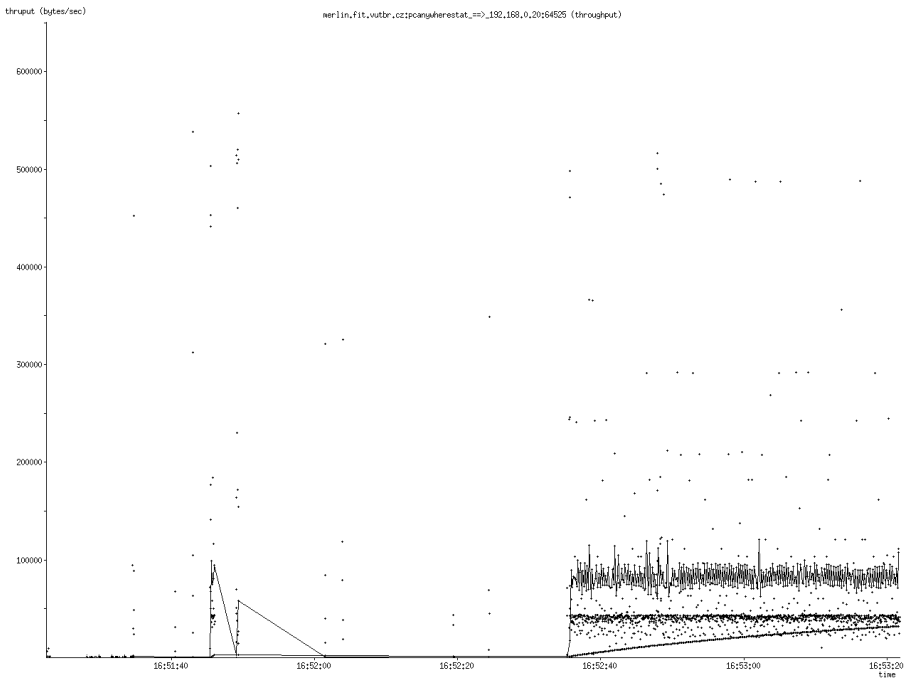

Figure 2: Non-interactive traffic over ssh

```
iptables -t mangle -A MYSHAPER-OUT -p tcp -s 192.168.2.183 \
--sport 3723 -d 209.85.137.83 --dport 443 -j MARK --set-mark 22
```

We can translate this rule into natural language like this: All packets of data flow, originating in our network on a device with IP address 192.168.2.183 using source port number 3723 and ending on remote device with IP address 209.85.137.83 and port 443, are assigned mark 22. This mark will be consequently used to classify the packets into a specific priority queue.

Type of traffic (class)	**Input direction**	**Output direction**
1. Download - FTP, HTTP, sftp	5	3
2. Streaming video, audio	4	3
3. Interactive http, interactive ssh	2	2
4. Connection control	3	3
5. Voice over IP	1	1
6. Upload - FTP, HTTP, sftp	3	5

Table 1: Classification into classes

Each type of traffic needs a different priority in each direction. For example, the download of a large file should have a small priority in the input direction, but we do not have a reason to limit it on the output. Another example can be VoIP which needs a high priority in both directions. The chosen scheme can be found in table 1, where the highest number is the lowest priority and vice versa.

The last part of this QoS system is a script which sets up a scheme of queuing disciplines on the given network interfaces. This script sets up the parameters of a Linux kernel, which will then queue the packets into the priority queues according to the information that was inserted

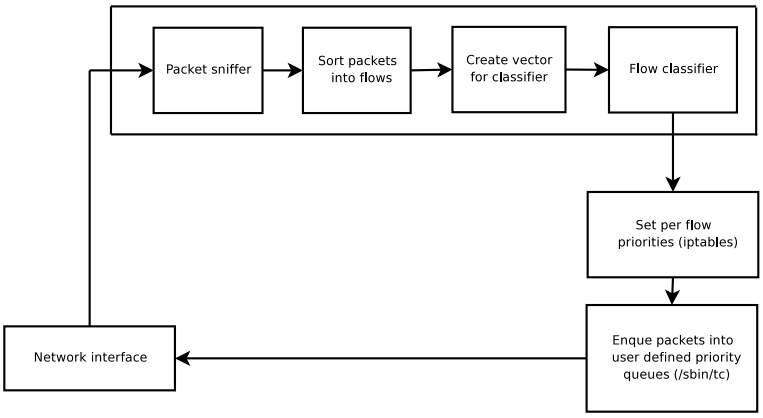

Figure 3: Scheme of a whole QoS system

into the packet in the previous module by iptables. The scheme of the system is shown on the figure 3.

5 CONCLUSIONS AND POSSIBLE FUTURE WORK

The designed system is trying to deal with a problem of ensuring quality of service in a different way than the traditional systems. Instead of difficult and in most cases computationally expensive examining of each packet, it works on top of data flows and computes a simple feature vector for classification. This process is not computationally intensive and works with any application and also with encrypted traffic.
However this system is only a "proof of concept" and is far away to be optimal. This system needs to store a huge amount of data structures which represents individual data flows and actual state of a network traffic on the given network interfaces, thus the system consumes too much memory to be used in production environment.
The future work on this subject could include finding optimal size of data structures used to store the actual state of the network. There can be also done some experiments to find an optimal size of feature vectors and an optimal structure of used neural network. There could be also explored more ambitious plan and try to implement mechanisms into the network layer of operating system in the kernel space, which would provide our feature vectors for the classification in user space.

REFERENCES

[1] Plchot, O.: User Oriented QoS System, Master's thesis, Brno, FIT VUT BRNO, 2007

[2] Stevens, R. W.: Unix Network Programming, Addison Wesley, 2004, ISBN 0-13-141155-1

[3] Brown, M. A.: Traffic Control Howto:
`http://www.linux.com/howtos/Traffic-Control-HOWTO/index.shtml`
October 2006.