BANDWIDTH SHAPING IN COMPUTER AND MOBILE NETWORKS

Fabien Thalgott
2011-08-03
Subject: Software Technology
Level: Master
Course code: 4DV01E
Abstract

In our technological society, the growth of applications is not always followed by the growth of network capacity. There is an increasing demand of bandwidth which cannot always be answered properly. It is especially true in wireless networks such as 3G. Mobile devices can store and use more and more applications which are all using the bandwidth without any control. The Internet experience for the end user can thus be quickly degraded.

In this respect, my objective in this thesis is to fill the gap on this matter by implementing a traffic shaping software working on Android devices. In order to get the necessary knowledge to create such an application, series of tests are conducted on similar software based on Windows 7. The outcome of this research is my own solution to shape the traffic on Android mobile devices. The software allows the end user to choose which of his applications should get a higher priority to improve his overall experience with them.

Keywords: Bandwidth, bandwidth shaping, traffic shaping, bandwidth optimization, mobile device, Smartphone, Android, 3G.
Acknowledgments

I would like to show my gratitude to my supervisor, Ola Flygt, whose patience, support and knowledge allowed me to complete this thesis.

I am also grateful for the support provided by my family and my friends throughout my studies at University.
# Table of content

1 Introduction .......................................................................................................................... 1  
   1.1 Problem description ...................................................................................................... 1  
   1.2 Motivation ..................................................................................................................... 1  
   1.3 Goals .............................................................................................................................. 2  
   1.4 Scientific method .......................................................................................................... 2  
   1.5 Restrictions .................................................................................................................. 2  
   1.6 Report Structure ......................................................................................................... 2  
2 Fundamentals of Quality of Service .................................................................................... 3  
   2.1 Overview of Quality of Service .................................................................................. 3  
   2.2 Parameters ................................................................................................................... 4  
      2.2.1 Throughput ............................................................................................................ 4  
      2.2.2 Burstiness ............................................................................................................. 5  
      2.2.3 Delay .................................................................................................................. 5  
      2.2.4 Delay variation (jitter) ......................................................................................... 5  
      2.2.5 Loss and error rate .............................................................................................. 6  
   2.3 QoS service ................................................................................................................... 6  
      2.3.1 Guaranteed (Quantitative) service ...................................................................... 6  
      2.3.2 Differentiated (Qualitative) service .................................................................... 6  
      2.3.3 Best effort .......................................................................................................... 6  
      2.3.4 Per flow .............................................................................................................. 7  
      2.3.5 Per class ............................................................................................................. 7  
   2.4 QoS mechanism ............................................................................................................ 7  
      2.4.1 Classification ....................................................................................................... 8  
      2.4.2 Scheduling ......................................................................................................... 8  
      2.4.3 Policing .............................................................................................................. 8  
      2.4.4 Reservation ....................................................................................................... 8  
      2.4.5 Admission ........................................................................................................... 9  
      2.4.6 Traffic shaping ................................................................................................. 9  
   2.5 TC (Traffic control) ....................................................................................................... 9  
      2.5.1 Classless Queueing Disciplines ....................................................................... 10  
      2.5.2 Classfull queuing discipline ............................................................................ 12  
3 Mobile devices .................................................................................................................... 13  
   3.1 Usage ............................................................................................................................ 13  
   3.2 Multitasking ................................................................................................................ 14  
   3.3 3G Network specifications .......................................................................................... 15  
4 Practical measurements with existing products .................................................................. 16  
   4.1 Parameters for the tests ............................................................................................... 16  
   4.2 Netlimiter ................................................................................................................... 16  
   4.3 NetBalancer ................................................................................................................ 16  
   4.4 Cfosspeed .................................................................................................................... 17  
   4.5 Traffic Shaper Xp ....................................................................................................... 17  
   4.6 Technical aspects ........................................................................................................ 18  
   4.7 Comparison ................................................................................................................ 18  
   4.8 Interview ..................................................................................................................... 20  
5 Implementation on Android : tcDroid .................................................................................. 21  
   5.1 Overview ..................................................................................................................... 21  
   5.2 Environment ............................................................................................................... 21  
   5.3 Android restrictions ................................................................................................. 21  
   5.4 TC Script ..................................................................................................................... 21
5.5 User Interface ........................................................................................................... 24
6 Implementation evaluation ....................................................................................... 26
   6.1 Downstream ........................................................................................................ 26
   6.2 Upstream ............................................................................................................ 27
   6.3 Conclusion .......................................................................................................... 27
7 Results ..................................................................................................................... 28
8 Conclusion and future work .................................................................................... 29
REFERENCES ............................................................................................................. 30
Appendix : Interview ................................................................................................. 32
1 Introduction

Day by day the internet population is growing. It accelerated these past few years with the rapid development of mobile devices such as PDA (Personal Digital Assistant) and Smartphone. Nowadays there are not many things a computer can do on the internet that a mobile device cannot, but the latter has several restrictions. The CPU (Central Processing Unit) and RAM (Random Access Memory) are lower so applications need to be coded more efficiently in order to maintain a low battery usage. Moreover mobile internet is using 3G connection which is, in most countries, expensive. In the past, Smartphone possibilities were limited so the bandwidth usage was only directed to one application at a time and the data consumption was staying relatively low. However with recent technological improvement and the entrance of new companies on the market such as Google, this pattern is changing. It is now possible to have several applications using the bandwidth at the same time through multitasking and it is used more and more. Usage like VOIP (Voice Over IP) or media streaming on top of that multitasking can make the mobile internet experience unpleasant with lag or even non responsiveness of the connected application. The first purpose of this paper is to study and understand the current usage and behavior of mobile users. Once the results are understood, I will have an overview of the current existing solutions for bandwidth optimization on Windows 7 in order to understand the concepts and principles. In the last part, I propose my own implementation especially designed for Android devices.

1.1 Problem description

The year 2000 is known to be the year when the Internet truly started to grow and develop quickly. With time passing, more and more application are staying connected to the Internet. In the past, this was not a real problem as there were only a few applications having such a behavior and the bandwidth they were using was very limited. However, nowadays the required bandwidth is always increasing and some critical services such as VOIP are not working properly with an overloaded network interface. The conversation will cut and not be understandable by the other party. Some simple actions like web browsing can become very frustrating with pages displaying very slowly. For people playing online game, the latency can become too high and result with a very bad gaming experience and disconnection from the server.

In many cases, the Internet connection is shared between several users within the same house or same building. High latency can arise if some of them start downloading heavily or using peer to peer software such as Bitorrent. For the other users the Internet will become very slow or even unusable. One way of preventing this is to implement a traffic shaping solution to get a fair spread of the available bandwidth between users. This last point is the main objective of this thesis.

1.2 Motivation

Today more than in the past, people are sharing the Internet connection. It can be within a house using the Wifi or even on mobile devices with a 3G access. Not everybody is using its connection the same way so a traffic shaping implementation can be needed to improve the experience of all the users. Personally I am using most of the internet possibilities and often reached the point where I needed to shape my bandwidth in order to keep all my applications usable.
1.3 Goals
This thesis has three main objectives. The first one is to give an understanding to the reader about the importance of traffic shaping on networks today, either applied to a LAN or the Internet. The second objective is to give an overview of the existing host based solutions for Windows 7 and compare them. The last objective is to provide a solution for mobile devices, more specifically Android for which I propose an implementation of a traffic shaper.

1.4 Scientific method
Throughout this research, numerous electronic article databases have been used, supplied by Linnaeus University. Online libraries allowed me to do my researches quickly and efficiently. Appropriate keywords have been used to reach the most relevant literatures. I have browsed several scientific books in network and traffic shaping and academic publications to address my findings. Combining both the library itself and its electronic databases helped me to construct my thesis’ background. Information has been obtained using these data finding process. The data has been sorted in order to gain an insight of the topic. After gathering all these data, I started working on my implementation and during this process; I observed the previous experiences, suggestions and ideas of professionals and created my own solution. I contacted people to ask about their opinions for testing various software that were relevant for my study. I tested my software when my implementation was ready and I did some experiments on my implementation part. After obtaining all these knowledge, I came up with some results including my suggestions and ideas about traffic shaping.

1.5 Restrictions
My work is divided in two parts. The first one focuses on studying the current status of bandwidth shaping with a host based point of view. I decided to look at the most recent operating system from Microsoft: Windows 7. The second part of my work is an implementation of a traffic shaping software on a mobile device. I had the choice to implement my solution on windows mobile 7, iOS 4.0 or Android 2.2. I chose Android mainly because I own such a mobile device so my tests could be easily done.

1.6 Report Structure
First of all, I will give a definition about traffic shaping, reasons to study on it and its aims in chapter 2. Chapter 3 will discuss mobile devices, its usage and recent growth. Chapter 4 will contain my measurements about existing products in the market for bandwidth shaping solutions. I will make proper comparisons and come up with different perspectives about each of them. My implementation, in some detail, will be covered in chapter 5. In the chapter 6, I will run several tests on my implementation and expose my results. Finally in chapter 7 I will discuss my findings and conclusion will be presented in chapter 8.
2 Fundamentals of Quality of Service

In this chapter, background information about QoS (Quality Of Service) fundamentals are being discussed including the overview of the concept, its parameters that needs to be taken into consideration in general, types of services offered by QoS, and QoS mechanisms used to improve the end users’ experience. These points will lead us to the traffic shaping which is the main focus of this study.

2.1 Overview of Quality of Service

As the technology is improving day by day, the development of computer and networking technologies came into importance for computer users. Systems started to advance and it brought us the need of bringing more quality of complex systems in our lives. Similarly, distributed media systems also began to be more complex including variety of services and infrastructure management within. With this respect, both the needs of the users and the demands of such complex systems should be managed in terms of obtaining good functionality that fits the users’ budget. The expectations of users can be met with those systems with quality of service. Quality of service deals with both sides of this situation to create a consensus, combining users’ expectations and requirements with the system's response (Malamos, Malamas, Varvarigou and Ahuja, 1999).

There are several factors to consider finding out what affects the performance of applications (Ganz, Ganz and Wongthavarawat, 2003):

Users: Users have a role in utilizing the applications and it directly affects the performance of those applications in use.

Host machine: These are the machines that enable applications to run and they have variety of components like hard drive, network interfaces and operating systems.

Application: These can be considered as the skeleton of the mechanisms to create multimedia applications.

Network: Network’s role is to transport application contents between source and destination.

It is possible to state that QoS has its way to create a system of «managed unfairness» in networks (Szigeti and Hattingh, 2004). In other words, QoS is the ability of a network component to have some kind of assurance that the traffic and service expectations work together in a satisfactorily harmony (Marchese, 2007).

In technical point of view, Cisco is giving the following definition of QoS : “Quality of Service (QoS) refers to the capability of a network to provide better service to selected network traffic over various technologies, including Frame Relay, Asynchronous Transfer Mode (ATM), Ethernet and 802.1 networks, SONET, and IP-routed networks that may use any or all of these underlying technologies. The primary goal of QoS is to provide priority including dedicated bandwidth, controlled jitter and latency (required by some real-time and interactive traffic), and improved loss characteristics. Also important is making sure that providing priority for one or more flows does not make other flows fail”. Microsoft’s definition for Quality of Service (QoS) is a set of technologies for managing network traffic in a cost effective manner to enhance user experiences for home and enterprise environments (Microsoft, 2011).

The definition of traffic shaping appearing in Network Emulation, Pattern Based Traffic Shaping and KauNET Evaluation is as follow: “Traffic shaping is used to control network traffic to provide Quality of Service, low latency, and increase usable bandwidth by delaying packets. It deals with packet classification, handling of Queues, scheduling of packets for transmission. The basic function of a traffic shaper is to insert delays on packets to ensure guaranteed service for different traffic classes. Queues are used to buffer traffic packets which
are transmitted with fixed delay time depending on the traffic classes to ensure better services utilization.”(Azim, A. Awan, Z.I, 2008).

2.2 Parameters

Today’s end users are dedicated to learn more about the application services rather than considering about the transmission of communication networks (Kusano, Saydam and Yucel, 1998). In this sense, networks should serve the best for the users providing secure, predictable, measurable and guaranteed services. (Cisco, 2011). In this part, the parameters that are related to multimedia applications will be explained.

2.2.1 Throughput

If we think about throughput (also known as bit rate of bandwidth) in an application point of view, it means the rate of data produced by the application, and it needs to be handled properly, because it is the network resource for the running applications. Throughput is based on the application's properties. For instance, an end user who wishes to launch a video streaming application, he or she can do some modifications on the quality of the video by using its different characteristics, explained below (Ganz, Ganz and Wongthavarawat,2003):

**Frame size:** It refers to the number of pixels in rows and columns of the screen.

**Frame rate:** Video frame rate is proportional to the bandwidth consumption, which means if one decreases the video frame rate, consumption decreases too, however the quality of the video (movement, and smoothness) will be affected badly.

**Colour depth:** It is the number of colours possible to obtained by a pixel.

**Compression:** If the bandwidth consumption is decreased, image quality changes respectively.

The popular website Youtube is hosting videos viewable in different qualities, 360p, 480p, 720p and 1080p. The tables on figure 2.1 and 2.2 are showing which encoding has been used and their respective bitrate. As seen before, the higher the resolution (and the quality), the higher the bitrate (McFarland, P., 2010).

<table>
<thead>
<tr>
<th>Format</th>
<th>Video Codec</th>
<th>Audio Codec</th>
<th>Container</th>
</tr>
</thead>
<tbody>
<tr>
<td>37</td>
<td>H.264 1920x1080 24/30 fps</td>
<td>AAC 44.1kHz Stereo</td>
<td>mp4</td>
</tr>
<tr>
<td>22</td>
<td>H.264 1280x720 24/30 fps</td>
<td>AAC 44.1kHz Stereo</td>
<td>mp4</td>
</tr>
<tr>
<td>35</td>
<td>H.264 854x480 24/30 fps</td>
<td>AAC 44.1kHz Stereo</td>
<td>flv</td>
</tr>
<tr>
<td>34</td>
<td>H.264 640x480 24/30 fps</td>
<td>AAC 44.1kHz Stereo</td>
<td>flv</td>
</tr>
<tr>
<td>18</td>
<td>H.264, 480x360 24/30 fps</td>
<td>AAC 44.1kHz Stereo</td>
<td>mp4</td>
</tr>
<tr>
<td>5</td>
<td>Sorenson Spark, 320x240 24/30 fps</td>
<td>MP3 22khz Stereo</td>
<td>flv</td>
</tr>
<tr>
<td>17</td>
<td>MPEG-4 ASP, 12 fps, black bordered to fit 176x144 frame</td>
<td>AAC 22khz Mono</td>
<td>mp4</td>
</tr>
<tr>
<td>13</td>
<td>H.263+, 15 fps, stretched to full frame 176x144 ignoring source aspect ratio</td>
<td>AMR 8kHz Mono</td>
<td>3gp</td>
</tr>
</tbody>
</table>

Figure 2.1 : Video and audio codec used by Youtube (McFarland, P., 2010)
2.2.2 Burstiness

Burstiness is the degree of bit rate variability of an application (Ganz, Ganz and Wongthavarawat, 2003). It is calculated by Mean Bit Rate (maximum number of bits in short time schedule) and Peak Bit Rate (average number of bits in long time).

2.2.3 Delay

Delay is the waiting time for the destination application to receive the packet sent by the source application (Xiao, 2008). Applications require to be delivered to its desired destinations through some period of time. During this process, delays cause problems and reduces the quality of the output. Delay can be grouped as follows (Ganz, Ganz and Wongthavarawat, 2003):

**Source-processing delay**: This delay is dependent on the source's hardware configuration, and its current number of running applications and required resources.

**Transmission delay**: Packet size and the transmission speed is the main concern to calculate the transmission time.

**Network delay**: Network delay can be described as the estimated time required for the first bit of the packet put in the wire and the time that the last bit of the packet is sent to the desired destination (RFC2679).

2.2.4 Delay variation (jitter)

All the packets have a delay value. Delay variation of a packet refers to the difference between the delay inside of the packet itself and the delay of the desired packet. Delay variation can be caused by the following points (Xiao, 2008):

- Different packets having different queueing delays in the same network component.
- Different packets having different processing delays in the same network component.
Different packets transmitted through different network destinations and obtaining different queueing and propagation delays.

2.2.5 Loss and error rate
Packet loss rate is the certain rate of the packet that is lost, and the packet error rate are the errors which were already in a packet because of some of corrupted bits. Both of them should be in minimum amount, because it is directly related to the obtained quality of the application. Consequently, the content of the data changes and the service becomes out of order. The reasons can be the network congestion and noisy communication channels. (Balasubramanian, 2006) and (Ganz, Ganz and Wongthavarawat, 2003).

2.3 QoS service
Applications are giving their QoS requirements to the network, through quantitative or qualitative expression. The network will then answer these requirements by supplying QoS services using a number of QoS mechanisms. There are two categories of services as follow:
- The kind of service provided by the application: qualitative, quantitative or best effort service
- The recipient to which the network provides the service: per-flow or per-class QoS service

A network may use a combination of the QoS services stated above. For example, Guaranteed integrated services (IntServ) is providing a per-flow and quantitative QoS Service. Another known example, Differentiated Services (DiffServ) is providing per-class and qualitative QoS service. The next subsections are giving an overview of each of QoS services stated before (Ganz, Ganz and Wongthavarawat, 2003).

2.3.1 Guaranteed (Quantitative) service
Guaranteed Service, also known as hard QoS, is ensuring the application gets its quantitative requirements in term of network performances (bandwidth, delay, jitter). With this service in use, the application will get network performances which will not go below an attributed value (Ganz, Ganz and Wongthavarawat, 2003). This service is especially beneficial for multimedia application (video and audio) as well as interactive voice application (Szigeti,T., Hattingh, C. 2004)

2.3.2 Differentiated (Qualitative) service
Differentiated service, also known as soft QoS, is providing relative services. This means that one class of application will be better served in term of performances than another class. This can be done by implementing class priority where the highest priority will be served before lower priority (Ganz, Ganz and Wongthavarawat, 2003). This service does not give any guarantee about bandwidth or delay, it only differentiates the traffic in order to apply specific treatment to each class. This service is adapted for bandwidth-intensive data applications so other concurrent application can still obtain a basic network connectivity at all time (Szigeti,T., Hattingh, C. 2004).

2.3.3 Best effort
Best effort service is giving an equal treatment to all traffic. It is how a basic Internet connection is working nowadays (Szigeti,T., Hattingh, C. 2004). This service works well
with applications which do not require any minimum bandwidth or time delivery such as FTP, Email or web pages (Ganz, Ganz and Wongthavarawat, 2003).

2.3.4 Per flow
Per-flow QoS services are ensuring minimum performances to individual flows (applications) quantitatively or qualitatively. An interactive video has stricter delay requirements than streaming video so each application needs to be treated separately in order to meet their individual needs (Ganz, Ganz and Wongthavarawat, 2003).

2.3.5 Per class
Per-class QoS services are providing different performances for each individual classes. These classes are populated by applications sharing similar criteria (QoS requirement, application types, and protocol families). Applications within the same class will have the same treatment (Ganz, Ganz and Wongthavarawat, 2003).

2.4 QoS mechanism
Qos mechanisms can be divided into two groups according to how the application traffic flows (Ganz, Ganz and Wongthavarawat, 2003). The figure 2.3 illustrates the following mechanisms:
Traffic handling mechanisms (also known as in-traffic mechanisms): These are the mechanisms that are responsible for classification, handling and monitoring for the network traffic. These mechanisms can be classified as follows: classification, channel access, packet scheduling and traffic policing.
Bandwidth management mechanisms (also known as out-of-traffic mechanisms): These mechanisms’ role is to manage the network resources. They are responsible for the coordination and the configuration of network devices. These mechanisms include: resource reservation signaling and admission control.

![Figure 2.3: Qos mechanisms (Ganz, Ganz and Wongthavarawat, 2003)](image)
2.4.1 Classification
In order to provide a specific QoS for a single or a group of applications it is necessary to be able to identify and separate their data flow. Classification has this specific purpose and relies on several mechanisms working at different levels of the OSI model as shown in figure 2.4. Classification mechanism can be implemented on most network devices and can apply per-user, per-flow, or per-class classification according to what is needed. The separated data flow is then sent into a specific queue where the packet scheduler will decide which treatment to apply to it (Ganz, Ganz and Wongthavarawat, 2003).

<table>
<thead>
<tr>
<th>OSI Layer</th>
<th>Classification Techniques</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>User/Application Identification</td>
</tr>
<tr>
<td>Transport</td>
<td>Flow (5-tuple IP Address)</td>
</tr>
<tr>
<td>Network</td>
<td>IPTOS, DSCP</td>
</tr>
<tr>
<td>Data Link</td>
<td>802.1p/Q Classification</td>
</tr>
<tr>
<td>Physical Layer</td>
<td></td>
</tr>
</tbody>
</table>

Figure 2.4 : Examples of existing classification on each OSI layer (Ganz, Ganz and Wongthavarawat, 2003)

2.4.2 Scheduling
Traffic scheduling is used to provide QoS by handling how packet queues are emptied. By using packet scheduling algorithms such as FIFO (First In First Out) or Weight Fair Queue, the system decides which queue and host are scheduled for transmission in a certain time. Packet scheduling controls bandwidth allocation to host, classes and applications. There are two level of packet scheduling (Ganz, Ganz and Wongthavarawat, 2003):
- Inrastation packet scheduling which retrieves a packet from a queue of the same host
- Interstaton packet scheduling which retrieves a packet from a queue from different hosts.

2.4.3 Policing
Policing is making sure that all traffic that passes through is conforming to defined traffic parameters. If one of the parameter is violated, a mechanism will be activated to shape the traffic (Ganz, Ganz and Wongthavarawat, 2003). For example if an application is using more bandwidth than allowed, the policing will start dropping packets to reach its maximum allowed level. Policing is usually applied when packet loss is not a critical criterion (Azim, A. Awan Z.I., 2008).

2.4.4 Reservation
The QoS mechanisms discussed previously are enabling QoS services for a device. However to obtain a proper end-to-end QoS, all the network devices along the path of the data flow need to cooperate with each other. Reservation is making sure that it is the case by sending
the performance requirements to each of the devices. The devices will then manage their resources to accommodate these requirements and once the path has been set up, the actual transmission can start. RSVP (Resource Reservation Protocol) is the most known resource reservation signaling protocol (Ganz, Ganz and Wongthavarawat, 2003).

2.4.5 Admission
This mechanism is the one responsible to accept or not new network session according to the resources available. If the new session degrades the performances of the already existing sessions, it will be either rejected or it will notify the user that QoS cannot be provided anymore. Reservation and Admission mechanisms are working closely together and are implemented on the same network device (Ganz, Ganz and Wongthavarawat, 2003).

2.4.6 Traffic shaping
Traffic shaping is used to obtain a low latency and increase the usable bandwidth by delaying packets. To do so, it is using most of the QoS mechanisms discussed before. It is using classification to separate specific data flows, it then sends these packets to a queue which will act as a buffer and the scheduler will transmit the data according to the QoS settings (Azim, A. Awan 2008).

2.5 TC (Traffic control)
Iproute2 is a set of utilities designed to control TCP and UDP IP traffic in Linux. These utilities are working for both IPv4 and IPv6 and are intended to replace an entire suite of legacy Unix networking tools (often called “net-tools”) that were previously used for the tasks of configuring network interfaces, routing tables, and managing the ARP table, but which have not been updated since 2001. The table 2.1 shows which tools were replaced with their equivalent.

<table>
<thead>
<tr>
<th>Purpose</th>
<th>legacy &quot;net-tools&quot;</th>
<th>iproute2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address and link configuration</td>
<td>ifconfig</td>
<td>ip addr, ip link</td>
</tr>
<tr>
<td>Routing tables</td>
<td>route</td>
<td>ip route</td>
</tr>
<tr>
<td>Neighbors</td>
<td>arp</td>
<td>ip neigh</td>
</tr>
<tr>
<td>Tunnels</td>
<td>iptunnel</td>
<td>ip tunnel</td>
</tr>
<tr>
<td>Multicast</td>
<td>ipmaddr</td>
<td>ip maddr</td>
</tr>
<tr>
<td>Statistics</td>
<td>netstat</td>
<td>ss</td>
</tr>
</tbody>
</table>

Table 2.1 : Tools replaced by iproute2

iproute2 unifies the syntax for these various commands, which evolved over many years of Unix development. The iproute2 syntax is much simpler and more consistent for all of the functions that it provides, and imitates the syntax of Cisco's IOS operating system.

TC is a binary contained in iproute2. Its goal is to control the traffic on a given interface by applying queues and filters. There are two types of queues: classless and classfull queuing disciplines, each of them are able to use a set of shaping algorithms (Hubert, B. et al, 2002).
2.5.1 Classless Queueing Disciplines

A classless queuing discipline accepts data and only reschedules delays or drops it. This class had three different disciplines: pfifo_fast, TBF, SFQ.

→ pfifo_fast

This discipline is the default one. It uses three bands on which the rule First In, First Out is applied. Pfifo_fast is sorting the packets according to their TOS field value and filling the bands with them. As long as there are packets in band 0, band 1 will not be processed and as long as band 1 is not empty, band 2 will not be processed. The TOS (Type Of Service) field is contained in the header of each packet. It is coded with 4 bits and has the following meaning:

<table>
<thead>
<tr>
<th>Binary</th>
<th>Decimal</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>8</td>
<td>Minimize delay (md)</td>
</tr>
<tr>
<td>0100</td>
<td>4</td>
<td>Maximize throughput (mt)</td>
</tr>
<tr>
<td>0010</td>
<td>2</td>
<td>Maximize reliability (mr)</td>
</tr>
<tr>
<td>0001</td>
<td>1</td>
<td>Minimize monetary cost (mmc)</td>
</tr>
<tr>
<td>0000</td>
<td>0</td>
<td>Normal Service</td>
</tr>
</tbody>
</table>

Using this field, tc has a predetermined priority map and directs the packets in one of the three band according to the value of the TOS field. The map used is the following:

<table>
<thead>
<tr>
<th>TOS</th>
<th>Bits</th>
<th>Means</th>
<th>Linux Priority</th>
<th>Band</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0</td>
<td>0</td>
<td>Normal Service</td>
<td>0 Best Effort</td>
<td>1</td>
</tr>
<tr>
<td>0x2</td>
<td>1</td>
<td>Minimize Monetary Cost</td>
<td>1 Filler</td>
<td>2</td>
</tr>
<tr>
<td>0x4</td>
<td>2</td>
<td>Maximize Reliability</td>
<td>0 Best Effort</td>
<td>1</td>
</tr>
<tr>
<td>0x6</td>
<td>3</td>
<td>mmc+mr</td>
<td>0 Best Effort</td>
<td>1</td>
</tr>
<tr>
<td>0x8</td>
<td>4</td>
<td>Maximize Throughput</td>
<td>2 Bulk</td>
<td>2</td>
</tr>
<tr>
<td>0xa</td>
<td>5</td>
<td>mmc+mt</td>
<td>2 Bulk</td>
<td>2</td>
</tr>
<tr>
<td>0xc</td>
<td>6</td>
<td>mr+mt</td>
<td>2 Bulk</td>
<td>2</td>
</tr>
<tr>
<td>0xe</td>
<td>7</td>
<td>mmc+mr+mt</td>
<td>2 Bulk</td>
<td>2</td>
</tr>
<tr>
<td>0x10</td>
<td>8</td>
<td>Minimize Delay</td>
<td>6 Interactive</td>
<td>0</td>
</tr>
<tr>
<td>0x12</td>
<td>9</td>
<td>mmc+md</td>
<td>6 Interactive</td>
<td>0</td>
</tr>
<tr>
<td>0x14</td>
<td>10</td>
<td>mr+md</td>
<td>6 Interactive</td>
<td>0</td>
</tr>
<tr>
<td>0x16</td>
<td>11</td>
<td>mmc+mr+md</td>
<td>6 Interactive</td>
<td>0</td>
</tr>
<tr>
<td>0x18</td>
<td>12</td>
<td>mt+md</td>
<td>4 Int. Bulk</td>
<td>1</td>
</tr>
<tr>
<td>0x1a</td>
<td>13</td>
<td>mmc+mt+md</td>
<td>4 Int. Bulk</td>
<td>1</td>
</tr>
<tr>
<td>0x1c</td>
<td>14</td>
<td>mr+mt+md</td>
<td>4 Int. Bulk</td>
<td>1</td>
</tr>
<tr>
<td>0x1e</td>
<td>15</td>
<td>mmc+mr+mt+md</td>
<td>4 Int. Bulk</td>
<td>1</td>
</tr>
</tbody>
</table>

Some applications are setting their TOS field by default; the following table is showing examples from the RFC 1349.

<table>
<thead>
<tr>
<th>Application</th>
<th>TOS</th>
<th>Band</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>TELNET</td>
<td>1000</td>
<td>(minimize delay)</td>
<td></td>
</tr>
<tr>
<td>FTP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Control</td>
<td>1000</td>
<td>(minimize delay)</td>
<td></td>
</tr>
<tr>
<td>Data</td>
<td>0100</td>
<td>(maximize throughput)</td>
<td></td>
</tr>
<tr>
<td>NNTP</td>
<td>0001</td>
<td>(minimize monetary cost)</td>
<td></td>
</tr>
</tbody>
</table>

This discipline is limited to TOS field and is not customizable.
Token Bucket Filter (TBQ)
This discipline is allowing packets to pass at a set rate and is allowing short bursts exceeding this rate. This qdisc should be the default choice when the aim of the user is only to slow down the speed of an interface. The implementation consists of a buffer (called bucket) which is constantly filled with information called tokens, at a specific rate (token rate). The most important parameter for this qdisc is the size of the bucket as it will decide how many tokens can be stored. Each arriving token collects one incoming data packet from the data queue and is then deleted from the bucket. This method shows two flows, data and token, which are leading to three different scenarios:
- The data arrives in the bucket at the equal rate of incoming token. In that case each packet has its matching token and passes the queue without delay.
- The data arrives at a rate smaller than the token one. Only a part of the tokens are deleted at output of each data packet that's sent out the queue, so the tokens accumulate, up to the bucket size. The unused tokens can then be used to send data at a speed that's exceeding the standard token rate, in case short data bursts occur.
- The data arrives in TBF at a rate bigger than the token rate. This means that the bucket will soon be devoid of tokens, which causes the TBF to throttle itself for a while. This is called an 'overlimit situation'. If packets keep coming in, packets will start to get dropped.

The last scenario is the most important as it is in that case that the shaping is occurring. The accumulation of tokens allows a short burst of overlimit data to be still passed without loss, but any lasting overload will cause packets to be constantly delayed, and then dropped. Finally in the actual implementation tokens correspond to bytes and not packets.

Stochastic Fairness Queuing (SFQ)
This discipline is an implementation of the fair queuing algorithms family. SFQ is based on flows which correspond to TCP session or UDP stream. The traffic is divided into several FIFO queue, one for each flow. The traffic is then sent in a round robin way in order to give each flow a chance to send the data in turn. This configuration creates fairness between the flows and prevents a single flow from drowning out the rest. SFQ is not really creating a queue for each flow, instead it has an hashing algorithm which divide traffic into a limited number or queues. The problem with this hashing algorithm is that it can send multiple sessions in the same bucket which is reducing their chances to send a packet. To avoid this, SFQ is changing its hashing algorithm often (how often can be set by the user) so collision occurs only for a limited amount of time. It is important to note that SFQ is only useful when the interface is really full. If not, this qdisc will not have any effect. In general this qdisc is used with other shaping classes, more information are given later in this chapter.

When to use which queue
To summarize, the following is recommended to obtain the best results:
- In order to only slow down outgoing traffic it is advised to use TBF as long as the bucket is scaling the bandwidth.
- When the network is saturated, use SFQ to ensure that no single connection is taking away all the resources.
- Any rule set for the incoming traffic should be done with the Ingress policer which will limit the bandwidth. In that case we speak about policing.
2.5.2 Classfull queuing discipline

Classfull queuing discipline, unlike classless ones are working with a hierarchical structure. Each class can have subclasses and these subclasses can have further subclasses. A class can have another class or a qdisc as a parent, a class without any child is called a leaf class. The default behaviour is to attach a FIFO qdisc to every class but other queuing discipline can be used instead. Filters are used in order to guide each packet to the class it belongs and apply its own treatment. Classfull qdisc are especially adapted to handle different kinds of traffic and apply rules through scheduling, policing and traffic shaping. For every network interface, an egress root queue disc is attached. It is also possible to use an ingress qdisc to deal with incoming traffic using policing.

➔ PRIO qdisc

PRIO qdisc is not really shaping the traffic. It creates three classes and applies different priority to each of them. Class 1 will always have the highest priority, the two other classes will only be used if there is no packet to send in class 1. In each of these classes, a FIFO behaviour is applied by default. Other classes can be used on each of these classes created by PRIO in order to get some shaping. If there is enough bandwidth to transmit the data, it will be sent with no delay.

➔ CBQ

CBQ stands for Class Based Queuing. It is the most famous queuing discipline but also the most complex. In order to do the shaping, CBQ is calculating the idle time between average packet. The problem is that idle time value is difficult to pinpoint as it depends on the dequeue time and the bandwidth of the link. Other factors like badly implemented network drivers or not common protocols also have an impact on the calculation. As a result CBQ can sometimes be unreliable and a very precise configuration is necessary to make it work properly. It also acts like a PRIO queue in a sense that it is possible to set different priorities between the classes and the lower priority numbers will be processed first. A Weighted Round Robin process (WRR) is applied to all traffic classes starting from high priority to lower priority classes. If two classes have the same priority, they are grouped together and dequeued one after another.

➔ HTB

Hierarchical Token Bucket (HTB) works like CBQ but instead of basing the shaping on idle time calculation, it uses a classful Token Bucket Filter. With CBQ, we allocate an amount of bandwidth to a class, if more bandwidth is available, it will no be used. On the other hand, with HTB it is possible for classes to borrow bandwidth to other classes when possible. Higher priority classes can borrow the available bandwidth from lower priority classes.
3 Mobile devices

As more and more people are using PDA nowadays, we should think about bandwidth optimization in mobile devices too. However, there are no current solutions dedicated to this matter. Existing products are just monitoring the network traffic and showing the results to the users. In this part I will discuss about different aspects of mobile devices usage and specifications.

3.1 Usage

Allot Communications Ltd. made a research about bandwidth usage in mobile devices and their report, indicates significant growth in mobile bandwidth usage during the last six months of 2010. The figure 3.1 shows a datagram classifying the growth of bandwidth usage by application type. The figure 3.2 is cutting down the global bandwidth usage by application type (Allot Communications Ltd, 2011). This shows that more and more customers are using their device in at least four tasks: Video streaming, web browsing, file sharing and Instant Messaging (IM). These tasks can be done while multitasking (e.g. preloading a video stream while browsing the web) and will lead to a handset having a poor response time when it comes to applications connected to the Internet.

![Figure 3.1: Growth in mobile data usage for the last semester 2010 (Allot Communications Ltd, 2011)](image-url)
Additionally a study has been conducted by AKQA and dotMobi in May 2008 to see and understand the current needs, usage and wish of mobile internet users. Out of this study, it appeared that one of the main reason users are not inclined to connect the internet is the price and decreasing it would attract an even larger group of customers. The study also shows that the main factor for unsatisfied customer was a slow connection speed. The documents also reported the most common usage of the 3G connection: checking information (weather, sports…), sending/receiving emails, using maps, browsing the internet, downloading/listening to music (dotMobi & AKQA, May 2008).

3.2 Multitasking

The multitasking topic is recurrent in the Smartphone area. This feature is omnipresent on computer where the system resources are important; however mobile devices need to deal with several constraints. Firstly, the size of the screen is limiting the possibilities of doing several things at a time. Second, the battery life needs to be able to stand at least a full day. Third, the system resources, the CPU is slower to save battery life and the amount of memory is low compared to a desktop. It is that last constraint which is actually limiting the possibilities of multitasking on Smartphone. For a long time Apple stated that it was unnecessary and incorporated in the version 4.0 of its iOS with some limitation. When an application is running in the background, parts of its functionality are frozen and others can still be active. This behavior is dictated by how the developer chose to implement his application. The current position of Microsoft is against with windows mobile 7, their last mobile operating system. When an application is sent to the background, its state is saved in memory and becomes frozen. It will be reactivated only when brought back to the foreground. On the other hand WebOS (from Palm) and Android (Google) have a different approach and allow application to run on the background. For Android, when the user is switching to another application the process stays open as long as it can. Once the system runs low on memory, the state of the application is saved and the process is killed so that when it is brought back to the foreground again, it can resume the activity like it has never been killed.
In the past years, the usage of 3G connection was aimed at one task at a time like browsing the internet but nowadays with multitasking being more present, this behavior may change (Buchanan, M., 2010). The need for a bandwidth optimization on mobile device will become stronger so I will give several solutions aimed at the android platform later in this paper.

3.3 3G Network specifications

The mobile data traffic is growing at a very fast rate. It is expected that an average user will use 5 times more bandwidth in 2014 compared to nowadays. Additionally, the number of mobile devices connecting the internet is growing and expected to reach five billions units by 2014 according to Cisco (Marguerite Reardon, 2011). Global mobile data traffic tripled between 2009 and 2010, growing ten times faster than the voice traffic (Mike Hibberd, 2010). For most operators, managing the 3G connections in densely populated area is already complicated so these forecast show that solutions need to be found soon. The following solutions are currently being studied:

- Increase network capacity by building more cell towers and stations and upgrade the current infrastructure to support next generation technology such as WiMax and LTE.
- Optimizing data transfer by implementing compression technique to decrease the amount of data sent over the air.
- Wifi offloading whenever possible so that common internet connection is used instead of mobile one.
- Subscriber monitoring and flexible payment so that the user is conscious with what he is using and self monitors his data consumption to respect his data plan.

As of today, Wifi offloading seems to be the most promising solution and has already been in use in the US and Hong Kong where its part is reaching 20% and 80% respectively of the global data traffic (Luna, L., 2011). All of these solutions are network based and as of today none of them are designed to optimize the traffic on a specific device.
4 Practical measurements with existing products

After studying the theoretical part, it is interesting to see what commercial products can offer in term of traffic shaping. For that purpose, I selected four software and made a series of tests to understand their capabilities and performances.

4.1 Parameters for the tests

The four software tested in this paper have the same aim: optimizing the available bandwidth. The first parameter to test is, if the objective of the software is fulfilled and to compare their approaches. The second and third parameters are the CPU and memory consumption of the application. The last parameter is to find out how user-friendly the software is.

In order to check the CPU and RAM usages of each software, I will run several tests. The first one will check the system usage while idle. Second test will check while heavily using the bandwidth but without setting up any traffic shaping rules. The third test will heavily use the bandwidth and then limit it. The user-friendliness can be a subjective factor, so I will base my decision according to how easy the settings can be found and used and how clear the interface is.

The computer used for the tests is a laptop, Dell Inspiron 1720 running Intel Core 2 duo T9500 (2.60Ghz) and 4GB of RAM. In order to get a similar environment for each of the software tested, I set up a virtual machine running Windows 7 with 1GB of Ram allocated and cloned it. On each of the clone I installed one traffic shaping software and ran the tests with one clone at a time. The virtual machine was connected to the internet through a bridged connection to the host’s network interface.

4.2 Netlimiter

Netlimiter only offers speed limitation based on applications. The software can also act like a firewall by blocking incoming or outgoing traffic. Each application running is displayed on the main page and its bandwidth usage can be set up very easily by ticking a box and entering the wanted value in kB/sec. Every application can be unfolded to let the user know which process is in use, and again each process can be unfolded to show all the current connections. This allows the user to limit the application or a specific connection at will. It is also possible to create filters using different criteria:

- IP range
- Port range
- Protocol
- Application
- Adapter / zone

The software is very easy to use, the interface is simple and all the important information is present on the main page. Again the software can only limit and/or block the incoming and outgoing per application, so it somehow limits the options offered to the user.

4.3 NetBalancer

Netbalancer has the ability to limit the bandwidth to a wanted value as well as creating QOS rules. The software displays all the processes in use by default and this can feel really complicated and unclear. Fortunately, it is possible to show only the online ones by modifying the settings. For each process it is possible to set the following rules:

- Low
- Normal
High
Block
Limit … (and then set the value on the limit)
Ignore

For each process it is possible to get all its related information: all the connections established to it, the current/total bandwidth usage since Netbalancer is running. It is also possible to ping and trace each connection which can be an interesting feature for troubleshooting problems.

The QOS part is specified in the settings, it is called level severity. A value is defined for it, (default is 50) and it dictates the speed difference between each application with different network priority. The bigger is the value, the bigger will be the speed difference. In the settings, it is possible to activate the traffic shaping when a desired percentage of bandwidth usage is reached (by default this value is 80%). In order to know what the maximum bandwidth available to the machine is, the software makes a calculation every hour by default.

The interface is simple but displaying all the processes on the main page can be confusing. There is no possibility to create a specific rule using criteria so each process needs to be dealt with individually.

4.4 CfosSpeed

CfosSpeed is the most complex software of the list. It allows the user to use QOS and limitation but only for outgoing traffic. There is no main page when launching the application, only a graph displaying the bandwidth usage on the bottom right of the screen. By right clicking on the icon situated on the taskbar, it is possible to access the settings. The software offers a lot of options, it allows specifying the priority based per protocol and/or application. Each of these protocols and applications are sorted per category. When it comes on bandwidth limitation, only the upload speed can be set. An interesting feature of this software is the ability to create what is called “online budget”. It allows the user to set a limit in GB of bandwidth usable per month for the selected interface. This feature is perfect for users who have a limitation from their provider (e.g. 2GB per month). CFOSspeed also integrates a firewall providing protection against network intrusions. The program is aimed at LAN as it provides the option to broadcast information on the network, so all the computers running cfos can work together.

The user interface is complicated, and it takes time to fully understand where which option is. Once used to the configuration, it becomes easy to create rules based on each protocol. Cfos provide a basic graph showing the bandwidth usage, but if the user wants to get more information about the current connections, the window provided only gives basic information: protocol, local address, remote address, program, tx, rx, tx/s, rx/s, time and class. The window looks like a text document open with notepad and no user interaction is possible.

4.5 Traffic Shaper Xp

This software has not been implemented specifically for windows Seven and is several years old, but it has been reported working so I decided to include it into my comparative. Traffic shaper XP allows the user to control priority and limitation of bandwidth. However, this action is made possible by setting up rules. Each rule must be created one by one and follow the logic of a firewall, meaning that the top rule will be applied first and the bottom one will be applied last. The possibilities are limited, only TCP protocol is supported and for each rule the user has to specify the remote and local address and port. This shows that the software is only oriented on controlling connections and not the process related to it. Each rule can limit
the upload and/or download of a connection to a fixed value. It is also possible to decide its priority. Finally, it is possible to select for which network interface the rule should be applied.

The user interface is simple and clear, the main page is divided into two parts: the first part shows the set of rules created and the second part shows all the current connections. It is possible to right click on a connection to create a custom rule. By doing that, all the information related to the traffic will already be filled and the only thing left to do is to choose the limitation and/or the prioritization.

4.6 Technical aspects

In order to have a clear understanding of the software mechanics, I asked each of the companies developing the application a basic explanation about how it works. SeriousBit developing Netbalancer gave me the following answer: “Basically it works like a firewall - it has a network intermediate driver, a windows service and the user app. The driver catches all net packets, sends them to the service which analyzes them and returns them back to the driver in a special order to get the desired prioritization. Finally the user app connects to the services to get all the display info and send user commands and settings. All parts except the driver are done in .NET/C#, the driver is written in C.”

The answer from CfosSpeed was less detailed but is still giving an idea: “Basically it's very easy:

- We measure the congestion of the connection by sending pings and timing the replies.
- We buffer outgoing packets and decrease the send speed if the line is congested in tx direction.
- We close the TCP windows of receiving streams when the line is congested in rx direction.”

Unfortunately I did not receive any answer from Netlimiter nor TrafficShaper. Each of the software installed a driver on the network interface so we can assume that they all work more or less like Netbalancer.

4.7 Comparison

The two first criteria to test are CPU and Ram usage. In order to check those values, I ran the “idle” test with the bandwidth software installed and running with no bandwidth usage in the background. The two other tests are using heavily the bandwidth by using Youtube, downloading a large file on newsgroup, downloading a large file on bittorrent and running a FTP server. The “no rule” test is checking the system CPU when no specific rule is applied; the software controlling the bandwidth is not used to do any traffic shaping but is still inspecting the packets. The “rule applied” test is limiting the overall bandwidth available to 100MB/sec. Note that the maximum bandwidth available during these tests was about 800MB/sec download and upload. The tables 4.1 to 4.4 are showing the result of those tests.

<table>
<thead>
<tr>
<th></th>
<th>Netlimiter</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>idle</td>
<td>no rule</td>
<td>rules applied</td>
</tr>
<tr>
<td>CPU usage</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>Memory usage</td>
<td>service : 2.5MB client : 6MB</td>
<td>service 3.5MB client : 6MB</td>
<td>service 3.5MB client : 6MB</td>
</tr>
</tbody>
</table>

Table 4.1 : Netlimiter results
We can see that all four software are using a low amount of Ram, on the other hand the CPU usage is variable. Netlimiter and CfosSpeed are not using CPU in any condition. Netbalancer can use up to 20% when no rule is applied. My assumption is that when no rule is applied, the software is still inspecting every packet and this can take a lot of resources under heavy bandwidth usage. When limiting rules are applied, the incoming data is lowered, thus less packet needs inspection. Overall Ram usage can be negligible, but a too heavy CPU usage can be a problem as it will slow down the entire system.

The four software are not offering the same possibilities and are not using the same approaches. Netlimiter is only offering bandwidth limitation with no possibility to create prioritization. Netbalancer allows doing both, but has a higher CPU usage. CfosSpeed approach is limited; it is only possible to set how much data a program/protocol can send. It is possible to create QOS rules but there are many protocol and program to look into and selecting the right priority rule for each of them can be long and complicated.

Finally, Traffic Shaper XP is using a moderate amount of CPU but is limited to TCP protocol. It is not possible to create any rule managing UDP, the result is that no QOS rules can be set to improve VOIP or similar services using that protocol.

The four software presented are not offering the same capabilities but all of them are improving the quality of the network connection when set up properly. With them the risk of congestion is very limited and the user is free to decide about the allocation of his network resources.
4.8 Interview

After finishing the tests on the software stated above, I wanted to get more information about the approach used by the developers. I also wanted their opinion about mobile device software and maybe get some hints about a possible implementation.

What appeared from the answers is that nowadays the high price of a 3G connection is limiting the use of bandwidth intensive software. However these prices will sooner or later drop and it can then be interesting to have such a solution which will make a difference between a usable and a slow clogged device. Among all the companies interviewed, none of them is planning on developing a solution for mobile devices, the main reason being commercial uncertainty.

These answers are showing that a traffic shaper for mobile device is not yet necessary but would still be welcome in prevision for the future. Handsets are becoming more and more polyvalent with IP telephony, phone tethering, video streaming and online gaming so with this information in mind, a lowered cost of 3G connection would develop even further these services and in parallel make a traffic optimization software more needed.
5 Implementation on Android : tcDroid

The previous chapter gave an overview of what is possible on a computer when the user wants to manage his bandwidth. On this chapter we move on to the implementation of Traffic shaping software designed for the Android platform. The objective is to give the user control over the software using network access.

5.1 Overview

The aim of the software is to control the bandwidth usage on the device. Unfortunately the Android SDK (Source Development Kit) does not provide any API (Application Programming Interface) giving access to the network configuration. HTC integrated a well known tool into their build called TC (traffic control). The problem is that this tool is working only using command line and requires a dozen of them to create a simple configuration. In order to make this process easier, I will create a User Interface which will create a script using user defined parameters. Once the script will be generated, it will be executed as root by the user interface. The script itself will be saved in the sdcard in order to use it again after a reboot.

5.2 Environment

Android is an operating system working on a great amount of different hardware. Manufacturers are also adding their own tools and drivers for their devices. As a result the solution I propose may not work on all Android devices available on the market. The handset used is a HTC Desire HD running Android 2.2 Froyo. The manufacturer build is 1.32.405.6. It is important to note that each build have slight differences and the compatibility of the software can be altered by it. The development part is done using Android SDK on Eclipse running on Windows 7 x64.

5.3 Android restrictions

The first thing to take into consideration when developing software for a mobile device is the system capability. The amount of RAM and CPU is limited and an intensive usage will cause a battery drain lowering the life time of the device. For my solution, I am using TC which is a utility part of the iproute2 package. This tool is integrated in HTC devices but has several restrictions. It requires a root access on the device or no rules can be applied. Additionally, the following queuing discipline are not supported : SFQ, TBF, PRI0. It leaves only the classfull queuing discipline CBQ and HTB. TC is the most adapted tool as its system resource consumption is very low. The user interface will not do much processing either so the final solution should not drain the battery.

5.4 TC Script

In order to create a working script on Android, I started to work on it on Linux to understand and be more comfortable with the syntax TC is using. I followed the same restriction applying on Android and used only the parameters available on the mobile platform. As seen before in this thesis, CBQ can sometimes be unreliable, for that reason I will be using HTB queuing. Figure 5.1 shows the different parameters applicable for it.
The idea is to create a simple starter configuration and increase its complexity when necessary. The figure 5.2 shows a basic script.

```bash
# interface to shape
IF="wlan0"

set -v
# adjust ceil to 75% of upstream
CEIL="1000"
# set prio ratio, high = medium : ceil/2 ..., low = ceil/4
HIGH=$(( ($CEIL / 2) )
MEDIUM=$(( ($CEIL / 2) )
LOW=$(( ($CEIL / 4) )
# flush previous tc config
tc qdisc del dev $IF root
tc qdisc add dev $IF root handle 1 htb default 20
# create 3 classes to handle low, medium, high prio
tc class add dev $IF parent 1: classid 1:1 htb rate $CEIL ceil $CEIL
  tc class add dev $IF parent 1: classid 1:10 htb rate $HIGH Kbit ceil $CEIL Kbit
  tc class add dev $IF parent 1: classid 1:20 htb rate $MEDIUM Kbit ceil $CEIL Kbit
  tc class add dev $IF parent 1: classid 1:30 htb rate $LOW Kbit ceil $CEIL Kbit
# pifers:
HIGH="1:10"
MEDIUM="1:20"
LOW="1:30"
U32="tc filter add dev $IF protocol ip parent 1:0 u32"
$U32 match ip dport 80 0xffff flowid $HIGH
$U32 match ip dport 21 0xffff flowid $LOW
```

This script will shape the interface wlan0, which is in this case the wifi, and set a maximum bandwidth of 1000 Kbits. This available bandwidth is then divided into three
values attributed to high, medium and low priority. Then a root qdisc is created using HTB queuing. Three children classes are then added using the predefined bandwidth allocated by priority. The last part is creating filters in order to send the packets to the right queue and thus making the traffic shaping effective. The figure 5.3 shows the configuration in a form of a diagram for easier understanding.

![Figure 5.3 : HTB script illustration](image)

With the user interface, it will be possible to customize the number of queue, the bandwidth available for each of them and which protocol to send in which queue.

In order to obtain an efficient traffic shaping solution, it is important to clearly understand how the packets are handled by the system. The figure 5.4 describes how the data is treated by the kernel. First it reaches the ingress interface which redirects the packet to the IP stack or forward it directly according the rules already set up. The IP stack is responsible for transmitting packets to the relevant program it belongs to. Then it will forward the packets to the egress interface which will in turn send them in the relevant qdisc where the shaping will occur. It is important to note that traffic shaping can only be applied on the data we send. The only treatment possible for the data we receive is policing, which means dropping the packets which arrive at a too high rate.

With this information in mind, I created a default configuration for the egress node in order to improve general performances. In this script, I created three level or priority. The higher priority handles packets which have the ToS minimum delay set up so that interactive protocols (such as telnet or ssh) are very responsive. It will also handle ACK packets so that downloads are not slowed down when an upload is ongoing (for example an email with an attached file). The low priority will be used for bulk background traffic such as email and FTP transfer. The medium queue will handle all the other cases. The user will have the ability to edit the default settings and add his preferences.

For the ingress node, only traffic policing can be applied. By default no rule will be applied but the user will have the possibility to limit the traffic on his selected port at a specific rate. Thanks to this function, the user can select which application should get more bandwidth in the case of concurrent download.
5.5 User Interface

My application is aimed at users who have a basic understanding of network connection. The main page is leading to four sections: upload, download, wipe rules and exit. The upload page is shown in figure 5.5, it gives the ability to add port on 3 level of priority. It then creates a script aimed toward the Wifi or 3G interface. Once the script is generated, it is possible to launch it. The script will remain on the sdcard so if the device is restarted, the user should just launch the script instead of re-entering all his settings.
The download page is almost similar to the upload page, the difference being that the user needs to enter a value to limit the bandwidth rather than selecting a priority queue. The download page is shown in figure 5.6.

The last section called wipe rules do not lead to a new page, it only clears all the rules and reset tc to its original settings.
6 Implementation evaluation

During the whole development of my application, I ran tests to be sure that the code was working. Now that the main part of the software is done, it is possible to run extended tests to actually observe an improvement of the user experience. The first part of the test will focus on the downstream. As it is not possible to shape the download, the only test possible is to limit the traffic at a fixed rate and see if this parameter is respected. Then several tests are conducted on the upstream to observe the behavior of the traffic in different conditions.

6.1 Downstream

When a software is downloading at its maximum capacity, other concurrent applications requiring bandwidth can be slowed down to the point that they are not usable anymore. In order to counter this behavior, I offered the user the ability to limit a stream to a selected fixed rate for a specific port. For this test I downloaded SpeedTest available on the Android Market. This application starts by selecting a nearby server, it then pings it and makes a download and upload test on the port 80. After the tests are complete the results are reported on the main page.

On the first test, I ran SpeedTest three times without any kind of limitation and got the result shown on figure 6.1. I then limited the port 80 to a rate of 512 Kbps and ran the test three times again, the results are shown of figure 6.2.

The results obtained are the ones expected. The downstream has been reduced below 512Kbps while the upstream and ping remained more or less similar.
6.2 Upstream

When the Upstream is used at full capacity, it often creates congestion and the whole link becomes very slow. Thus, it is important to keep the speed under its maximum capacity to ensure a pleasant user experience. For the test, I installed File Expert which allows the user to share the content of his sdcard other FTP or HTTP using user-specified port. After setting up the HTTP share, I downloaded a large file with a computer from the phone and ran a speed test. The result is shown in figure 6.3. I ran the same test with the optimization set to have a ceil value of 1000Kbps and to put port 8080 in low priority with available bandwidth of 250Kbps, the result are shown on figure 6.4.

![Figure 6.3: no restriction](image1)

![Figure 6.4: port 8080 restricted](image2)

We can see from the screenshots that with no control, the latency is becoming higher and the download speed is decreasing due to the uplink being saturated. With optimization, the latency and download speed came back to normal values.

6.3 Conclusion

The tests made on my implementation showed that the software is working. The implementation of the tc tool on Android is efficient and offer a lot of flexibility to the user. With more time it is possible to improve the user interface so that the rules can be applied to applications rather than just port numbers. With proper settings, a user can now effectively improve his internet experience with concurrent connections.
7 Results

There are numerous ways to optimize the traffic on a network. This is commonly done by the ISP for some companies’ specific needs. For example, if the organization needs a high performances VOIP service, the ISP will ensure that the path of that specific data flow is prioritized using MPLS technology. With that method each router on the path will reserve the needed network resources to guarantee the best results. For common customers, the Internet connection is using a simple FIFO queue where all the data transfer has an equal priority. If the user wants better performances, he has to manage his traffic before the dataflow leaves his computer. For that purpose several commercial products exist for computers. According to the tests done in this thesis, they all fulfill their objective and offer variable options. The approach of a software designed for Windows operating system is always the same: implementing a specific driver for the network interface and create a GUI that interact with that driver. On Linux environment this kind of software is completely free and even integrated into the system. It only requires a basic understanding of how the tool is working but can be confusing for someone who has a limited knowledge about network. Moreover, Linux is offering a transparent solution; the user knows exactly what he is doing and which algorithms are used so that he can choose what will fulfill his needs. On the other hand, Windows based solutions give no information about the method employed.

The growth of mobile devices in units as well as in bandwidth usage is exponential. This growth is superior to what the current infrastructure can handle and ISPs are looking for solution to relieve their 3G network. Among those solutions, one can find WiFi offloading, flexible subscriptions or expanding the network capacity. All of them are aimed to optimize the ISP’s network but none of them are offering the user the same possibility as on a computer: managing our own traffic before it enters the 3G. When questioned, none of the commercial companies offering a Windows solution was interested in a mobile device solution, the main reasons being the commercial uncertainty and the work involved in developing for an entirely new platform.

The decision to base my solution for mobile devices on Android comes from the fact that this Operating System it is actually based on Linux. As a consequence, the method used on Linux could be applied on Android. This is true with some restrictions as Google’s operating system is using a limited version of the tool found on Linux. However this binary, just like its Linux counterpart requires a root access to be used. The GUI created in this thesis is generating scripts more easily so that common users can still use it. With no user interaction, the generic script is applying common rules used to improve the traffic in general. The application becomes really useful when the user is applying his own rules which will scale with his usage. The tests showed that the solution is effective and fulfilling its role; the software is offering the ability to manage the bandwidth going in and out of the mobile device.
8 Conclusion and future work

The main objective of this thesis was to provide a traffic shaper for Android in order to give an answer to several problems related with poor bandwidth management. For a user wishing to multitask, the absence of bandwidth control means low performances for all the concurrent applications causing cuts in a Skype call for example. My application allows the user to manage the traffic so that he can decide which software deserves more attention. While using the Android phone as a hotspot, it is also possible to sort the priority between the different users connected and the services they are accessing. As a result tcDroid is making a difference between a slow device and a responsive one. The problems posed in this paper are properly answered.

For now my application is a prototype rather than fully optimized software. The interface can be improved to offer more possibilities to the users such as adding filters to application or IP addresses rather than just port numbers. Others approaches can also be studied, for instance implementing a proxy solution so that the device itself do not handle the shaping. Such a solution would be platform independent and work for any mobile device. Local solutions for iOS can also be studied. My current implementation would not work as it is because iOS does not have an integrated implementation of tc. However the system being based on Unix BSD, a jail-broken iPhone can install the utility ipfw and theoretically get similar result with Android.
REFERENCES

Allot Communications Ltd. Global mobile broadband traffic report. 2011


dotMobi, AKQA,. Mobile Internet Usage and Attitudes Study. May 2008

Fineberg, V. A practical architecture for implementing end to end QoS in an IP network, January 2002


Marchese, M., 2007. QoS over heterogeneous networks. West Sussex: John Willey & Sons Ltd


SpeedTest. [online] Available at : <http://www.speedtest.net/about.php> [Accessed 6 May 2011]


Appendix : Interview

**Question**: What do you think about traffic shaping/ bandwidth optimization on mobile devices? Is it necessary? Do you think traffic shaping would be useful to implement on mobile devices?

*Netbalancer*: I'm not sure about the commercial perspective of traffic shaping on mobile devices, usually mobile users don't tend to use traffic intensive apps on their phones due to the high prices for mobile traffic, but at the same time these prices could stimulate them to use a traffic limitation tool. So still the commercial perspective is uncertain.

*Cfos*: sure, it's necessary for every device. it depends on the traffic you have: if you only read web-pages, then it's not that important. But if you have volume transfers (up or down), then it makes the difference between a useable or a clogged and slow internet.

**Question**: Do you have a project of porting your application on Android platform? If no, what would be the reason; if yes what would be its limitation/extension?

*Netbalancer*: Currently we don't have any plans on porting NetBalancer on Android due to its commercial uncertainty.

*Cfos*: we're developing software for 17 years now for the dos/windows platform. It would be a lot of work to switch to a new platform.

**Question**: I am currently studying your software on windows 7 and I wonder if you could give me some technical information about how it works?

*Netbalancer*: Basically it works like a firewall - it has a network intermediate driver, a windows service and the user app. The driver catches all net packets, sends them to the service which analyzes them and returns them back to the driver in a special order to get the desired prioritization. Finally the user app connects to the services to get all the display info and send user commands and settings. All parts except the driver are done in .NET/C#, the driver is written in C.

*Cfos*: basically it's very easy:

1) We measure the congestion of the connection by sending pings and timing the replies.

2) We buffer outgoing packets and decrease the send speed if the line is congested in tx direction.

3) We close the TCP windows of receiving streams when the line is congested in rx direction.