

Automatic Network Bandwidth Control Application for Linux-based Internet Connections

Korinto Miguel G. Aguinaldo and Joseph Anthony C. Hermocilla.

I. INTRODUCTION

A. Significance of the Study

With the advent of data sharing and synchronization over the internet, many applications have become dependent on bandwidth allocation in order to function properly. Much of current user-friendly software requires frequent to constant internet connection to fully realize their primary function. The usual examples of this are real-time network games such as World of Warcraft and BitTorrent services. These kinds of programs may be given a larger bandwidth allocation, depending on their setting. In a network of computers using a common port gateway, a larger bandwidth allocation for one computer usually generates smaller bandwidth allocation for others.

Bandwidth control or traffic shaping is a necessity in maintaining a fair bit-rate for all computers in the network. Through bandwidth control, the bit-rate of a computer can be throttled down in order to give larger bandwidth allocations for other computers. The most important task in network administration is to regulate the connection of the network to internet services and to every connected computer. It is the prerogative of the administrator to be able to know how he or she will achieve a balanced bit-rate for all of the computers and when a computers connection should be sped up or throttled down.

Recent Linux distributions have the shell command *tc* which allows the user to filter and throttle network traffic. However, this tool may present confusion to those who are new to networking concepts because its command syntax may be complicated and hard to understand.

The purpose of this project is to produce a user-friendly application that serves as an command generator for *tc*. The application must present *tc* syntax in a way that is easily understood, and can be modified by keyboard and mouse. Finally, the application must be able to automatically apply *tc* commands so that the network administrator will not have to constantly monitor the network so that fair bit-rate for all computers is achieved.

B. Statement of the Problem

Network administration involves regulation of internet use. There are tools available to do this, but these tools are either difficult to be used or too complex to be understood by

inexperienced administrators. A simple program that could integrate known tools such that it enables the administrator to easily maintain fair use of the internet connection is ideal.

C. Objectives of the Study

The general objective of the Special Problem is to alleviate the difficulties involving network administration.

Specifically, the aim is:

- 1) To develop an application that can automate traffic control; and
- 2) To develop a user-friendly graphical user interface for the application.

II. REVIEW OF LITERATURE

Linux-based operating systems is popular in computer networking, especially when it is known that the Linux networking attributes are configured and inspected completely. [1] We can infer that popular, even local, networking services are usually run from Linux servers, especially when the functionalities and instructions for handling these programs are explicitly included in their installation. This allows scrutiny for young network administrators.

The other computers will be dependent on the server for their internet connection. The server, or router terminal, may allocate bandwidth to itself and other computers, or "dependent terminals", in the network.

Imagine that these dependent terminals are being used by members of an organization, as employees or students. Fisher implies that most of these kinds of people usually visit social networking sites such as Facebook and Twitter first before proceeding to the actual purpose of their internet access. [2] Also, commonly during their breaks, these people access sites involving streaming media such as entertainment and news sites, which are usually encountered in the internet. [3]

According to a statistic shown by a White Paper from Symantec MessageLabs, sites that has streaming media content is one of the highly preferred to be blocked by administrators, along with advertisement and popup-generating sites. [4] The common awareness of why this happens is because these sites take up a lot of bandwidth. Streaming media and other kinds of download services make up an increasing percentage of traffic in the internet. [5] Such high-bandwidth connections take a toll in local networks, especially when many dependent terminals access these sites at the same time such as during breaks.

Presented to the Faculty of the Institute of Computer Science, University of the Philippines Los Baños in partial fulfillment of the requirements for the Degree of Bachelor of Science in Computer Science

When these concurrent signals pass through the gateway(s) handled by the router terminal, a large volume of information may result to collisions such that data is lost and errors occur. [6] It is also known that a constant stream of bytes passing unimpeded through the gateway(s) may cause a slowdown since this stream has its own share of bandwidth. [7] According to Tunnicliffe, the access of certain sites must be regulated in order to optimize the bandwidth share for each computer. [8] Though commercial network traffic control tools available, these software do not present the actual configuration that is done on computers. Furthermore, simple techniques can effectively maximize bandwidth allocation with little overhead. [9]

III. MATERIALS AND METHODS

A. Theoretical Framework

1) *Bandwidth*: Bandwidth, is a measure that describes the rate of information transmitted over a connection. [10] This term is commonly used especially when multiple users share a common connection to the internet. In this setup, each user has a share of the bandwidth. However, as a user starts more programs to connect to the internet, it is possible that these programs expand the computer's bandwidth requirement. A simple analogy of this is being the number of lanes in a highway. More lanes (more bandwidth) mean more cars can pass through (more data can be transferred). [11]

The problem arises when many computers connect to the internet at the same time, such that the network cannot fulfill the requirements of all the computers. When this happens the administrator must regulate the network traffic.

2) *Traffic Control*: Traffic control is to manage the network traffic to maximize the use of network resources and reduce information loss due to congestion. [12] Congestion results from having too many concurrent connections, usually on a few ports, at the same time, so as to exceed the connection limit of the network devices. Congestion may cause loss of packets, since a congested connection may sometimes not serve all the packets that passes through it. In the previous analogy, congestion arises when too many cars are to pass through one highway, and the highway cannot possibly accomodate all of them (Fig. 1). Network administrators must regulate the connections of the computers connected to the internet so that all computers will have a fair share of the bandwidth.

The essence of traffic control is controlling how packets are trasmitted. When a Linux computer needs to send or receive packets, it decides which packets are to be sent, delayed or dropped. A buffer or 'queue' where packets that are needed to be trasmitted are stored momentarily is used to achieve this effect. How the packets are stored in this queue is defined by a 'queueing discipline' and is configurable in Linux systems. An example of this in Linux is Stochastic Fairness Queuing, where there are sub-queues and each sub-queue gets a turn to send their packets in a round robin fashion. [13] Setting the queueing discipline for a device or computer also allows setting its transmission rate.

3) *Java*: The entire application will be consisting of a Linux service and a web application to control it. The service

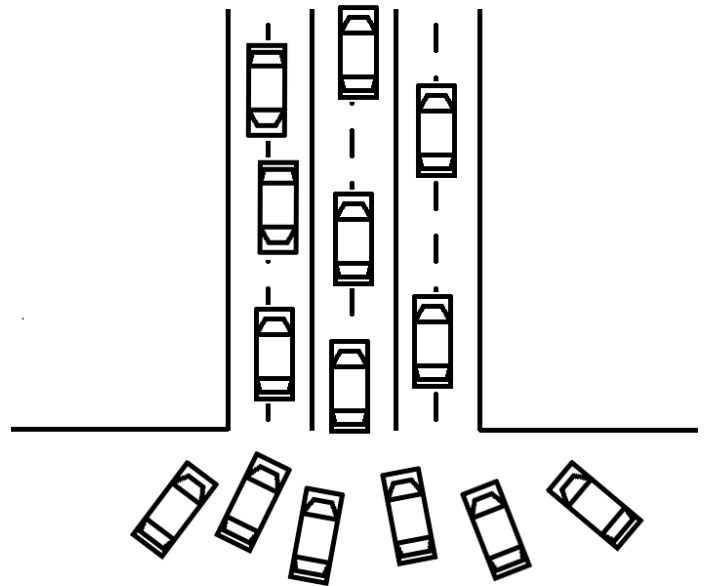


Fig. 1. Congestion. Too many cars wanting to pass will have a hard time going through.

will be implemented using Java and the web application will be built using Google Web Toolkit to have a user-friendly Graphical User Interface (GUI). The web application will use Java Servlets to pass commands from the web application host to the service.

4) *Command line tool – tc*: Traffic control in Linux is achieved through the command line tool *tc*. The tool can show and modify traffic control settings of other computers in the network, and can assign queuing disciplines to devices.

5) *Command line tool – netstat*: This command line tool displays the network connections of a computer. The actual command used is

```
netstat -Wn -tu
```

B. Date and Place of the Study

Software construction will be done from November 2011 to February 2012. The software will be tested at the laboratories of the Institute of Computer Science, University of the Philippines Los Baños.

C. Design and Implementation

1) *GUI Design*: The overall web application design is constrained by the capabilities of the Google Web Toolkit (GWT), browser Javascript limitations, and the host serving the web application. The GUI is composed of pages handled by clickable tabs. Fig. 2 will illustrate.

Though the internal structure of the client-side GUI is design to be deployed to any web server supporting Java Servlets as a Web Archive file, the product application's JAR file contains Jetty, a small web server written in Java, to launch the web application. HTML pages and GWT-compiled

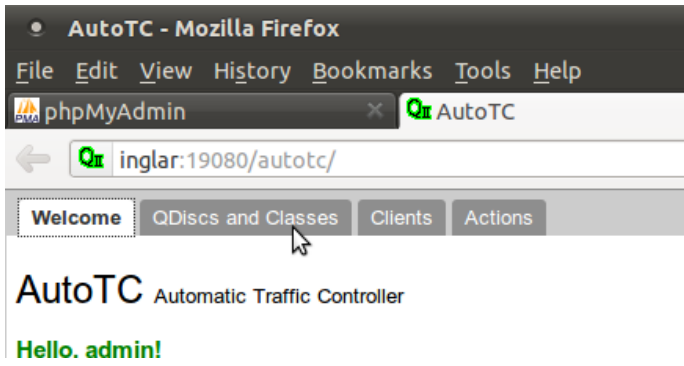


Fig. 2. Users can select a page by clicking on tabs.

Javascript files are put in a folder so they can be loaded by Jetty.

The following items can only be viewed in the web application.

2) *Welcome Page*: In the welcome page, information about the application and the current user is displayed (Fig. 3). If there is no user logged in, the welcome page is grayed out and a log-in dialog box is on top (Fig. 4).

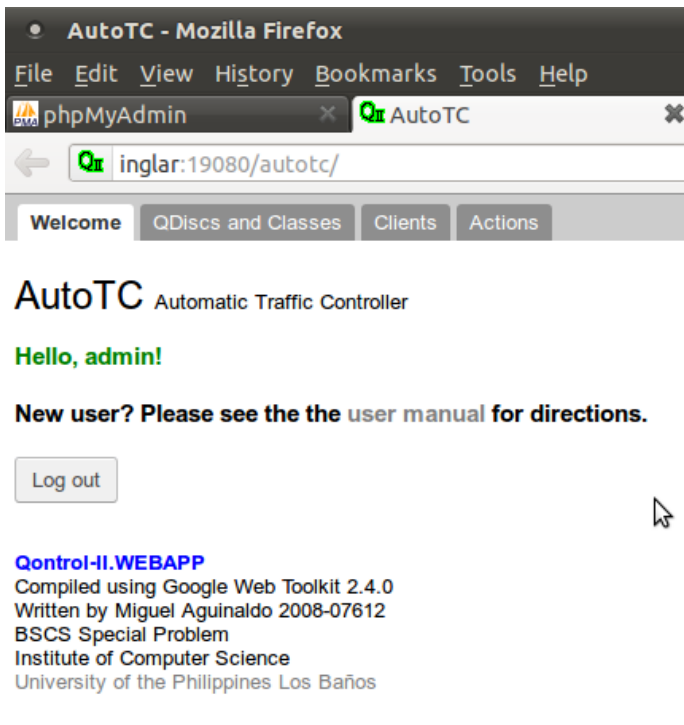


Fig. 3. Welcome page, when a user is logged in.

3) *QDiscs and Classes Page*: In the QDiscs and Classes Page, the user can edit the tc configuration of the current selected device. The current selected device can be changed from a drop-down list.

Each tc configuration command is of three types:

- 1) tc qdisc (parameters ...)
- 2) tc class (parameters ...)
- 3) tc filter (paramaters ...)

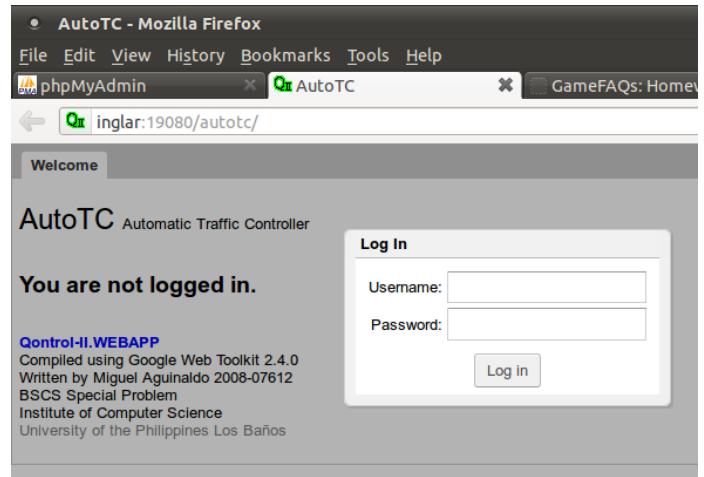


Fig. 4. Welcome page with a login dialog box.

Each type has a corresponding box, which contents show te current command's parameters and what can be done to that particular type. In the current implementation, tc QDisc, Class and Filter boxes are colored green, blue, and yellow respectively. These boxes are arranged in a tree-like manner, nesting boxes of dependent tc types if necessary. See Fig. 5 for an example.

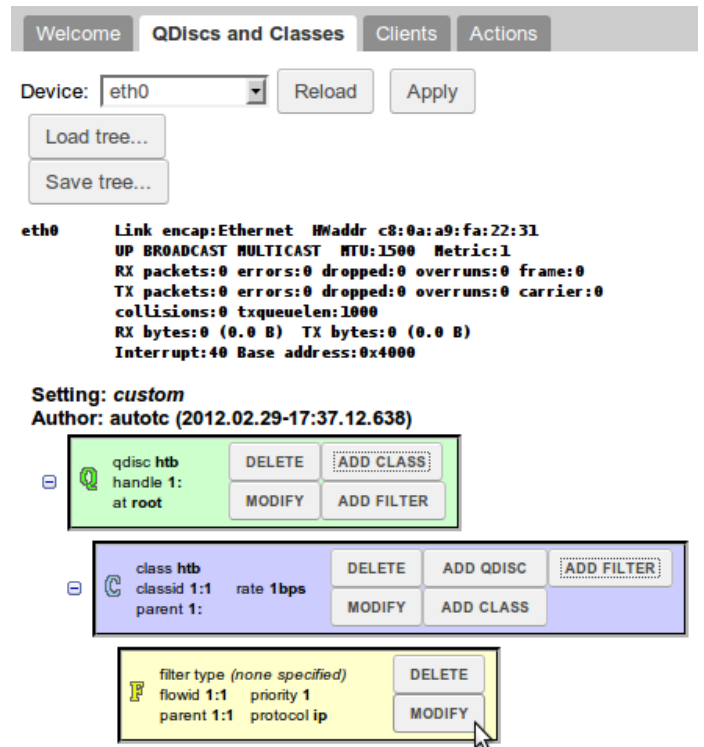


Fig. 5. A device's tc configuration is shown.

The user can edit the contents of each box by clicking the modify button. The parameters of each box is set through a dialog box. Fig. 6 shows an example where a tc QDisc box is edited.

The tc configuration of the current selected device can be

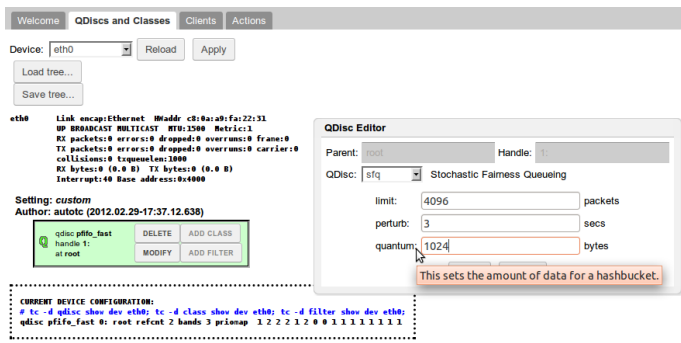


Fig. 6. Editing a QDisc.

saved into the MySQL database as a "template". This can be done by clicking the Save Tree button. Alternatively, the user can also apply previously-saved templates to the current selected devices by clicking the Load Tree button and selecting the template name from the dialog box that pops up (see Fig. 7).

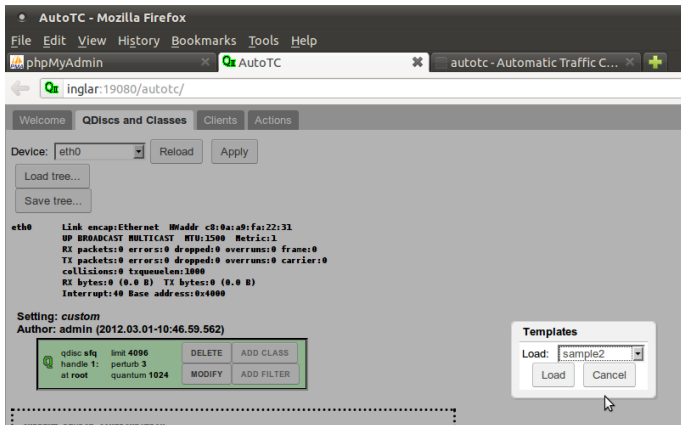
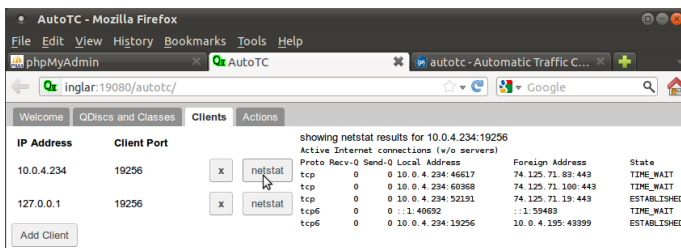


Fig. 7. Loading a template.

4) *Clients Page*: The clients page displays a list of clients that runs a small Linux daemon that sends the results of *netstat* through a socket. This daemon is contained in its own JAR file, so that it does not take up space for Java classes it does not use. This daemon is important in the automation process, since this gives the data that is used to know which *tc* configuration is to be applied.

The user may click a client entry's corresponding *netstat* button anytime to show the current active internet connections (Fig.8).

Fig. 8. Using *netstat*.

5) *Actions Page*: The Actions page shows criteria each client is tested, the *tc* configuration used when these criteria is satisfied, and the priority values of this pair (Fig. 9). The user can add and remove entries from this page, and by adding a row, the user can select which *tc* configuration is to be selected and applied and the criteria for its selection.

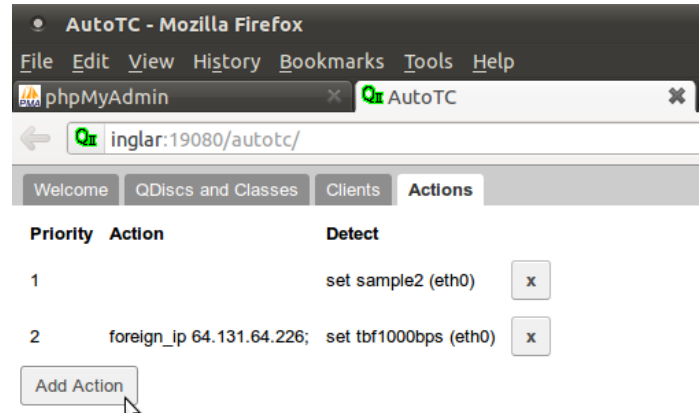


Fig. 9. A list of actions.

The Actions page shows the data used in automating the process of applying the *tc* configuration of a device. For example, anytime a computer in the network is using a Voice-over-Internet Protocol service, we would naturally want a fast internet connection for that computer. Given that the IP addresses and port numbers of the connection is known, the application must be able to detect this information and apply the necessary *tc* configuration if any. The logic for this is simplified by the following:

- 1) Get *netstat* data from each client
- 2) Find an action satisfying all IP address and port number criteria with the highest priority
- 3) Get *tc* configuration script and apply
- 4) Wait for a while and then go back to 1

IV. RESULTS AND DISCUSSION

To test the application, two JAR files are made: *autotc.jar*, which contains the web application and the daemon that automatically applies *tc*, and *autotc-client.jar* which contains the daemon that sends *netstat* results.

First, the file *autotc.jar* is run on the local machine (we shall call this computer "server"). A copy of *autotc-client.jar* is given to a computer (we shall call this "client") with an IP address of 10.0.4.234 and the daemon and the web application are run. The client is then registered into the web application (Fig. 8).

In this setup, it is not needed that the client is dependent on the server for its internet connection, since we must only prove that two things are running correctly:

- 1) That the server is able to get the *netstat* data from the client, and
- 2) That the server is able to change the *tc* configuration based on the *netstat* data.

Checking if the actual bitrate of the internet connections of clients has changed is not important, since that is restricted to the abilities of the *tc* command. What the application will consider is whether the *tc* script generated is run successfully or not.

In the test, an action was created to detect the IP address 64.131.64.226 and set the template **tbf100bps** when the server's daemon detects it from the client (Fig. 10).

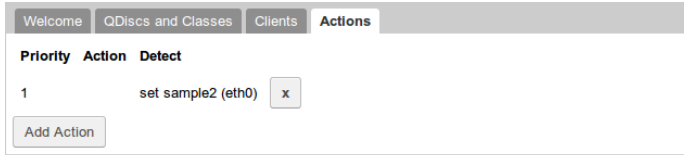


Fig. 10. Setting an action.

Fig. 11 shows the *tc* configuration from the web application (on a browser) and Fig. 12 shows the current *tc* configuration from the terminal.

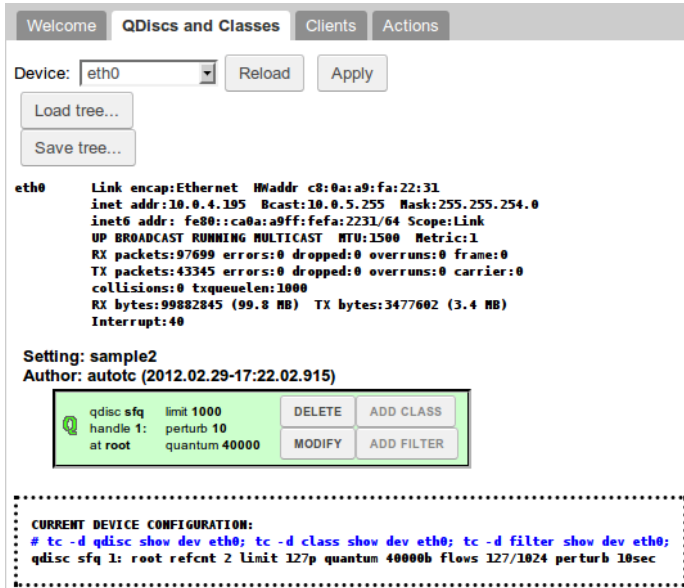


Fig. 11. How the *tc* configuration looks like before accessing 64.131.64.226.

While the client's daemon is running, it made an access to 64.131.64.226. After about three seconds, the *tc* configuration on the server changed as shown by Fig. 13 and Fig. 14.

V. CONCLUSION

As we have seen in the test, the application is able to detect the client's internet connections and apply the necessary *tc*

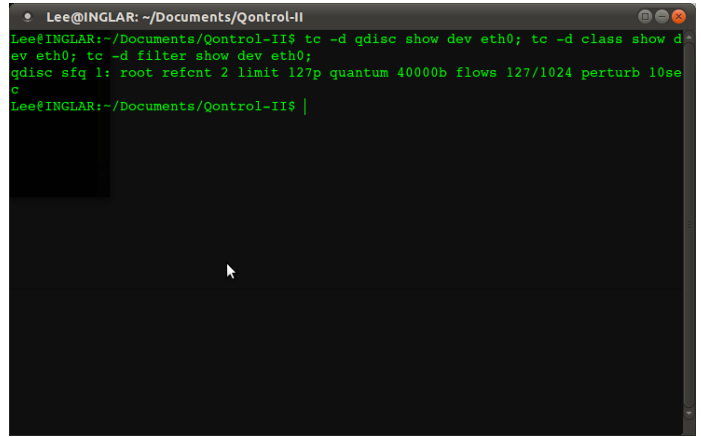


Fig. 12. The *tc* configuration as shown from the terminal before accessing 64.131.64.226.

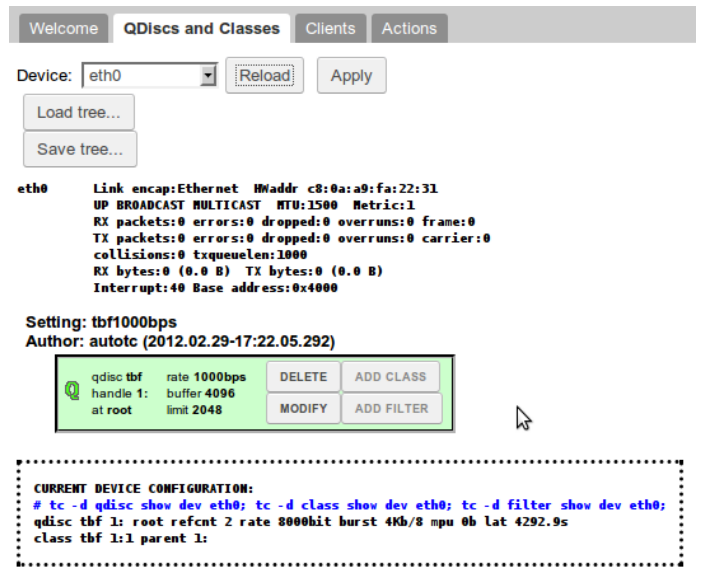


Fig. 13. How the *tc* configuration looks like after accessing 64.131.64.226.

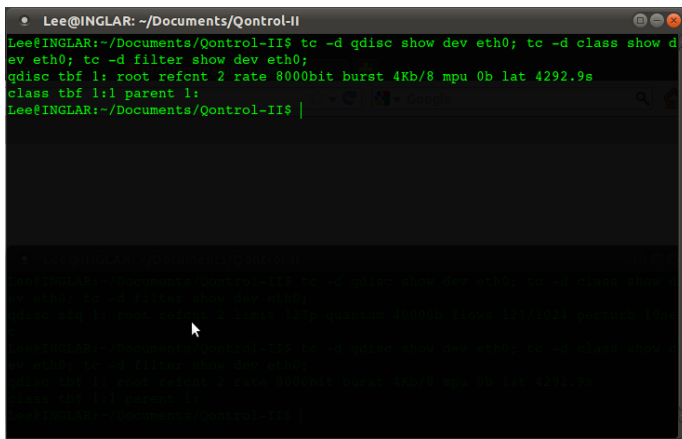
configuration for it. Furthermore, the application has fulfilled its primary objectives, which is being able to automatically apply traffic control commands and having a user-friendly GUI.

ACKNOWLEDGMENT

Many thanks to my adviser, Joseph Anthony Hermocilla, for his invaluable support for my Undergraduate Special Problem.

REFERENCES

- [1] B. McCarty, *Learning Debian GNU/Linux*. O'Reilly Media.
- [2] L. Fisher, "Study shows social network traffic is the least engaged," <http://www.simplyzesty.com/social-media/study-shows-social-network-traffic-is-the-least-engaged/>, 2011.
- [3] "How engaged is traffic from social sites?" <http://www.emarketer.com/Article.aspx?R=1008357>.
- [4] "Bandwidth bandits," White Paper, Symantec MessageLabs, Apr. 2010.
- [5] J. Kirk, "Study: Other network traffic surpassing p2p growth," <http://www.itworld.com/internet/62869/study-other-network-traffic-surpassing-p2p-growth>, 2011.
- [6] J. Awe, "The five most common network problems," <http://www.jidaw.com/itsolutions2.html>.



```

Lee@INGLAR: ~/Documents/Qontrol-II
Lee@INGLAR:~/Documents/Qontrol-II$ tc -d qdisc show dev eth0; tc -d class show dev eth0; tc -d filter show dev eth0;
qdisc tbf 1: root refcnt 2 rate 8000bit burst 4Kb/8 mpu 0b lat 4292.9s
class tbf 1:1 parent 1:
Lee@INGLAR:~/Documents/Qontrol-II$

```

Fig. 14. The `tc` configuration as shown from the terminal after accessing 64.131.64.226.

- [7] S. Lowe, "Common causes of network slowdowns," <http://www.zdnet.com.au/common-causes-of-network-slowdowns-120266265.htm>, 2002.
- [8] M. J. Tunnicliffe, "The measurement of bandwidth: A simulation study," <http://www.intechopen.com/articles/show/title/the-measurement-of-bandwidth-a-simulation-study>, 2010.
- [9] M. Marcon, M. Dischinger, K. P. Gummadi, and A. Vahdat, "The local and global effects of traffic shaping in the internet," in *Communication Systems and Networks (COMSNETS), 2011 Third International Conference on*, Bangalore, Jan. 4–8, 2011, paper 10.1109/COMSNETS.2011.5716420, pp. 1–10.
- [10] "What is bandwidth?" <http://www.wisegeek.com/what-is-bandwidth.htm>.
- [11] "Bandwidth explained," <http://www.findmyhosting.com/bandwidth-explained/>.
- [12] "Traffic shaping with linux," http://www.knowplace.org/pages/howtos/traffic_shaping_with_linux.php.
- [13] J. P. C. Ngo, "Qontrol: A linux-based network bandwidth control application," Institute of Computer Science, University of the Philippines Los Baños, BSCS Special Problem Report.



Korinto Miguel G. Aguinaldo is an undergraduate student under the BS Computer Science program of the Institute of Computer Science, University of the Philippines Los Baños. He likes programming, the Internet, and well-made coffee.