# UPLB NETWORK QUEUE SIMULATOR (UNQS): ANALYZING NETWORK PERFORMANCE FOR INTERNET BANDWIDTH MANAGEMENT

LEENSEY M. LAWAS

SUBMITTED TO THE FACULTY OF THE INSTITUTE OF COMPUTER SCIENCE
UNIVERSITY OF THE PHILIPPINES LOS BANOS
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE
DEGREE OF

BACHELOR OF SCIENCE IN
COMPUTER SCIENCE

JUNE 2018



UNIVERSITY OF THE PHILIPPINES LOS BANOS

Bachelor of Science in Computer Science



The following permission/s is/are granted for accessing this thesis:

| | |
|---|---|
| Available to the general public | YES |
| Available only after consultation with author/thesis adviser | NO |
| Available only to those bound by confidentiality agreements | NO |

LEENSEY M. LAWAS

Author

_____

Date Signed

PROF. DANILO J. MERCADO

Adviser

_____

Date Signed

The thesis attached hereto, entitled "**UPLB NETWORK QUEUE SIMULATOR (UNQS): ANALYZING NETWORK PERFORMANCE FOR INTERNET BANDWIDTH MANAGEMENT**," prepared and submitted by **LEENSEY M. LAWAS** in partial fulfillment for the degree of BACHELOR OF SCIENCE (COMPUTER SCIENCE) is hereby accepted.

PROF. CONCEPCION L. KHAN
Member, Guidance Committee

_____
Date Signed

PROF. JAIME M. SAMANIEGO
Member, Guidance Committee

_____
Date Signed

PROF. DANILO J. MERCADO
Chair, Guidance Committee

_____
Date Signed

Accepted as partial fulfillment of the requirements for the degree of BACHELOR OF SCIENCE (COMPUTER SCIENCE).

PROF. JAIME M. SAMANIEGO
Director, Institute of Computer Science

_____
Date Signed

DR. FELINO P. LANSIGAN
Dean, College of Arts and Sciences

_____
Date Signed

## BIOGRAPHICAL SKETCH

Leensey M. Lawas was born on September 25, 1996 and currently resides in Los Banos. She is the eldest daughter to Vernel and Cresencia, and has two siblings: Cresel (16) and Sevyer (7). Leensey finished her primary education in Maquiling School Incorporated, with honors, and her secondary education in University of the Philippines Rural High School, with honors. During her transition from elementary to high school, she was able to discover her interest in coding through a social networking site that allows HTML and CSS manipulation. She developed further interest upon taking up a class on coding in her high school years. Thus, she pursued a bachelor of science degree in Computer Science. Aside from coding, she also dabbles in creative writing as inspired by her other hobbies: watching films and shows, and reading books.

LEENSEY M. LAWAS

2013-68227

## ACKNOWLEDGEMENT

Extending my sincerest thanks to the people who helped me in this project: Mr. Mark Gironella, ICS technician; Ms. Ivy Joy Aguila, ICS instructor for helping me setup the server in the institute; Ms. Emergrace V. Puhawan, ITC Network Administrator; and Mr. John Eric H. Dao, ITC Network Technician for aiding in the port mirroring for my data collection.

Mr. Rene Lindio for some last minute help in debugging my program. Mr. Shem Cristobal for being my go-to for many of the problems I encountered during the programming aspect of my thesis.

Prof. Concepcion L. Khan and Prof. Jaime M. Samaniego for being encouraging and insightful panelists.

Prof. Danilo J. Mercado, adviser, for entrusting me with such a topic and for giving advice that goes beyond this subject matter.

Blockmates for the concern and support.

Online friends I met through my hobby of creative writing and the ones met through shared fandoms for being supportive and open to lend an ear when I needed it.

Friends for their unwavering support.

Family for their push and for bearing with me through my lowest of lows.

And last but not least, thanks to God, because only with Him are all things possible, including this study.

## ABSTRACT

LAWAS, LEENSEY M. University of the Philippines Los Banos. June 2018. **UPLB Network Queue Simulator (UNQS): Analyzing Network Performance for Internet Bandwidth Management.** Research Adviser: Professor Danilo J. Mercado.

Internet bandwidth is an expensive resource that is increasing in demand. As an alternative to meet those demands, UNQS was developed to simulate real traffic data and to help identify the optimal bandwidth setting, which can then minimize cost. Traffic data was collected using *ntopng* and had a bandwidth of 3,448.96 Kbps, a total of 554,392 flows and 432,616,056 packets with a total size of 273,715,167,385 bytes over a span of 8 days. The data was processed using the FIFO scheduling algorithm with different bandwidth constraints and a TTL of 60 seconds. Results showed that increasing the bandwidth reduced the amount of unserviced flows until eventually, all flows are serviced and that a minimum recommended bandwidth can be determined by observing the shift in the trend of the results. In conclusion, UNQS can be used to find the optimal bandwidth setting for a network given existing traffic data.

**TABLE OF CONTENTS**

## LIST OF TABLES

## LIST OF FIGURES

## LIST OF ACRONYMS

| | |
|---|---|
| **BOOTP** | Bootstrap Protocol Server |
| **DHCP** | Dynamic Host Configuration Protocol |
| **DNS** | Domain Name System |
| **FCFS** | First Come-First Serve |
| **FIFO** | First In-First Out Queueing |
| **HTTP** | Hypertext Transfer Protocol |
| **HTTPS** | Hypertext Transfer Protocol Secure |
| **IANA** | Internet Assigned Numbers Authority |
| **ICT** | Information and Communications Technology |
| **IP** | Internet Protocol |
| **ITC** | Information Technology Center |
| **LLMNR** | Link Local Multicast Name Resolution |
| **MS-DS** | Microsoft-Directory Services |
| **PQ** | Priority Queueing |
| **QoS** | Quality of Service |
| **SCFQ** | Self Clock Fair Queueing |
| **SIP** | Session Inititation Protocol |
| **SMB** | Server Message Block |
| **SNMP** | Simple Network Management Protocol |
| **SQL** | Structured Query Language |
| **SSDP** | Simple Service Discovery Protocol |
| **TE** | Traffic Engineering |
| **TTL** | Time-To-Live |
| **UNQS** | UPLB Network Queue Simulator |
| **UPLB** | University of the Philippines Los Banos |
| **VoIP** | Voice over Internet Protocol |
| **WFQ** | Weighted Fair Queueing |

**UPLB Network Queue Simulator (UNQS):**

**Analyzing Network Peformance**

**for Internet Bandwidth Management**

Leensey M. Lawas

## INTRODUCTION

With the prevalence of internet usage in this digital age, the rise of demand for fast and reliable execution of online services is inevitable. Whether it is for personal use, like video chatting with friends and family from abroad, or for commercial and business transactions, customers want to make sure their services are done efficiently without slowing down or timing out. Client requests are sent simultaneously that when the server responses (or traffic) are returned, the routers are unable to inspect long fields in Internet Protocol (IP) packet headers quickly and are unable to reassemble and segment packets fast enough, causing performance bottleneck (Pazos, Gerla, and Rigolio, 1999). A quick solution to the problem would be increasing bandwidth size, because as the demand for services increases, the bandwidth must also be increased (*Communication news*, 2001). However, this method is costly and inefficient, which is why traffic engineering (TE) takes place.

Awduche, Chiu, Elwalid, Widjaja, and Xiao of The Internet Society (2002) defined that internet traffic engineering deals with evaluating network performance and optimizing it. Bandwidth is the unit of measurement, usually in Kbps or Mbps, used to monitor network performance for quantifying how much information a communication channel can handle (Teach-ICT, n.d.).

**A. Background of the study**

Tong and Yang (2007) cited that there have been many studies on TE, but most of them dealt with route selection algorithms, and few tackled bandwidth management techniques. For this study, bandwidth management is the TE method chosen. Kanu, Kuyoro, Ogunlere, & Adegbenjo define bandwidth management as an optimization technique that helps differentiate the types of network traffic from each other and determine which client or service should be prioritized. In short, bandwidth management allocates the available bandwidth depending on network traffic and client/service priority.

In an article named *Bandwidth management pays off* (2002), two key devices were identified to help in bandwidth management: traffic shaping or congestion avoidance mechanisms and queueing techniques. *Congestion avoidance mechanisms* or *congestion control* locates where in the router the packets do not enter the system, and finds an alternative route so the packets do not block the way and cause timeout (Jacobson, 1988). *Queueing techniques*, on the other hand, help predict and direct the traffic flow by implementing a constraint or constraints to provide the services as demanded (Gross and Harris, 1974). In addition, queueing network models are known for accuracy and efficiency (Lazowska, Zahorjan, Graham, & Sevcik, 1984). By implementing these techniques, issues with cost and congestion will be dealt with while providing Quality of Service (QoS), thus increasing performance and reducing delay (Paul, 1999.).

There are three queueing techniques known to be applied in network scheduling. The simplest and most straightforward technique is the First-In First-Out (FIFO), also known as FCFS or First-Come First-Serve. Using a single buffer, FIFO schedules the packets based on arrival time and waits for the first traffic to finish processing before scheduling the next one (Islam Z., Islam S., Haque, & Moheuddin, 2012). Next is Priority Queue (PQ). PQ makes use of two queues, a priority and a wait

queue, and implements a set of rules known as a policy. Traffic that arrives will be sorted as either *priority* or *wait* using the policy, which is derived using traffic classification methods. The method can be as simple as using port numbers to assign priority, or as complex as statistical analysis to determine a policy. Once the policy is set, PQ algorithm will keep servicing the traffic in the *priority* and will only send the traffic *wait* if *priority* is empty. Rikli and Almogari (2012) noted the behavior of PQ will be similar to FIFO, only instead of arrival time, there is a different priority scheme. Last is the Weighted Fair Queue (WFQ), which introduces the concept of weights for fairness so that each flow is allocated a specific portion of the bandwidth to assure QoS (Dekeris, Adomkus, & Budnikas, 2006). In theory and in effect, each flow is given the chance to complete servicing its data the same rate as the other flows.

**B. Significance of the study**

Inefficient internet bandwidth management can lead to dissatisfied customers and reduced productivity. As long there is a need for "high quality of corporate customer satisfaction", bandwidth management will continue to grow as a body of knowledge (Duzbeck, 2006). This study simulated actual traffic data within the University of the Philippines Los Banos (UPLB) Network in order to determine whether the existing bandwidth is optimal or not. Additionally, the results can also be used for planning that can entail significant cost reductions, thus optimizing both bandwidth and budget to provide best quality service.

**C. Objectives of the study**

The general objective of the study is to efficiently simulate the UPLB network traffic by identifying the most optimal bandwidth setting. Specifically, the study intends to:

1. Collect traffic data from the UPLB network;
2. Simulate the traffic data using various bandwidth constraints;

3. Observe the result of the simulation by noting the duration, throughput, and flow loss for each constraint; and

4. Determine the most optimal network setting.

**D. Time and Place of the study**

The study was conducted from January 2017 until June 2018, at the Institute of Computer Science, UPLB.

**E. Scope and Limitations of the study**

The study was limited to monitoring and simulating a portion of the UPLB network. Also, it focused on the traditional queueing technique known as First In-First Out Queueing (FIFO).

In determining the most optimal bandwidth, the throughput, latency, and flow loss values was measured, noted, and compared. *Throughput* is the number of tasks accomplished over a period of time, *latency* is the time it takes for a fixed task to be finished, and *flow loss* is the percentage of dropped flows over the total number of flows.

## REVIEW OF RELATED LITERATURE

Several studies have been conducted which attempted to manage internet bandwidth as efficiently as possible. With a goal to provide speedy transaction of certain services such as e-mailing, video streaming, downloading, and many more, internet bandwidth management plays an important role not just for business and commerce applications, but as well as personal usage, for customer satisfaction and improved network performance. Because of the many details enumerated, the demand for internet bandwidth management has never been greater.

Internet Live Stats (2016) records that internet users from 2015 to 2016 had increased with an estimated 43.4% of the world population to 46.1%. It has been a dramatic increase since 1995, with users all over the world amounting to less than 1%. The implications of this statistics to the UPLB academe can also be applicable, as more students and workers enter the university to make use of the campus network. The increase entails a growth in demand for larger internet bandwidth. With a number of users sending multiple requests for different services with varying sizes, data traffic becomes congested. No end-user wants delays, slow downs, or timeouts, in accomplishing the services they requested. Instead, end-users want fast execution of their requests so they can proceed to doing other tasks.

To fix the problem, bandwidth management takes place. Instead of paying for an increase in bandwidth for a temporary fix, as mentioned in the article *Communication News (2001)*, bandwidth management aims to properly allocate the already existing bandwidth size as effectively and as efficiently as possible.

Before packets are received by the destination address, they first arrive in packet switches. These packet switches are in charge of queueing the packets and forwarding them eventually to the destination (Comer, 1999). Thus enters the scheduling algorithms used to identify which packets must be distributed first to the computers in the network.

An important aspect in network scheduling is the data that will be subjected to it. Jerkins and Wang (1999) emphasized the importance of characterizing these data in order to extend its applications to traffic management, and more specifically, for the purpose of this paper, bandwidth management.

Traffic classification is an initial stage that plays a key role to the scheduling algorithms, particularly for the PQ and WFQ algorithms. Because of bandwidth contraints, traffic classification helps in managing the fixed, limited, and available bandwidth. Classification can be payload-based, meaning a field of the payload is examined and used for classification (Cisco Systems Inc., 2008). An early traffic classification technique by Schneider (1996) made use of port numbers, which worked best for well-known or reserved ports. The other method for classification uses statistical analysis of traffic behavior (Cisco Systems Inc., 2008).

The simplest scheduling algorithm is the First-In First-Out (FIFO) queueing algorithm, wherein each introduced data is serviced based on their arrival time. A study by Mustafa and Talab (2016) showed that FIFO has a smaller queueing delay compared to Priority Queueing (PQ) and Weighted Fair Queueing (WFQ).

Karim, Nasser, Taleb, and Alqallaf (2012) proposed a priority packet scheduling algorithm that made use of three priority queues that gave importance to real-time traffic (priority 1) over non-real

time traffic (priorities 2 and 3). Its result suggested of a better performance opposed to FCFS and multi-level queue scheduler algorithms.

From the study of Muhilan, Arulselvi and Kiran Kumar (2013), packet scheduling algorithms using fair queueing and two additional variants were simulated to compare the delay. It showed that WFQ and Self Clock Fair Queueing (SCFQ) experienced a linear delay, whereas the Worst Case Weighted Fair Queueing (WF2Q) share the output link.

The aforementioned studies inspired and influenced this study, which used traffic data as input for the simulations of FIFO.

## METHODOLOGY

### A. Traffic Collection

With assistance from ITC, a mirror port was setup and connected to a 64-bit Ubuntu server named as *babage*. The traffic monitoring application, *ntopng*, was installed to the server. MySQL database management system was also installed, which shall contain the database where traffic flow data from *ntopng* will be dumped. To run *ntopng*, a configuration file needs to be set to identify the network interface(s) and network(s) to be monitored, the database and table to be dumped at, and the HTTP port where the web portal can be accessed. Data was collected from November 13 to 20. Using the *mysqldump* tool, the .sql file was generated for the researcher to have a copy of the database outside of the UPLB network. *ntopng* uses the Unix timestamp to label the arrival time of packets into the switch (known as FIRST_SWITCHED) and their exit time from switch to their destination hosts (LAST_SWITCHED). This timestamp counts the seconds that have passed since January 1, 1970 (Coordinated Universal Time/UTC).

The configuration file *ntopng.conf* looked like this:

```
--pid-path=/var/tmp/ntopng.pid
--daemon
--interface=eno1,eno2
--http-port=3000
--local-networks="10.0.0.0/8,172.16.0.0/16,202.92.144.0/22"
--dns-mode=1
--data-dir=/var/tmp/ntopng
--disable-autologout
--community
--hw-timestamp-mode=ixia
--user="admin"
--dump-flows="mysql;localhost;ntopng;flowsv4;root;[password]"
--dump-hosts=remote
```

***Figure 1. ntopng* configuration file.**

To run *ntopng* using the configuration file *ntopng.conf*,

```
ntopng "/etc/ntopng/ntopng.conf"
```

***Figure 2.*** **Run *ntopng* with configuration file.**

To save the data into a .sql file,

```
mysqldump -u [username] -p[password] [db_name] flowsv4 > flowsv4.sql
```

***Figure 3.*** **Dump data to sql file.**

Using *desc flowsv4* in the *mysql* command line interface, the structure of the table *flowsv4* is retrieved. Said information has been placed in Table 1.

**Table 1. Structure of database table flowsv4**

| FIELD | TYPE | NULL | KEY | DEFAULT | EXTRA |
|---|---|---|---|---|---|
| idx | Int(11) | NO | MUL | NULL | auto_increment |
| VLAN_ID | smallint(5) unsigned | YES | | NULL | |
| L7_PROTO | smallint(5) unsigned | YES | | NULL | |
| IP_SRC_ADDR | int(10) unsigned | YES | | NULL | |
| L4_SRC_PORT | smallint(5) unsigned | YES | | NULL | |
| IP_DST_ADDR | int(10) unsigned | YES | | NULL | |
| L4_DST_PORT | smallint(5) unsigned | YES | | NULL | |
| PROTOCOL | tinyint(3) unsigned | YES | | NULL | |
| IN_BYTES | int(10) unsigned | YES | | NULL | |
| OUT_BYTES | int(10) unsigned | YES | | NULL | |
| PACKETS | int(10) unsigned | YES | | NULL | |
| FIRST_SWITCHED | int(10) unsigned | YES | MUL | NULL | |
| LAST_SWITCHED | int(10) unsigned | YES | | NULL | |
| INFO | varchar(255) | YES | | NULL | |
| JSON | blob | YES | | NULL | |
| PROFILE | varchar(255) | YES | MUL | NULL | |
| NTOPNG_INSTANCE_NAME | varchar(256) | YES | MUL | NULL | |
| INTERFACE_ID | smallint(5) | YES | | NULL | |

The columns used in either the traffic description or simulation are as follows: IP_SRC_ADDR and IP_DST_ADDR to determine the two hosts that are interacting, L4_DST_PORT to have a rough idea of the network applications and services being accessed by the source host, IN_BYTES and PACKETS to count the size of the data being retrieved including the overhead, and FIRST_SWITCHED to identify the timestamp when a flow arrives for scheduling.

### B. Simulation Program

A class diagram (Figure 4) was constructed to get an overview of how UNQS was implemented in the Java programming language.

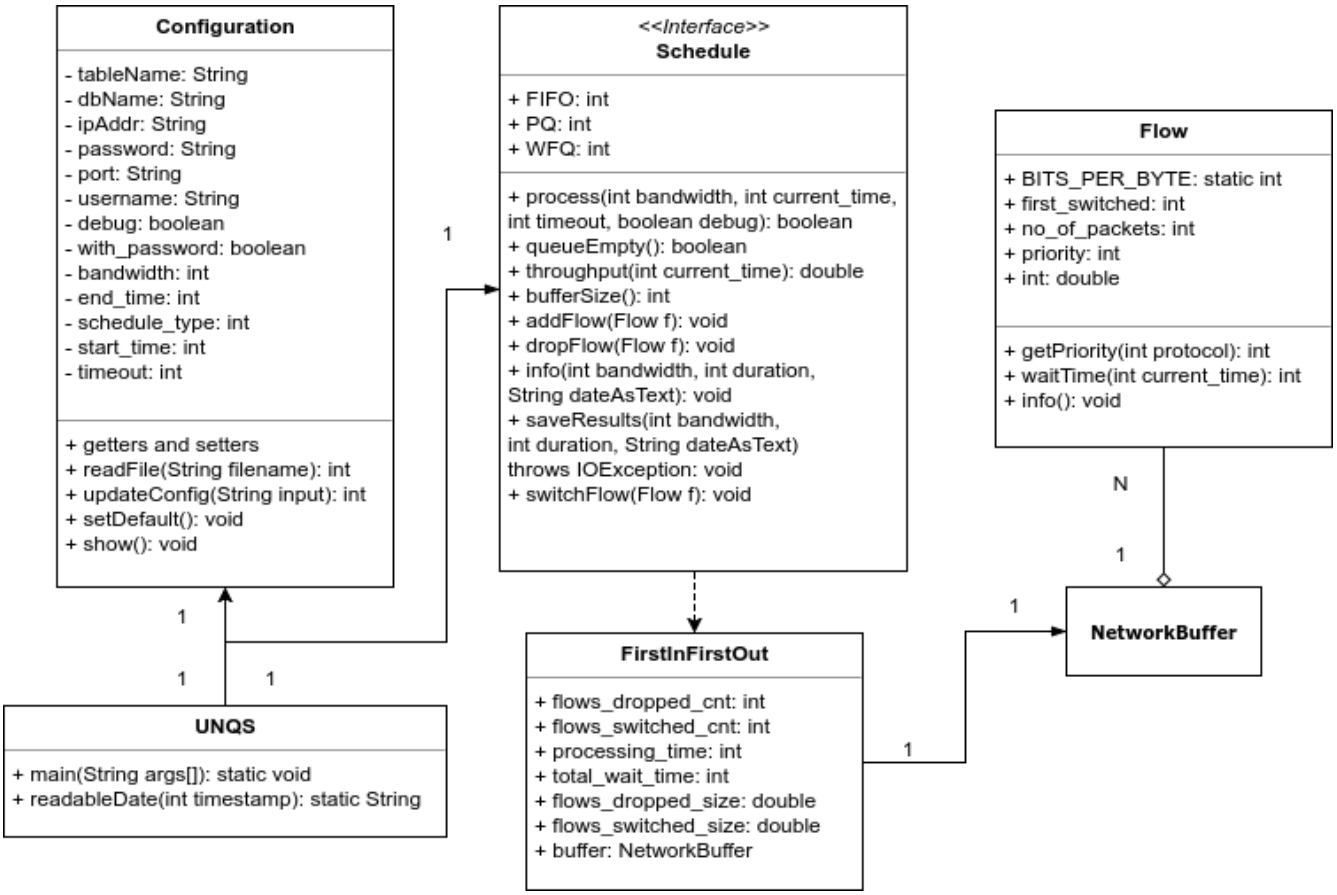


***Figure 4.*** **UNQS Class Diagram.**

For each instance of running the program, UNQS has one Configuration object and one Schedule object. FirstInFirstOut inherits the methods of the Schedule interface and uses one NetworkBuffer. That particular NetworkBuffer object can contain zero to many Flow objects.

Listed below are brief descriptions corresponding to a class or an interface.

1. **Cofiguration.java** is the class responsible for setting, updating, validating, and displaying the configuration for the database connection and simulation settings.
2. **Flow.java** is a class that represents a *flow*, which is a network link between a source host and a destination host. For the purpose of this program, the IP addresses of both hosts are not defined as this class' attribute.
3. **NetworkBuffer.java** is a class that defines the queue or cache that will contain all arriving flows that need to be switched.
4. **Schedule.java** is an interface that has final-static-defined variables FIFO (0), PQ (1), and WFQ (2). All of its methods are abstract and therefore must be overriden by the classes that will implement it.
5. **FirstInFirstOut.java** implements the Schedule interface and overrides its methods in order to perform FIFO's logic.
6. **UNQS.java** contains the main function where the connection to the database is established. Time is counted from the configuration's defined range of *start_time* to *end_time* and shall continue to iterate until all flows within the defined network buffer(s) are scheduled from the switch to their destination.

```
# information for the database connection
--username=root
--password
--ip-address=127.0.0.1
--port=3306
--database=FINAL
--table=flowsv4

# for the queue

# in bits per second
--bandwidth=600000000

# 0=FIFO, 1=PQ, 2=WFQ
--schedule=0

# in seconds
--timeout=60
--starttime=1510502646
--endtime=1511168758

# add if debugging
#--debug
```

***Figure 5.*** **UNQS configuration file.**

Focusing on the main program, the proceeding paragraphs detail the flow of the program logic.

Program begins with reading the configuration file (Figure 5), its format inspired by that of *ntopng*'s configuration file. The first block of information (*username*, *password* flag, *IP address*, *port*, *database name*, *table name*) is used collectively to connect to the mysql database. Meanwhile, the latter parts are used for setting the *bandwidth*, *schedule type*, *timeout* (or the TTL), the *start time* and the *end time* which bound the data set, and a *debug* flag.

After reading the program, the Schedule object is created, depending on the value that was set for the *schedule type* variable, *current time* is initialized with the value of *start time*, then the program proceeds to the processing. A *processing* flag is set to false.

While *current time* is less than or equal to the *end time*, check if there are flow/s at the current time and retrieve them from the database. If one or more flows are retrieved, Flow objects are created and added to the Schedule object's *buffer*. If *processing* is false, meaning, Schedule object's process method is available, then buffer is processed via the *schedule type*'s algorithm. In *process* method, *processing time* is computed. If its resulting value is greater than or equal to *timeout*, the flow is dropped and returns a value of *false* to set *processing* flag. If *processing time* is longer than 1 second, the flow is switched, thus setting *processing* flag's value to *true*. *Processing time* is decremented in order to continue processing for the next iterations. Increment current time and repeat while the condition is true; otherwise, proceed to the next batch of processing.

The main difference of the first loop to this second one is the initial condition, which for the latter is dependent on buffer's contents. As long as it is not empty, Schedule object's *process* method is invoked. The same logic described in the previous paragraph is then followed.

Once the buffer is emptied, the Schedule object will output the resulting network performance metrics to a file named *YYYY-MM-DD_bandwidth.txt* located in the subfolder *results.*

In summary, the simulation's main program logic is visualized in the flow chart in Figure 6.

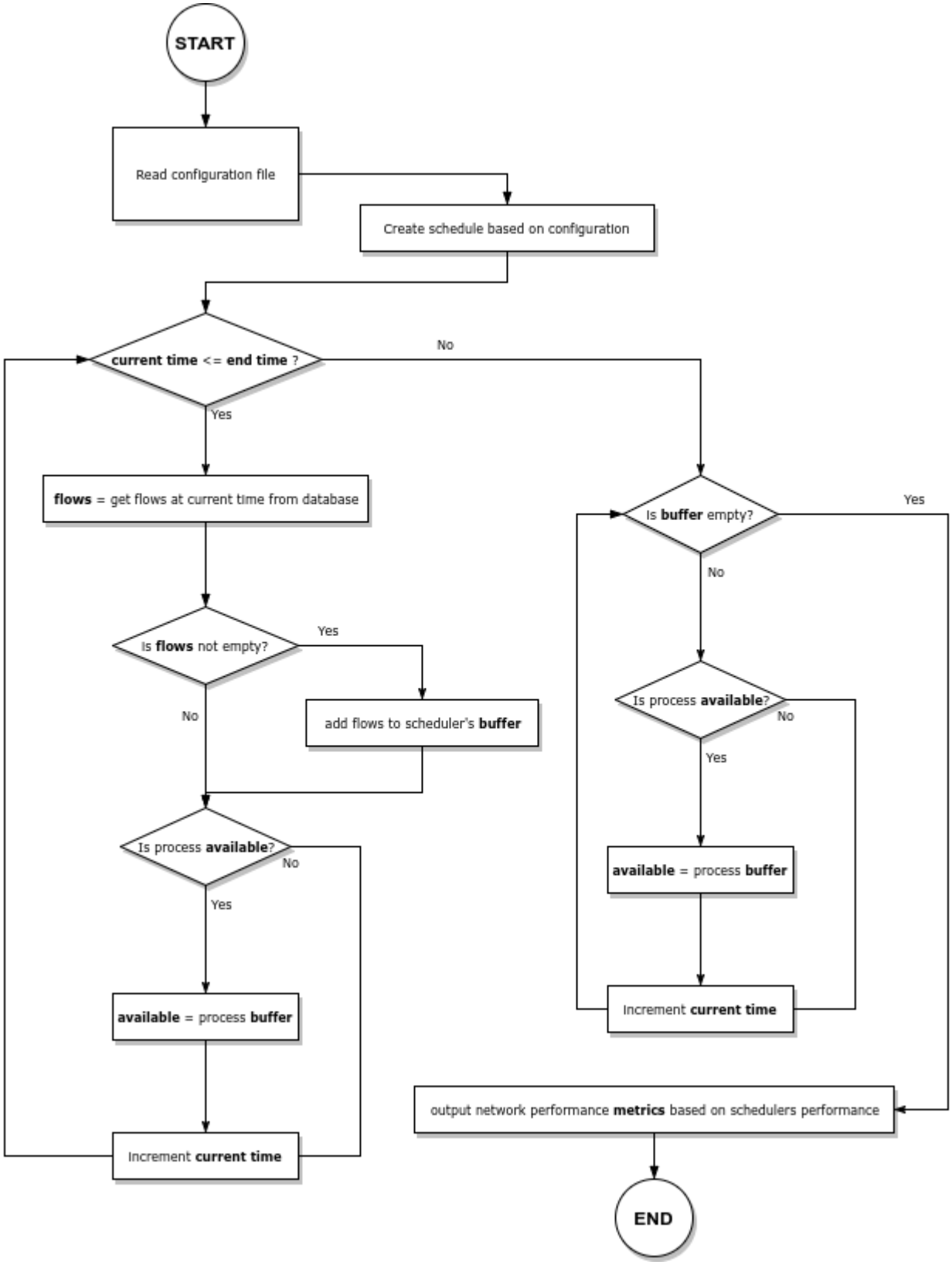


***Figure 6.*** **UNQS Main Program Logic.**

To compile the program,

```
javac -cp ".;mysql-connector-java.jar;" *.java
```

***Figure 7.*** **Compile UNQS.**

Given *conf* is a subfolder containing configuration file *simulation.conf,* run the program as

```
java -cp .:mysql-connector-java.jar UNQS conf/simulation.conf
```

***Figure 8.*** **Run UNQS.**

## RESULTS AND DISCUSSION

### A. Traffic Description

#### General traffic description

Network traffic was collected from November 13-20. There were a total of 554,392 flows, 432,616,056 packets, and 2,297,393,747,224 bits observed in a period of 7.71 days (computed as the difference between the maximum FIRST_SWITCHED minus the minimum FIRST_SWITCHED, both written in Unix timestamp). The overall bandwidth of the observed traffic is tabulated below.

**Table 2. Bandwidth per unit time**

| | per UNIT TIME | | |
|---|---|---|---|
| | **per DAY** | **per HOUR** | **per SECOND** |
| **BANDWIDTH (bits)** | 297,975,842,700.91 | 12,415,660,112.54 | 3,448,794.48 |

*note – divisor for per day = 7.71, for per hour = 7.71x24, for per second = 7.71x24x60x60

#### Most active source and destination IPs

The most active source IP address, 172.16.8.8 (Private), sent the most data, accounting for 5.4324% of the total flows while the IP address 239.255.255.250 (Multicast) received the most interactions, thus describing 9.9321% of the total flows.

#### Top Out-bound and In-bound flows

The top destination IP addresses for out-flow traffic are shown in Table 3 and the top source IP addresses for in-flow traffic are seen in Table 4.

**Table 3. Top 9 Out-Flows**

| DESTINATION (# of flows) | TOTAL BYTES | % TOTAL |
|---|---|---|
| KDDI CORPORATION (1) | 62,409,079,786 | 60.6 |
| MULTICAST (3) | 12,310,903,701 | 11.9 |
| Google LLC (3) | 5,202,456,530 | 5.0 |
| Facebook, Inc. (1) | 2,771,346,197 | 2.7 |
| Apple Inc. (1) | 1,054,547,100 | 1.0 |

**Table 4. Top 9 In-Flows**

| SOURCE (# of flows) | TOTAL BYTES | % TOTAL |
|---|---|---|
| KDDI CORPORATION(1) | 79,135,454,093 | 43.0 |
| Google LLC(3) | 27,958,817,762 | 15.2 |
| WorldStream B.V.(1) | 3,426,880,570 | 1.9 |
| M247 Ltd(2) | 4,286,472,241 | 2.3 |
| Converge ICT Solutions Inc.(2) | 3,967,550,023 | 2.2 |

Listed are brief descriptions for each company that is associated with the identified IP addresses:

1. **KDDI Corporation** is a telecommunications business based in Japan. It provides content hosting over optimized networks, and ICT and business services and solutions.
2. **Google LLC** is an American multinational technology company that specializes in Internet-related services and products such as online advertising technologies, search engine, cloud computing, software, and hardware.
3. **WorldStream B.V.** is a popular Internet Service Provider based in the Netherlands and is used by customers from all over the world. It provides cost-effective services to secure hosting environment, and offers hardware and Operating System technologies.

4. **M247 Ltd** is a UK-based technology company that offers services and tools to secure network and data while providing connectivity and internet infrastructure that expands to a global scale.
5. **Converge ICT** is a Philippine technology company with the fastest growing fiber internet and services offered to ensure pure end-to-end fiber internet connection, thus reducing data loss, increasing speed and bandwidth.
6. **Facebook** is an American online social media and social networking service company.
7. **Apple Inc.** is an American multinational technology company that designs, develops, and sells consumer electronics, computer software, and online services. Multicast IPs allow group communication to be sent simultaneously to multiple computers.
8. **Multicast** is not a company, but rather a form of data transmission that allows multiple hosts to receive data simultaneously.

**Top Applications**

The Internet Assigned Numbers Authority (IANA) is responsible for associating port numbers with certain internet protocols used by network applications. These identifications can be found in IANA's *Service Name and Transport Protocol Port Number Registry*. Table 5 listed the top destination port numbers as recorded in the data. Each port number has a corresponding application protocol, which are as follows:

1. **Hypertext Transfer Protocol Secure (HTTPS)** is responsible for securing communication over a network.
2. **Domain Name System (DNS)** helps match names to IP addresses and vice versa to facilitate network communications.
3. **Simple Service Discovery Protocol (SSDP)** is used to advertise presence information to locate available network services.

4. **Session Inititation Protocol (SIP)** allows signaling and controlling of multimedia communication sessions in VoIP applications like online voice and video calls.
5. **Microsoft-Directory Services (MS-DS)** follows the SMB protocol for shared access to files, printers, and serial ports and other communications between nodes on a network.
6. **Hypertext Transfer Protocol (HTTP)** is used to facilitate data communication for the World Wide Web.
7. **Link Local Multicast Name Resolution (LLMNR)** is a protocol based on DNS and allows both IPv4 and IPv6 hosts to perform name resolution for hosts on the same local links.
8. **0 (Reserved) and 7437 (Faximum)** have no specific protocols linked to them, and are likely abused ports to send computer attacks and harmful computer and network content like viruses.
9. **Simple Network Management Protocol (SNMP)** functions in collecting and organizing information about managed devices on IP networks, and can modify information to change device behavior.
10. **Bootstrap Protocol Server (BOOTP)** is exclusive for IPv4; Dynamic Host Configuration Protocol (DHCP) server receives requests upon booting the client's computer.

**Table 5. Top 10 Destination Ports**

| PORT NUMBER | PROTOCOL | TOTAL FLOWS | % TOTAL |
|---|---|---|---|
| 443 | HTTPS | 65,088 | 11.7 |
| 53 | DNS | 60,765 | 11.0 |
| 1900 | SSDP | 52,579 | 9.5 |
| 5060 | SIP | 43,101 | 7.8 |
| 445 | MS-DS | 28,746 | 5.2 |
| 80 | HTTP | 18,485 | 3.3 |
| 5355 | LLMNR | 17,090 | 3.1 |
| 0 | Reserved | 15,732 | 2.8 |
| 7437 | Faximum | 13,734 | 2.5 |
| 161 | SNMP | 9,142 | 1.6 |
| 67 | Boostrap Protocol Server | 8,602 | 1.6 |

## B. Network Performance Metrics

Three network performance metrics were accounted for as a result of the simulation, namely, duration, throughput, and flow loss. They are described in the items listed below.

1. **Duration** *d* is a function of the arrival time $t_0$ seconds minus the last switched time $t_n$ seconds.

$$d(t_0 , t_n ) = t_n - t_0$$

***Figure 9.*** **Formula for duration.**

2. **Throughput** *tput* was computed as

$$tput = \frac{count(t_S)}{d}$$

***Figure 10.*** **Formula for throughput.**

where $t_S$ is the total size (in bits) that was successfully switched, and *d* as the computed duration.

3. **Flow loss** is measured by the total number of dropped flows all over the total number of both dropped ($f_l$) and switched ($f_S$) flows, then multiplied by 100 to get the percentage.

$$l = \frac{count(f_l)}{count(f_l + f_S)} \times 100$$

***Figure 11.*** **Formula for flow loss.**

### C. Simulation Results

Simulation was run for the bandwidth settings 32.5 Mbps, 35.0 Mbps, 37.5 Mbps, 40 Mbps, and the timeout was set as a constant of 60 seconds.

Table 6 contains a summary of the simulation using the network performance metrics as defined in the previous section. It is observed that as the bandwidth is increased, the flows being dropped reduced while the flows being switched increased and became flat. To visualize this, the total size per bandwidth was graphed in Figure 12. Aside from the information concerning the flows, the simulation results showed that the througput and duration values across the four bandwidth constraints have significantly small difference and thus have similar performance.

**Table 6. Simulation results per bandwidth**

| | **BANDWIDTH (Mbps)** | | | |
|---|---|---|---|---|
| **METRICS** | **32.5** | **35.0** | **37.5** | **40.0** |
| **flows_dropped_size (Gb)** | 72.59 | 18.87 | 0 | 0 |
| **flows_dropped_cnt** | 36 | 9 | 0 | 0 |
| **flows_switched_size (Gb)** | 866.46 | 920.18 | 939.05 | 939.05 |
| **flows_switched_cnt** | 554,356 | 554,383 | 554,392 | 554,392 |
| **duration (days)** | 8.22 | 8.22 | 8.22 | 8.22 |
| **throughput (Mbps)** | 1.2 | 1.3 | 1.3 | 1.3 |

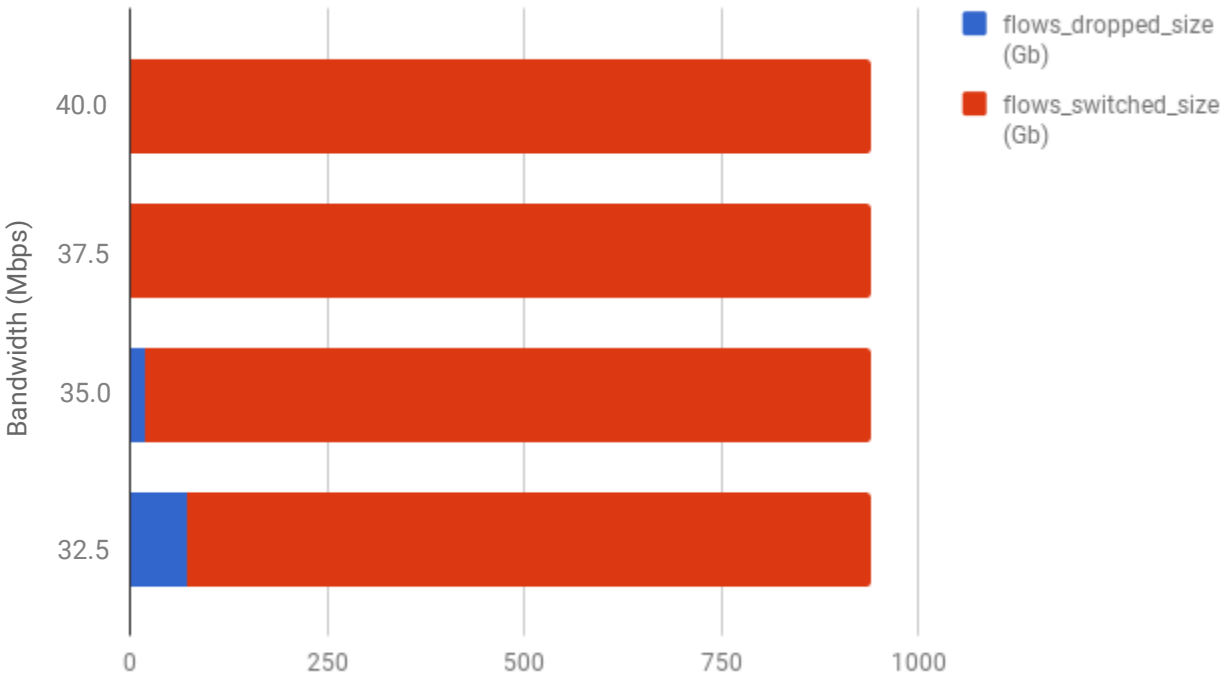


***Figure 12.*** **Observed dropped flows for different bandwidth settings**

## CONCLUSIONS AND RECOMMENDATIONS

Traffic data that was collected using *ntopng* was successfully processed using the FIFO scheduling algorithm as simulated by UNQS. The output performance metrics showed the effect of increasing the bandwidth in order to switch all the flows. It was observed that the actual traffic data had a throughput of 3.45 Mbps. This value was used as a starting point to find out the probable advised bandwidth setting. After several attempts in varying the bandwidth constraint with intervals of 500 Kbps, the shift in the network performance metrics was observed between 35.0 Mbps and 37.5 Mbps. While there were dropped flows under bandwidth constraint 35 Mbps, there was none under a bandwidth of 37.5 Mbps. The results for 32.5 Mbps and 40.0 Mbps were included to have an additional data for a better view of the emerging pattern. Therefore it is concluded that with the existing traffic data, a bandwidth of 37.5 Mbps is recommended as the most optimal.

Recommendations based on the analysis of the traffic data that was collected are listed below.

1. Block ports that are still not tagged by the firewall, like ports 0 and 7437, which have potential to receive harmful traffic data.
2. Maintain and use an internal DNS server to further reduce external DNS traffic.
3. Turn off MS-DS traffic that consumes a large amount of the out-bound bandwidth.
4. Mirror Linux, Microsoft, and Apple traffic to monitor possible intrusions and unnecessary traffic within the network in order to formulate and enforce policies to block and limit such traffic.

Future works may explore more complex scheduling algorithms aside from the native FIFO queueing algorithm like PQ and WFQ in order to determine more optimal scheduling techniques that will minimize the bandwidth constraint and save expenses. Additionally, data could be collected over a

longer time period to discover a pattern in network traffic behavior depending on certain periods of time, such as peak season during enrollment period, etc.. Doing so could give a better picture of the maximum bandwidth usage and thus determine the optimal, cost-efficient, and effective bandwidth setting, which can be paired with a queueing technique for further cost reductions and improved performance.

## BIBLIOGRAPHY


Awduche, D., Chiu, A., Elwalid, A., and Xiao, X. (2002). Overview and principles of internet traffic engineering, *The Internet Society*.

Azmi, Z.R.M., et al (2011). Performance Comparison of Priority Rule Scheduling Algorithms Using Different Inter Arrival Time Jobs in Grid Envorionment, *International Journal of Grid and Distributed Computing,* 4(3), 61-70. Retrieved from https://pdfs.semanticscholar.org/df51/1b75b80ddcd9a11d400d8a201593e50e902d.pdf

Cisco Systems Inc. (2008). *Wan and Application Optimization Solution Guide*. [PDF document]. Retrieved from http://www.cisco.com/c/dam/en/us/td/docs/nsite/wan_optimization/WANoptSolutionGd.

Communication News. (2001). Bandwidth management pays off*, Communication News,* 38(11), 54-56.

Comer, D. (1999). *Computer Networks and Internets.* New Jersey: Prentice-Hall International, Inc.

Dekeris, B., Adomkus, T. and Budnikas, A. (2006) Analysis of QoS Assurance Using Weighted Fair Queueing (WFQ) Scheduling Discipline with Low Latency Queue (LLQ), *28th International Conference on Information Technology Interfaces, 2006*, Cavtat/Dubrovnik, Croatia.

Duzbeck, F. (2006). Bandwidth management is here to stay, *Network World,* 23(13), 37. Retrieved from http://search.proquest.com/docview/215977831?accountid=47253

Goyal, P. (1997). *Packet Scheduling Algorithms for Integrated Services Networks (Diploma thesis)*. [PDF document].

Gross, D., and Harris, C.M. (1974). *Fundamentals of queueing theory*. New Jersey: Wiley.

Internet Live Stats. (2016). *Number of internet users.* Retrieved from http://www.internetlivestats.com/internet-users/

Islam, Z., Islam, S., Haque, F., and Moheuddin, A. (2012). A comparative analysis of different real time applications over various queuing techniques, *International Conference on Informatics, Electronics & Vision*, 2012, Dhaka, Bangladesh, New Jersey: IEEE.

Jacobson, V. (1988). Congestion avoidance and control, *ACM SIGNCOMM Symposium 1988.*

Jerkins, J.L. and Wang, J.L. (1999). From network measurement collection to traffic peformance modeling: challenges and lessons learned, *Journal of the Brazilian Computer Society,* 5(3). Retrieved from http://dx.doi.org/10.1590/S0104-65001999000100003

Kanu, R., Kuyoro, S., Ogunlere, S., and Adegbenjo, A. (2012). Management and control of bandwidth in computer networks, *International Journal of Computer Networks and Wireless Communications*. 2(3). 342-348.

Karim, L., Nasser, N., Taleb, T., and Alqallaf, A. (2012). An efficient priority packet scheduling algorithm for wireless sensor network, *IEEE ICC 2012 Ad-hoc and Sensor Networking Symposium*. 335-338.

Lazowska, E., Zahorjan, J., Graham, G., and Sevcik, K. (1984). *Quantitative system performance: Computer system analysis using queueing network models.* New Jersey: Prentice-Hall, Inc.

Martin, M. and Roth, A. (n.d.). *Cis 501: Computer architecture*. [PDF document]. Rerieved from https://www.cis.upenn.edu/ milom/cis501-Fall11/lectures/02_performance.pdf

Muhilan, R., Arulselvi, S., and Kiran Kumar, T. (2013). Packet scheduling algorithms simular WFQ, SCFQ, and WF2Q, *International Journal of Scientific & Technology Research*. 2(10). 249-252.

Mustafa, E.G.M. and Talab, S.A. (2016). The Effect of Queueing Mechanisms First in First out (FIFO), Priority Queueing (PQ) and Weighted Fair Queueing (WFQ) on Network's Routers and Applications, *Wireless Sensor Network*, 8(5), 78-84. Retrieved from https://file.scirp.org/pdf/WSN_2016053115445482.pdf

Paul, A. (1999). QoS in Data Networks: Protocols and Standards, *Recent Advances in Networking 1999*, St. Louis, Missouri.

Pazos, C.M., Gerla, M., and Rigolio, G. (1999). Flow control and bandwidth management in next generation internets, *Telecommunication Systems.* 11(304). 394-412. Retrieved from http://search.proquest.com/docview/212047202?accountid=47253

Rikli, N-E. and Almogari, S. (2012). Efficient priority schemes for the provision of end-to-end quality of service for multimedia traffic over MPLS VPN networks, *Journal of King Saud University*, 25, 89-98. Riyadh: King Saud University.

Schneider, P. (1996). *TCP/IP traffic classification based on port numbers (Diploma thesis)*. [PDF document]. Retrieved from http://www.schneider-grin.ch/media/pdf/diploma_thesis.pdf

Teach-ICT. (n.d.). *Bandwidth*. Retireved from http://www.teach-ict.com/as_a2_ict_new/ocr/A2_G063/333_networks_coms/bandwidth/miniweb/pg2.htm

Tong, S., and Yang, O. (2007). Bandwidth management for supporting differentiated-service aware traffic engineering, *Parallel and Distributed Systems*. 18(8). 1320-1331.