UPLB Network Queue Simulator (UNQS): Surveying Network Performance For Internet Bandwidth Management

Leensey M. Lawas and Danilo J. Mercado

***Abstract*— Given the increasing demand for bandwidth, UNQS was developed to survey and to identify the optimal bandwidth setting as measured by duration (seconds), throughput (bits per second), and flow loss (percentage). The results show that while the simplest, FIFO showed the best performance with minimal duration, maximal throughput, and minimal packet loss.**

***Index Terms*— bandwidth management, internet, network performance, network simulation, queueing, traffic engineering**

I. 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 [1]. A quick solution to the problem would be increasing bandwidth size, because as the demand for services increases, the bandwidth must also be increased [2]. 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) [3] 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 [4].

A. Background of the study

Tong and Yang (2007) [5] 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 [6] 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) [2], 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 [7]. *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 [8]). In addition, queueing network models are known for accuracy and efficiency [9].

B. Significance of the study

Inefficient internet bandwidth management can lead to dissatisfied customers and reduced productivity. As long there is a need for âĂIJhigh quality of corporate customer satisfactionâĂİ, bandwidth management will continue to grow as a body of knowledge [10]. 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 future planning that can entail significant cost reductions, thus optimizing both bandwidth and budget to provide 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 was able to:

1) Collect traffic data from the UPLB network;
2) Simulate the traffic data using various bandwidth sizes;
3) Take note of the duration, throughput, and flow loss for each simulation; and
4) Determine the most optimal network setting using graphs and simple statistics.

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

D. Time and Place of the study

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

E. Scope and Limitation of the study

The study is 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 will be 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 [11], and *flow loss* is the percentage of dropped flows over the total number of flows.

II. 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 users from 2015 to 2016 had increased with an estimated 43.4% of the world population to 46.1% [12]. It has been a dramatic increase since 1995, with users all over the world amounting 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 [2], 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 [13]. Thus enters the scheduling algorithms used to identify which packets must be distrbuted first to the computers in the network.

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 [14, Chapter 5]. An early traffic classification technique [15] made use of port numbers, which worked best for well-known or reserved ports. The other method for classifcation uses statistical analysis of traffica behavior [14, Chapter 5].

A proposed priority packet scheduling algorithm [16] 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 a different study, 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 [17].

The aforementioned studies inspired and influenced this study, which shall use traffic data classified by protocol as input for the simulations of FIFO, PQ, and WFQ scheduling algorithms.

III. 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 `ntong` 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 on April 17-18, 25, 28-29, and using the `mysqldump` tool, the .sql file was generated, which was used for the researcher to have a copy of the database outside of the UPLB network.

`ntopng` uses ticks to measure the arrival of packets into the switch and their exit from switch to their destination. There are 66 million ticks per second [18], which means 1 tick is equal to 0.000000015 seconds. This value in seconds was multiplied to the tick value that `ntopng` collected.

B. Simulation

1) Classes and their attributes:

1) **Cofiguration.java** - This class is responsible for setting, updating, validating, and displaying the configuration for the database connection (datestamp *MM-DD-YY* and interface number *i* for table name written as *flows-MM-DD-YYv4_i*, database name, IP address *xxx.xxx.xxx.xxx*, password, port number *p*, database username, database password) and queue settings (bandwidth *b*, maximum buffer size *m*, Maximum Transmition Unit *mtu_size*, queue type *qt*.
2) **IPProtocol.java** - This class contains static variables to identify the protocols used for categorizing the priority of each packet into three queues. High priority queue have Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) packets. Low priority packets

use the following protocols: Internet Control Message Protocol (ICMP) and Host Identity Protocol. Internet Group Message Protocol (IGMP) and other protocols are classified into the normal queue.

3) **Main.java** - This contains the main function where the connection to database is established, and where time is looped starting from the minimum `FIRST_SWITCHED` value (taken from the selected table and measured in ticks) until all packets are scheduled from packet switch to their destination.
4) **Packet.java** - A packet instance contains the following attributes: id, priority, protocol, size (in bytes), arrival time and send time to and from the scheduler (measured in ticks), and virtual time *vt* (used only for WFQ). Virtual time is computed as

$$vt = \frac{s}{w}$$

where *s* is the packet size and *w* is the weight of a queue. Queue weight's value is derived as

$$w = d \times m$$

where *d* is the fixed distribution rate for each priority (0.5, 0.3, and 0.2 for high, medium, and low priorities respectively), and *m* is the maximum buffer size.
5) **Queue.java** - Queue takes note of the current buffer size, packet count, total bytes transmitted, and maximum buffer size.
6) **Schedule.java** - Schedule has static-defined variables *FIFO* (0), *PQ* (1), and *WFQ* (2). Its attributes are the schedule number and the total number of bytes transmitted.

2) Code snippet of the queue processing: The code snippet in Figure III-B.2 shows how the queues are being processed depending on the configured queue type. This method is called on per every iteration of time *t*. The loop will stop once all the packets in the queues have been serviced.

IV. Results and Discussion

A. Description of Data per day

*1) **April 17, 2017**:* The data collected on the first day spanned 28,823 ticks, garnering a total of 31,259 flows.

The top 3 source ports are as follows: port 0 with 4448 flows, port 5353 with 2247 flows, and port 137 with 1097 flows. Meanwhile, the top 3 destination ports are port 1900 (8528 flows), port 8000 (6153 flows), and port 0 (4448 flows).

The top 3 source IP addresses were 10.0.53.67 (4399 flows), 10.0.3.110 (3769 flows), and a tie for 10.0.53.53 and 10.0.3.148 (1764 flows). As for the destination IP addresses, 239.255.255.250 (9523), 10.0.3.53 (6174), and 224.0.0.252 (4088) ranked highest among other IP addresses.

*2) **April 18, 2017**:* For the second day, the data collection process took a total of 14,392 ticks, and recorded 20,574 flows.

The 3 most used source ports are ports 0, 5353, and 137, which were found in 2372, 1237, and 655 flows respectively. On the other hand, the top 3 frequented destination ports are noted to be port 8000 with 5439 flows, port 1900 with 5048 flows, and port 5355 with 2523 flows.

```
public boolean process(LinkedList<Queue>
    queues){
boolean success = false;

// process in order of arrival
if(schedule==Schedule.FIFO){
        Queue q = queues.remove();
        if(q.isEmpty()) return true;
        success = q.process(schedule);
        queues.add(0, q);
        return success;
}

// process from highest to lowest
    priority
else if(schedule == Schedule.PQ){
        for(int i=2; i>=0; i--){
        if(!queues.get(i).isEmpty()){
                Queue q =
                    queues.remove(i);
                success =
                    q.process(schedule);
                queues.add(i, q);
                return success;
        }}
}

// process the smallest virtual time
else if(schedule == Schedule.WFQ){
        int pri = 0,
                smallest_virtual_time =
                    2147483647;
        Queue q;

        for(int i=2; i>=0; i--){
                q = queues.remove(i);
                if(!q.isEmpty()){
                if( q.isEmpty() )
                    continue;
                else
                    if(q.peek().virtual_time
                < smallest_virtual_time){
                smallest_virtual_time =
                    q.peek().virtual_time;
                pri = i;
                }}
                queues.add(i, q);
        }
        q = queues.remove(pri);
        success = q.process(schedule);
        queues.add(pri, q);
}

return success; // true = nothing was
    processed or queue is empty. false =
    queue is not empty
}
```

Fig. 1. Code Snippet of *process* method in `Schedule.java`

The 3 highest talkers for the day were IP addresses 10.0.53.53 (5444 flows), 10.0.3.110 (1825 flows), and 10.0.3.160 (1052 flows). Looking over to the receiving end, the top three destination IP addresses were 239.255.255.250 (5748 flows), 10.0.3.53 (5451), and 224.0.0.252 (3068).

*3) **April 25, 2017:*** The third day when the data was collected had a duration of 2498 ticks and a total of 3,892 flows, both of which are the shortest time span and smallest count of flows for the entire data set.

The top 3 source ports were port 0 (433), port 5353 (281), and port 138 (134). Destination ports most actively used were ports 8000 (1344 flows), 1900 (890), and 0 (433).

Hosts that sent most packets through the network came from IP addresses 10.0.53.177 (1359), 10.0.3.110 (319), and 10.0.3.89 (168). Three IP addresses that received the most requests that day were 10.0.3.53 (1359), 239.255.255.250 (991), and 224.0.0.252 (365).

*4) **April 28, 2017:*** Data gathered for the fourth day lasted for a length of 11,812 ticks and observed a total of 11,483 flows in the network.

Ports that were used to send out the most interaction are ports 0 (2220), 5353 (897), and 138 (532). Ports that received the most interaction are ports 1900 (3789), 0 (2220), and 5355 (1473).

The most active source IP address is 10.0.3.110 (1529), followed by 10.0.3.148 (817), and 10.0.3.89 (689). On the other hand, the most active destination IP addresses are 239.255.255.250 (4312), 224.0.0.252 (1959), and 224.0.0.251 (1423).

*5) **April 29, 2017:*** The fifth and final data collection counted 14,027 flows within a time frame of 7183 ticks.

Topping as a source port is port 0 with 1197 flows, followed by port 5353 at 600 flows, and lastly, port 138 with 281 flows. The three highest ranking destination ports are as follows: port 8000 counted 7924 flows; port 1900 had 2253; and port 0 registered 1197 flows.

Three source nodes most notably active were of IP addresses 10.0.53.9 (7967), 10.0.3.110 (932), and 10.0.3.148 (369). On the receiving end, the following are the top 3 IP addresses: 10.0.3.53 (7970), 239.255.255.250 (2525), and 224.0.0.251 (861).

~~In summary, the observed frequented applications being~~ used can be seen by the ports used most. Port 8000 is used by the Hyper Text Transfer Protocol, port 1900 is used by the Simple Service Discovery Protocol, port 0 is the initial 'binding' port before the operating system returns a new value, which is an available port, and port 5355 is used for Link-Local Multicast Name Resolution.

B. Metrics Considered in the Data Analysis

The following figures show bar graphs of the average network performance metrics to be looked at: duration, throughput, and packet loss. The dates per collection, from left to right, are April 17, 18, 25, 28, and 29. The data that spawned these results were collected 4 hours per date. For each scheduling algorithm, ten varying buffer sizes were tested, from 10MB, 20MB...until 100MB. The averages for each network performance metric were collected and visualized in the figures.

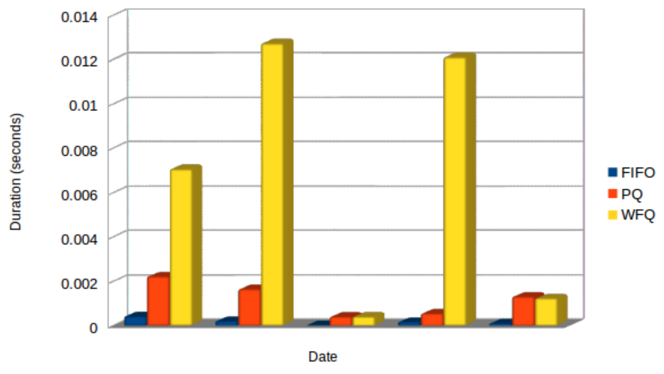

Fig. 2. Average Durations per Date and Queue Type

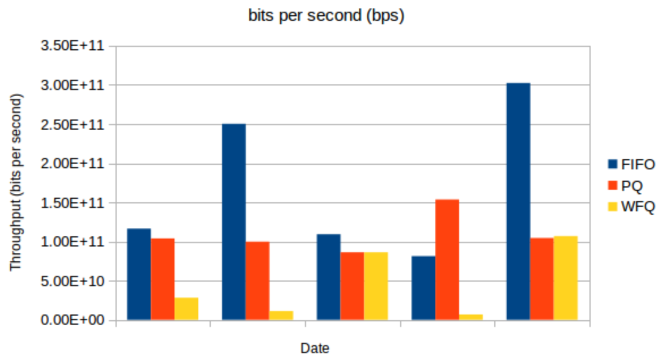

Fig. 3. Average Throughputs per Date and Queue Type

1) Duration: It is evident from IV-B.1 that in terms of how fast the packets are scheduled, `WFQ` takes the most time to finish in scheduling the packets. Duration d is a function of the arrival time t_0 and last switched time t_n seen as follows:

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

2) Throughput: Throughput *tput* was computed as

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

where *ts* is the total size (in bytes) that was successfully switched, and d as the computed duration. Focusing on IV-B.2, `FIFO` scheduling queue outputted the most throughput.

3) Packet Loss: The last network performance parametric looked at is the packet loss l, solved by

$$l = \frac{count(p_l)}{count(p_l + p_s)} \times 100$$

where all dropped packets p_l are counted, divided by the total of dropped p_l and switched packets p_s multiplied by 100, to get the percentage. Similarly, the results show that `WFQ` has the worst performance when packet loss is considered. Notably, `PQ` shows lower packet loss compared to `FIFO`.

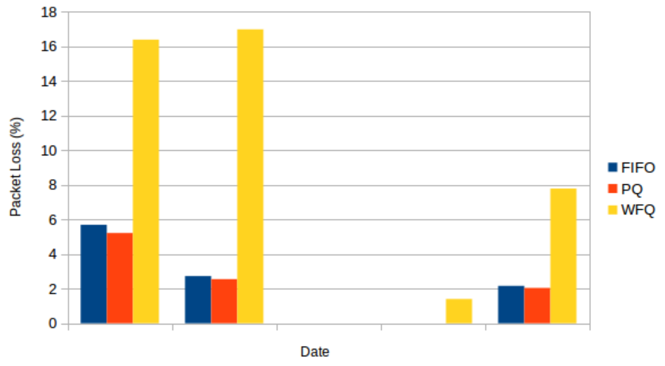

Fig. 4. Average Packet Losses per Date and Queue Type

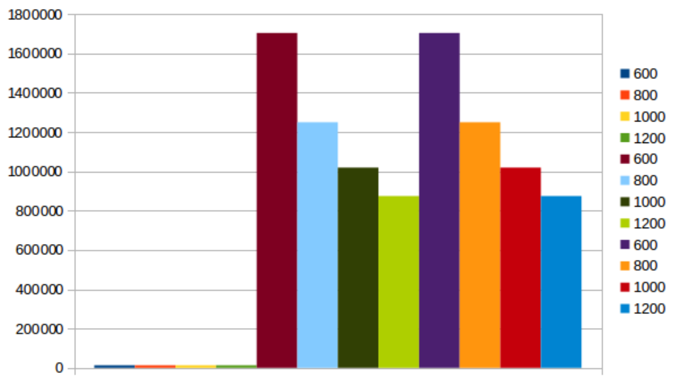

Fig. 5. Average Durations with Applied Bandwidth Constraint

4) Bandwidth Constraint: In attempt to check if the same trend is observed when a bandwidth constraint is added, the following bandwidth constraints were tested using a 100MBps buffer: 600Mbps, 800Mbps, 1000Mbps, and 1200Mbps. These values were converted into their byte form and with help from the conversion value for seconds to ticks, they were translated into 1.125 Bytes per tick (Bpt), 1.5Bpt, 1.875Bpt, and 2.25Bpt respectively. The result is visualized in IV-B.4. The throughputs noted for all ran simulations are very close to the bandwidth, and a 100MB buffer provided 0% packet loss, hence only the duration was left as basis.

As seen, PQ and WFQ amounted to the same duration that showed improvement per increase of the bandwidth. Yet, it is apparent that FIFO's performance is a lot better based on this network performance metric. FIFO, in any of the bandwidth constraints, displayed minimal and fast speed of switching the packets, while maintaining a similar throughput to the slow-performing PQ and WFQ.

V. Conclusion and Recommendation

After the experiment, the results show that the UPLB network works better using the `FIFO` scheduling algorithm. It was able to output the most number of packets over a period of time, as shown by IV-B.2, and switched the packets fastly IV-B.1 while maintaining the most integrity by minimizing lost packetsIV-B.3.

While `PQ` and `WFQ` algorithms were intended to improve the network performance in terms of packet switching, this study proves that the traffic of UPLB is more suited to a `FIFO` scheduling algorithm.

For future studies, the researcher would recommend that a different traffic classification method is to be used, such as statistical methods or machine learning methods. Another advisable method is to inspect a different field of the payload or packet that will be used to classify traffic, instead of the protocol field. A final recommendation is the use of hybrid or variation of the PQ and WFQ scheduling algorithms to further improve the switching, thus effectively managing UPLB's internet bandwidth.

References

[1] C. M. Pazos, M. Gerla, and G. Rigolio, "Flow control and bandwidth management in next generation internets," *Telecommunication Systems*, vol. 11, no. 304, pp. 394–412, 1999. [Online]. Available: http://search.proquest.com/docview/212047202?accountid=47253

[2] C. News, "Bandwidth management pays off," *Communication News*, vol. 38, no. 11, p. 54âĂŞ56, Nov 2001.

[3] D. Awduche, A. Chiu, A. Elwalid, and X. Xiao, "Overview and principles of internet traffic engineering," *The Internet Society*, 2002.

[4] Teach-ICT, "Bandwidth." [Online]. Available: http://www.teach-ict.com/as_a2_ict_new/ocr/A2_G063/333_networks_coms/bandwidth/miniweb/pg2.htm

[5] S. Tong and O. Yang, "Bandwidth management for supporting differentiated- service aware traffic engineering," *Parallel and Distrubuted Systems*, vol. 18, no. 9, pp. 1320–1331, 2007.

[6] R. Kanu, S. Kuyoro, S. Ogunlere, and A. Adegbenjo, "Management and control of bandwidth in computer networks," *International Journal of Computer Networks and Wireless Communications*, vol. 2, no. 3, pp. 342–348, 2012.

[7] V. Jacobson, "Congestion avoidance and control," *ACM SIGCOMM Symposium 1988*, 1988.

[8] D. Gross and C. M. Harris, *Fundamentals of queueing theory*. Wiley, 1974.

[9] E. Lazowska, J. Zahorjan, G. Graham, and K. Sevcik, *Quantitative system performance: Computer system analysis using queueing network models*. Prentice-Hall, Inc., 1984.

[10] F. Duzbeck, "Bandwidth management is here to stay," *Network World*, vol. 23, no. 13, p. 37, 2006. [Online]. Available: http://search.proquest.com/docview/215977831?accountid=47253

[11] M. Martin and A. Roth, "Cis 501: Computer architecture [pdf document]." [Online]. Available: https://www.cis.upenn.edu/ milom/cis501-Fall11/lectures/02_performance.pdf

[12] I. L. Stats, "Number of internet users," 2016. [Online]. Available: http://www.internetlivestats.com/internet-users/

[13] D. Comer, *Computer Networks and Internets*. Prentice-Hall International, Inc., 1999.

[14] *WAN and Application Optimization Solution Guide [PDF document]*. Cisco Systems Inc., 2008. [Online]. Available: http://www.cisco.com/c/dam/en/us/td/docs/nsite/wan_optimization/WANoptSolutionGd.

[15] P. Schneider, "Tcp/ip traffic classification based on port numbers (diploma thesis) [pdf document]," 1996. [Online]. Available: http://www.schneider-grin.ch/media/pdf/diploma_thesis.pdf

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

[17] R. Muhilan, S. Arulselvi, and T. Kiran Kumar, "Packet scheduling algorithms simulator wfq, scfq and wf2q," *International Journal of Scientific & Technology Research*, vol. 2, no. 10, pp. 249–252, Oct 2013.

[18] webopedia, "Clock tick," 2017. [Online]. Available: http://www.webopedia.com/TERM/C/clock_tick.html

Leensey M. Lawas She is a BS Computer Science undergraduate student. She not only writes code, but also songs, poems, and stories.