



Token Bucket in Traffic Shaping Using the Multi-Threading Based Preemption

¹Sasikala. T M, ²Vijimichiko M, ³R Charanya

^{1,2} Student, SITE, VIT University, Vellore, Tamilnadu, India

³ SITE, VIT University, Vellore, Tamilnadu, India

Abstract— This project involved emulation of the Token Bucket algorithm using POSIX threads in C. The aim was to simulate a traffic shaper that receives and transmits packets to a server, while being controlled by a token bucket filter. There were three major components of the system are given: 1. the input queue that received the packets 2. the token bucket that received the tokens 3. The output queue that send the packets to the Server. The input queue would receive the packets at a fixed rate, store them on First-Come-First-Serve (FCFS) basis and would then wait for the token bucket to fill up with the needed tokens. Once the token bucket has enough tokens for one packet, it would transfer that packet to output queue and would remove the tokens used for that packet from the token bucket. The server would read the packets from the output queue at its own fixed rate as they get accumulated in the output queue. Each of the queues and the token bucket had their own rate of processing the packets and the tokens, while the token bucket had a fixed capacity to store the tokens without overflowing them. A system with such a token bucket filter can be depicted.

Keywords— Token bucket using Multi-threading, First-Come-First-Serve, Quality of Service(QoS)

I. INTRODUCTION

The term congestion defines that the load on the network is greater than capacity of the network. Whereas Congestion control is the mechanisms to control the congestion and keep the load below the capacity. Congestion occurs because routers and switches have queues- buffers that hold the packets before and after processing. If the rate of packet arrival is greater than the packet processing time that leads to the input queue longer. And if the packet departure time is less than the packet processing time that leads to the output queue longer. The aim of the congestion control mechanism is to either prevent the congestion before happens or to remove the congestion after it happened in the processor. Thus it is defined in two broad categories such as prevention and removal. Some of Quality of Service(QoS) techniques and mechanisms are used here is traffic shaping using the token bucket. Traffic shaping is used to control the rate and amount of traffic sent to the networks. In the Token Bucket(TB) algorithm, the bucket holds tokens, it used to transmit a packet, the host should capture and destroy the token one by one. Tokens are generated at the rate of one token every t sec, Idle hosts will capture and save up tokens (up to the max. size of the bucket) in order to send larger bursts later. TB accumulates fixed size tokens in a token bucket. Transmits a packet (from data buffer, if any are there) or arriving packet, if the sum of the token sizes in the bucket add up to packet size. More tokens are periodically added to the bucket (at rate t). If tokens are to be added when the bucket is full, they are discarded. Thus the Multi-Threading concept is used here i.e it gets input only once based on that it generates the output automatically until it is stopped by user. The process of multithreading is shown in fig 1.

II. RELATED WORK

A. QoS in LINUX

The some of QoS flow characteristics are: *jitter, delay, reliability and bandwidth*. The interface in QoS rearranges entering packets to the queue in with the previously configured rules. The techniques used are Traffic shaping, scheduling, resource reservation, Admission control and FIFO Queuing.

1) FIFO Queuing

FIFO is default queuing discipline in Linux. Classful queuing disciplines handle packet forwarding differentially according to the class of the packet. Filters are usually attached to the classful queuing discipline to classify the ingress packets. Examples of classful queuing discipline includes Class Based Queue (CBQ), Priority queuing and weighted fair queuing.

2) Priority queuing

The packets will be assigned to the priority class and has its own queue of its each. The Reference [6] states a priority list will contain the definiton for priority queue. The packets of higher priority will be processed first. And in the weighted fair queuing techniques. The packets arrives the classifier and here it is divided into higher and lower priority queues. If it is full it gets discarded else it allows to the queue and the process takes place in the processor.

3) Weighted Fair Queuing

Here the queues are weighted based on the priority of the queue. It takes packets in the queue in an round robin fashion. The packets are selected by turning switch located near the processor that selects the queue based on the priority of the weight. Then the processor will lead to departure. Reference [2] states that in many situations it might be desirable to give more bandwidth to the file and server, so they are given with two more bytes per cycle and the modified algorithm of congestion control algorithm is called as weighted fair queuing.

B. DiffServ on Linux

The different multi threading should be offered to fulfill different customer needs and requirements, the differentiated service (DiffServ) architecture. As explained in previous subsection, Linux kernel can support differentiated service provisioning with queuing disciplines, classes and filters. The filter classifies the incoming packets according to the predefined rules, and assigns one of the predefined classes. Packets in each class will be handled differently according to the priority and traffic parameters assigned to the class. Currently, CBQ, TBF and HTB queuing discipline can be used to provide differentiated services on Linux.

As stated in the reference [5] UNIX, Linux, FreeBSD and so on have the potential for both being secured and exploited, and some of the features make UNIX good for the security attacks to be operated safely. In addition to the differentiation in per-hop packet forwarding, the traffic engineering at each router should also provide QoS-guaranteed queuing and scheduling with maximized bandwidth utilization. Even though CBQ and TBF provide differentiated packet forwarding, they do not provide efficient re-distribution of extra bandwidth on the link. HTB is implemented deficit round robin (DRR) queuing, and supports hierarchical bandwidth sharing with *sharing hierarchy*. Implementation Details implement the token bucket filter i used POSIX threads in C to create a multithreaded environment for the simulation. For each of the components mentioned above i created a thread for doing the task specific to that component while controlling access to shared resources using mutexes and condition variables. The details of the threads is as follows.

C. Models of Multi-Threading:

There are three different kinds of multi-threading process. They are kernel level, user level and the hybrid threading. The difficulties and the possibility ways which are faced by software and users has distinguished as the models of multi-threading.

1) **Kernel-level:** The kernel level threading are the simplest possibility way of implementing the multi threading concept in an 1:1 corresponding level. This approach can be done using the Linux using C[3].

2) **User-level:** The application-level is mapped into single kernel level mapped entity. whereas the kernel-level threading as no knowledge about the threads we are using. So the user-level kernel is used to make the multi-threading easier. The one of the drawback of using user-level kernel is we cannot schedule more than one thread at a time, when we try to do that the whole process will be blocked to do that one process. The context switching can be done using user-level threading

3). **Hybrid level:** The hybrid level of threading is considered to be the complex one to implement because both the kernel and user level of threading codes should be modified before we are stepping in to the hybrid level of threading concept. As it avoids system calls it can be able to do the context switching process very quickly[3].

III. IMPLEMENTATION DETAILS

A. Queuing Mechanism:

There are three different kinds of multi-threading process. They are kernel level, user level and the hybrid threading. The difficulties and the possibility ways which are faced by software and users has distinguished as the models of multi-threading.

1) Packet Arrival Thread:

This thread would sleep for a certain amount of time (to simulate the inter-packet arrival time), creates a new packet and puts it into the input queue. If there are enough tokens in token bucket, then remove one packet from input queue and put it in the output queue. Also delete the used tokens from the token bucket. In case the packet requires tokens greater than the token bucket capacity then that packet is dropped.

2) Token Depositing Thread:

This thread would sleep for a certain amount of time (to simulate the inter-token arrival time), creates a new token and puts it into the token bucket. When there are enough tokens in token bucket, then remove one packet from input queue and put it in the output queue. Also delete the used tokens from the token bucket. In case the token is full then that token is dropped.

3) Server Thread:

This thread would sleep until a packet is received in the output queue. Once a packet is received it, would remove it from the queue and sleeps for a certain amount of time (to simulate the packet service time). The packet is then deleted.

4) Monitor Thread:

This thread would wait for the stop signal (to stop the simulation Ctrl+C keys are pressed) and would empty the queues and kill the other threads once the signal is received. When the simulation is executing each event is printed with its corresponding timestamp and once it is over the statistics are calculated and displayed. The input to the simulation could be in form of commandline arguments or location of a file that contained the input. Finally Ctrl+C could be used to signal the simulation to stop and print the statistics without processing all the packets.

B. Delay and Loss Estimation

We have to Assume that a traffic flow with poisson of arrival and its simulation parameters that has been listed in Table III. Average queuing delay obtained by simulation shown by the stimulated time and different token rates are given by the calculation model. Results of simulation and our calculation model have been closely watched. This means our model is precise. Mean arrival rate) $\sim 1 \times 10^7$ Then it shows the result of finite queue case, mean while parameters of the simulation are said to be same as infinite queue case except an extra parameter, queue size (The queue size is 5120 bits). perhaps Fig. 1 shows simulation obtained by average delay and calculation mode are given by the different token rates. We can find both queuing delay and loss rate are very close between the simulation and our calculation model viewed by the previous steps. After doing this The error percentage of queuing delay is (5.1%) at most. This proves that our models are accurate and precise.

C. Multiplexing (aim to share expensive resource):

In telecommunication and computer network, Multiplexing is a method by which multiple analog message signal. Reference [4] states that the process of sending several simultaneously communication channels over one link is called as multiplexing and all the communication link cost can be brought down by sending them in one link this is the advantage of using the multiplexing. we have to assume that there are n number of Poisson rtPS connections [c1, c2, ..., cn] are available .whose mean arrival rate are said to be [1 ...,n]. If we give these n connections of rtps, then they have the same delay,loss requirements and same bucket size b, which are d and l, while doing this we

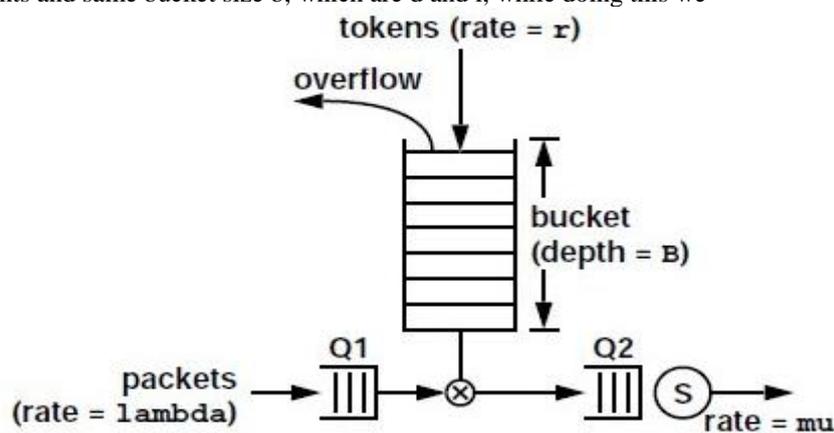


Fig 1. Multithreading.[10]

can easily find out the appropriate token rates [$r1, r2, \dots, rn$] by applying the same simple search algorithm typically ,a traffic shaper receives packet in burst. Instead of that now we assume there is a Poisson (rtPS) connection whose mean arrival rate is $(1 + 2 \dots + n)$ and as those of n connections has the same delay and loss requirement.

The mean arrival rate of combining two connections with mean arrival rate 1 has the property of poisson connection and 2 respectively is $1+2$.

Kitti Wongthavarawat, and Aura Ganz proposed an uplink packet scheduling mechanism with CAC is based on token bucket [9]. The overall bandwidth is clearly mentioned according to strict priority whereas (UGS class has the highest priority, and the BE class has the lowest priority)[8]. The class of lower priority one have to follow the higher priority that must be served earlier. The scheduling of UGS class is defined by the 802.16 standard. discipline to rtPS class They applied earliest deadline first (EDF) service. scheduled first might be the Packets with earliest deadline. They applied weight fair queue (WFQ) service discipline to this service flow. te weight of the connection based on the They schedule nrtPS packets (ratio between the connection's nrtPS average data rate and total nrtPS average data rates). The rest of the bandwidth is equally distributed to each BE connection. to predict the deadline of rtPS packets the arrival -service mechanism is prefferd in this paper that make use if it(They also proposed a CAC model[7]). This paper is has high contribution value and serves as our primary reference. According to the similar packet scheduling methods several research have been proposed. Xiaojun Xiao, Winston K.G. Seah, Chi Chung Ko, and Yong Huat Chew proposes a scheduling with hybrid and hierarchical architecture. BS and a secondary scheduler in SS consider as the solution design. The authors analyze the bounds of token rate and bucket size of a given traffic pattern. Derivations of measurement-based traffic specification (MBTS) which are presented as some useful algorithm. Our method of describing is different from others. Markov Chain state find out by appropriate rate by analyzing Markov Chain state and according to delay requirements on connections. Mohammed Hawa, and David W. Petr proposes a hybrid mechanism of weighted fair queue (WFQ) and priority queue. size in uplink sub-frame is also discussed in this paper by allocating the BW-request

contention. The size of BW-request contention period was checked according to the amount of transmitting data. Channel condition and some other factors were considered in. To maximize the bandwidth utilization In this paper, the policy-based scheduling has been proposed by Reuven Cohen and Liran Katzir. According to different load of synchronous, channel condition, and tolerated jitter (variation in the arrival of packets) [8], they proposed some different scenarios. Each scenario has its scheduler tasks. Some research about characterizing a traffic flow by token bucket had been done such as. Puqi Perry Tang and Tsung-Yuan Charles Tai use the measurement-based approaches to find appropriate token rate and bucket size.

IV. CONCLUSION

In this paper we proposed a mechanism in which multiple threads of execution remain ready to run (multi threading uplink packet scheduling and mechanisms). instead of promising their delay requirements bandwidth needed by real-time flows can be correctly reserved. We also proposed a model to convert (Poisson traffic flow into token bucket based connection). Sometimes it is contracted to muxing also mentioned and evaluated in this paper.

V. FUTURE WORK

Emulation of the Token Bucket algorithm using POSIX threads in C for a traffic shaper system, compiles on Linux. The aim was to stimulate a traffic shaper that receives and transmits packets to a server, while being controlled by a token bucket filter. Used multithreading, mutex, condition variables, signals for implementation.

REFERENCES

- [1] *Networking All-In-One Desk Reference by Doug Lowe.*
- [2] Andrew S.Tanebaum, *Computer Networks* Third edition, Vrije University Amsterdam, The Netherlands.
- [3] http://en.wikipedia.org/wiki/Thread_%28computing%29
- [4] Timothy S.Ramteke, *Networks* second edition, DeVry Institute of Technology.
- [5] Eric Cole, Ronald Krutz and James W.Conley, *Network Security* second edition.
- [6] http://www.cisco.com/c/en/us/td/docs/ios/12_2/qos/configuration/guide/fqos_c/qcftpq.html
- [7] Tzu-chieh Tsai, Chi-hong Jiang, Chuang-yin Wang, *Journal of communication*, vol. 1, no. 2, pp. 30-37, 2006.
- [8] <http://searchunifiedcommunications.techtarget.com/definition/jitter>.
- [9] Kitti Wongthavarawat, and Aura Ganz, "Packet scheduling for QoS support in IEEE 802.16 broadband wireless access systems", *International Journal of Communication Systems*, vol. 16, issue 1, February 2003, pp.81-96.
- [10] <http://www-scf.usc.edu/~trigunay/token-bucket.html>