TUNING RED FOR WEB TRAFFIC

MK MUSHTAQ, 17XJ1A0526

ABSTRACT

Classic RED (Random Early Detection) is a queue management system employed in routers for better performance and to avoid traffic congestion in the network. The paper 'Tuning RED for Web Traffic' refutes RED's performance being better than the regular tail-drop (or FIFO) queue management system, with the performance metric being end-to-end response time. The paper also discusses about RED's poor response times when link's maximum utilization happens. However, RED performs better, between 90%-110% offered loads, if the parameters are carefully tuned, however the response times are highly sensitive to the change in the parameter values. Post the experiments - after examining the results - up to 90% load RED does less than or equal to FIFO in terms performance - even after the parameters of RED are tuned, between 90%-110% tuning parameters result in better performance than FIFO but the response times are sensitive to parameter tuning and finally RED cannot achieve good response time while making maximum utilization – a trade-off exists between response time and link utilization.

INTRODUCTION

Reducing web traffic has been prominent domain of research in networking. One possible way to reduce web traffic is to have better handling strategies for queues when there is an influx of data packets coming onto the router. Many queue management systems have been proposed to manage the flow of incoming data; one such is RED also known to be Random Early Detection. RED is a queue management system that drops off packets pre-emptively unlike the traditional tail-drop that waits for the buffer to get filled for the packet-drop to take place. RED makes use of a predictive model in order to determine which packet to drop-off. The predictive model of RED uses statistical probability to selectively choose packets for the packet drop-off. The intuitive idea of how RED operates is, it accepts almost all packets when the filled queue length is zero and increases its rate of dropping with increase in filled queue length, therefore the highest drop rate will be experienced when the queue is almost full.

The initial drop probability is given by,

$$\mathbf{Pb = MAXp\ (Qavg - MINth)/(MAXth - MINth)}, \quad -(1)$$

where **MAXp** is maximum drop probability value, and **'Qavg'** is weighted average queue length. **MAXth**, **MINth** are maximum and minimum threshold values for queue length respectively, the use of these values will be explained as we progress along the report. The weighted average queue length is found out by equation,

$$\mathbf{Qavg = (1-W)*Qavg + W*Qsample,} \quad -(2)$$

Qsample denotes instantaneous queue length and W is another parameter of RED. The actual drop probability is **Pa = Pb / (1-count* Pb)**, where count is the number of packets in the present in the queue. So, the value of Pa determines what action to perform on the incoming packet - whether to enqueue onto the buffer or to drop it.

The steps followed by RED whenever a packet arrives is, first the average queue length is calculated using (2), if the value is less than **MINth** (minimum threshold), the packet is enqueued, similarly, if the average queue length is greater than **MAXth** (maximum threshold), the packet is dropped. If the average queue length falls between **MINth** and **MAXth** then, the probability of drop is calculated, if it is high then the packet is dropped if not it is enqueued into the queue. The steps above briefly explain the basic mechanism followed by RED.

The list of parameters of RED are put together in table 1

Parameter	Description
MINth	Minimum queue length threshold value

MAXth	Maximum queue length threshold value
W	Value to alter Qavg
Qlen	Maximum Buffer size
MAXp	Maximum probability for early drop-off

Table 1, RED parameters

Although RED was designed to perform better than FIFO, findings of the experiments performed explain that RED does not do better than FIFO with the metric of performance being end-end response time. The results of the experiment prove that, first up to offer loads of 90% RED has minimal or almost zero effect in improving the response time. Second, beyond offer loads of 90% but not more than 110%, RED can do better only if the above parameters mentioned are carefully tuned, however altering the parameters have effect on the response times as they are sensitive to change in the parameter's values, and also the best parameter setting can be achieved by trial and error method. Third, RED cannot attain both good response time and maximum utilization of link there exists a trade-off between them.

PROBLEM STATEMENT

Queue management systems have been a plausible approach to reduce web-traffic and give good response time to the users. Many such systems or algorithms have been researched and tried out. With many choices one is not really sure which queue management system to employ in his/her network. A comparative study among these queue management systems will help users discern which queue management system to make use of. One such queue management system is RED among many others. The idea kept in mind while designing RED was to improve the throughput of TCP connections. However, one of the basic and a user centric parameter for measuring the performance is end-to-end response time, that is used in the experiment. Therefore, RED's performance was studied under an experimental setup to observe how good response time does RED give under varied offered loads when compared to FIFO. By exploring possible values for the parameter's best performance of both RED and the conventional tail-drop was tried to observed. This observation was made under dynamically varied TCP connections with highly variable lifetimes. This strategy also sets path to compare new queue management systems performances with the already existing one.

SOLUTION DESCRIPTION

To observe RED's performance, a whole organizational network setup had been imitated in a laboratory setup. A program has been written that mimics a user or a browser actions in the laboratory network along with another program that mimics server's behavior, and these programs follow HTTP 1.0 protocols(only). The number of users, servers, requests and type of response among others have also been imitated like in the actual organizational network to monitor the performance of RED.

However, the simulations made above do not take care of the end-end delay caused due to the round-trip-time, therefore, again a program has been written that reproduces the round-trip-time behavior in the laboratory network. The delay value stretches from 7-137 milli-seconds. These values have been obtained from NetStat.net website, this way the end-end latency was generated.

Two more programs were written to monitor, (1) the HTTP response times; the program checks continuously the response times of the requests and makes a record of it, (2) average queue length; this is done because, if we refer to equation (2) above, it includes a variable **Qsample** which is instantaneous measure of queue length, therefore the program calculates the mean and variance of it every 3 milli-seconds and the record of this is logged every 100 milli-seconds, which is used in calculating **Qavg**.

One more point that is important to note is, the experiment is limited only to classic RED and not to its variations, WRED (Weighted RED), ARED (Adaptive RED) and RRED (Robust RED).

Data for traffic generation:

Apart from the programs mentioned above that reproduced users and browsers behavior, programs have been written for number of requests etc., that chose values from its respective data distribution. The data has been taken from UC-Berkeley campus in year 1995 that comprised 1.6 million HTTP protocols, a number sufficient for any satisfactory analysis. The essential elements that are required in the HTTP model include,

1. Request and reply length
2. Number of files referenced in a page
3. Time taken for two successive pages to return from the server
4. Number of requests continuously requested by the user

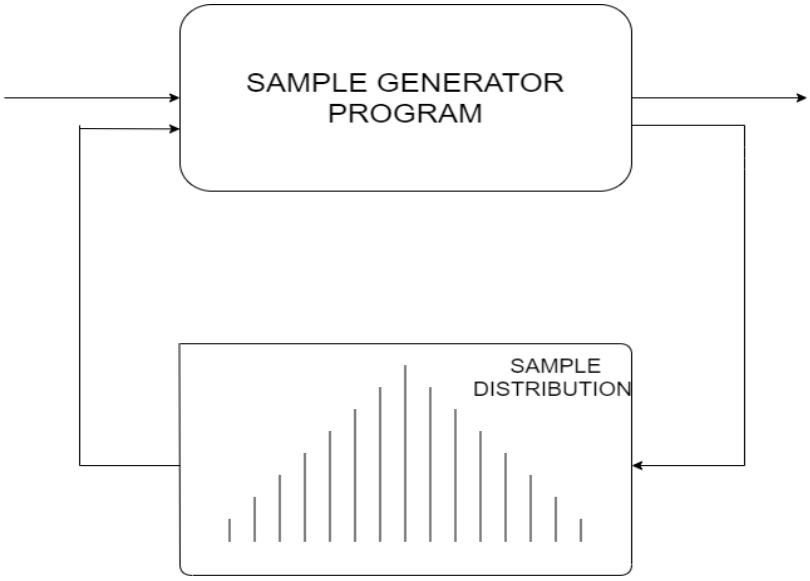

Figure 1

As mentioned above the programs were synthetic data generators. Synthetic data generator programs have been written that sample values from its respective distribution; for instance, the request length of a browser can be emulated by sampling a value from the request length distribution that has been obtained from UC-Berkeley campus. Similarly, other element's values have been sampled from their distribution. Figure 1 represents how sample generator programs run pictorially. The prominent elements responsible for web-traffic are the server's response size that includes the referenced files included in the response, the number of requests required to download all the files included in the response and the time spent by the user between two successive requests. These are the 3 essential factors responsible for creating a web traffic that can be satisfactory for the study.

To understand the process on a high level, it goes like this - for a request to be sent to the server, the request size is sampled from the request size distribution and sent to the server, before the request is sent to the server a TCP connection is initiated. The server sends the response of a particular size based on the message present in the request. Once the response has reached the user, the connection is closed and the response time is calculated and logged. Later, the amount of time spent for the next request to happen is, again, sampled from the "think" time or successive-request-time distribution, this is how the setup mimics the actual network.

As earlier mentioned, this experiment is made using HTTP 1.0 protocols, although 30% of the total HTTP protocols are comprised of HTTP 1.1, a major chunk still belongs to HTTP 1.0. HTTP 1.1 makes uses pipelined requests, which is more effective than HTTP 1.0, but the reason for not considering HTTP 1.1 is due insufficient data. Hence analysis was only made on HTTP 1.0 protocol.

Other constraint checks:

To conduct the experiment smoothly and not have any other hindrances some checks were made, only to abstain from erroneous results. Therefore 2 constraint checks were made only to prove that there is no effect of these constraint on RED's performance, these include

1. Whether there are other bottlenecks like CPU and interface speed for performance other than router's link.
2. Whether offered load can be controlled by simulating the number of users by being a parameter of web-traffic-generator.

To verify the presence of first point, an experiment has been carried out with two types of CPU's. First CPU was of 66Mhz and the other was of 200Mhz. The browser program was run on both the CPU's and the number of browser's were increased equally on both of them. The performance of both the machines were same and did not deviate much, this justifies that the type of CPU is not concern and we can conduct our original experiment without any protuberances it also verifies that the program doesn't have any resource limitation.

For the second constraint check, 7 machines were used where 5500 users were made use of and the offered load value was observed by increasing the number of users from 0 to 5500, the load value increased with increase in number of users linearly and hence the traffic could be controlled by passing the number of users as a parameter to browser generating program. The increase in number of users also attributed to the scalability of the program. These verifications resulted for smooth conduct of the experiment without any intervention of other unaccounted parameters.

The Experiment:

The experiment had been carried out with 3500 users and the experiment was made to run for 90 min. The first 20 minutes showed varied response times, due to the initialization and the stabilization effects hence they were discarded and the rest 70 minutes were taken into consideration. The size of the request generally is smaller than size of the response as the response contains embedded files. Since, the response size is big and considerate, the web traffic was studied on the server to user link, therefore the effects of different queue management systems were studied on this server to user link or the output link with performance metric being user-centric end-end response time. 90% of the requests have been responded in 500 milliseconds and the cumulative probability curve for number of responses vs number of users starts to saturate at 500 milliseconds implying most of the responses were completed by 500 milliseconds.

FIFO vs RED Comparison:

Before we directly compare FIFO's performance with RED's, first let us discuss how each of them individually performed and things that were done to get the best performance for each of them. First let us discuss about, FIFO or the conventional tail-drop method that drops packets only after the buffer gets full. Here the only parameter that can be tuned to get the best performance is the queue length, hence experiments were conducted to monitor how FIFO performed at different loads – at 50%, 70%, 80%, 90%, 98% and 110%. However, one cannot exhaustively search over all the possible integer values for queue length, hence a good and a pragmatic way was to have a queue size of 2 to 4 times the bandwidth delay product.

Bandwidth delay product value is the bandwidth of the link times round trip time for all connections, which is not a fixed value, hence a mean value 96KB has been considered. This implies the queue value to be between 190 and 380 approximately. Yet, for the benefit of doubt the experiment was performed with queue size varying from 30-240 and its performance was recorded at various offered loads. It turned out that queue size of 120-190 performs better at all the mentioned loads hence making it the ideal choice for the size of the queue. If the queue size is beyond this range and yet gives a good response times it cannot make maximum utilization of the link. Hence the ideal choice would be between 120 and 190.

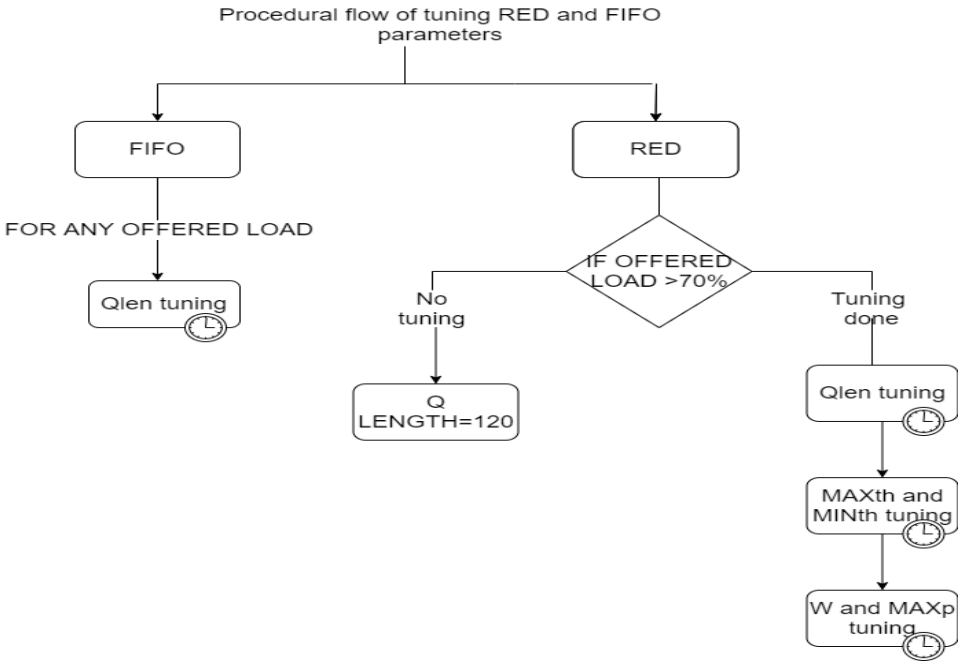

Figure 2

Figure 2 represents a high-level procedural view of tuning of parameters done in the experiment.

For optimum performance of RED - similar to FIFO - the parameters had to be tuned but FIFO had only 1 parameter, while RED has 5 parameters to tune. However, tuning is not always required in every situation at some offered loads without tuning also RED can perform well. Tuning of parameters becomes essential once the offered load exceeds 90%. Up to 90% offered loads, tuning of parameters is extraneous. To recollect, the 5 parameters include **MAXth**, **MINth**, **W**, **Qlen** and initial probability **Pa**.

Again, exhaustively searching all possible values for all the 5 parameters through trial and error in not advisory. Therefore, an initial set of experiments were carried out and guidelines from the original paper of Floyd and Jacobson were followed so as to choose a domain of values for each parameter instead exploring the whole number space. If remembered. a similar problem occurred in FIFO queue management system where the parameter was queue length and the order of it was determined using the bandwidth delay product.

So, the experiments were carried out by following the mentioned guidelines values, **W** = 0.002, **MAXp** = 0.1, **MINth** = 5 to 120, **MAXth** = 3*(**MINth**) and **Qlen** = 480. First experiment was conducted at an offered load of 50% and the drop rate observed was 0.01%, this behavior continues up to offered load of 70%. It is also observed that the queue doesn't get full even at 110% offered load, however the response time performance lessens. Once the load value exceeds the response time value deteriorates and exacerbates at 90-110% load. Hence, parameters need to be tuned at these load values.

The above experiment was carried out with MINth = 5, clearly not a good choice as it performs bad. Hence, we tune the MINth value and the best performance is obtained at MINth = 30 and 60, therefore (MI MINth Nth, MAXth) = (30,90) or (60,180). While changing these parameters the other parameters stay intact i.e., W = 0.002, MAXp = 0.1 and Qlen = 480. Later, the ratio of 3:1 for MAXth to MINth is altered to see any performance gains, after the altering the values a better performance was not achieved therefore the ideal values for MINth and MAXth have been set back to 30 and 90 respectively

Then the other parameters W and MAXp were tuned and the reason for tuning them together is they are closely related, therefore, staring with W=1/512, 1/256 and 1/128 and MAXp = 0.05, 0.1 and 0.25 the experiment was carried out and the optimal values turned out to be W = 1/512 and MAXp = 0.1 complying with the guideline's

recommended values. Another point to note is, MAXp at 0.25 had negative effects, because larger value of MAXp meant bigger drop-rates of packets, hence MAXp value was not kept too large and therefore limiting it to 0.1. Finally, Qlen needs to be tuned for which 120,160 and 480 values have been tried out and the best performance was yielded at Qlen being 120 - a value close to the one in FIFO. So, the best setting for the parameters were MINth = 30, MAXth = 90, Qlen = 120, W = 1/512 (or 0.002) and MAXp = 0.1. Parameters beyond the guideline's range were also tried but did no better than the tuned parameter values.

Comparing FIFO and RED's performance, RED did not perform better than FIFO. Their performances were similar, the only case where RED performed well than FIFO was at offered loads 90% - 110%, but only after careful tuning of parameters. However, changing of parameters also has adverse effects on the response time as the response times are highly sensible to the change in parameter values.

CONCLUSION

To conclude, RED performs equally good or worse and not better than FIFO and can perform better than FIFO only by tuning the parameters at 90% offered load. But there are some ideas that could be potential are of research:

1. Although, total response time is a good metric of performance but this metric alone cannot be used to decide how good the queue management system is. Inclusion few other metrics like throughput of TCP connections etc., should also be considered for overall performance of the queue management system.
2. The analysis made above had only HTTP 1.0 protocols only, a potential area of research could be to make analysis independent of type or version of HTTP protocol.
3. Lastly, the parameters are tuned in fragments, meaning first 2 parameters are tuned, then next 2 and so on and so forth, instead tuning all the 5 at once should be done and the performance should be observed.

Exploration on these lines can give more proper and complete insights about RED or any other queue management system.