Value-Based Utility Adaptation of RED Parameters

A Thesis Proposal

by Rachel Pua Villacorta

Candidate for Master of Science in Electrical Engineering
(Computers and Communications)

Department of Electrical and Electronics Engineering
University of the Philippines, Diliman

with Dr. Cedric Festin
as Thesis Adviser

October 2003

Abstract

Random Early Detection (RED) is used as a congestion avoidance mechanism in a router wherein the average queue size is compared to the minimum and maximum thresholds of the buffer. Packets are dropped, marked, or allowed to pass depending on the size of the queue relative to the thresholds. While RED may be suitable for flows which have similar loss requirements, it remains to be seen if it can be effective when flows have varying expectations. A solution is to make the RED thresholds adaptive to requirements. Studies on finding suitable values for these parameters are available but they are not directly applicable when there are differences in requirements. This study aims to provide solutions to configuring these parameters which uses information differentiating flows and their requirements. In particular, we will be using Value-Based Utility (VBU), a utility-based cost function to assess each flow's packet loss requirements. This additional information will then be used to dynamically tune and adapt the RED parameters.

1.0 Literature Review

1.1 Introduction

The number of Internet users is rapidly growing. Internet traffic load is thus increasing making congestion a big issue in computer networks.

Transport Control Protocol (TCP), a reliable connection-oriented transport protocol, uses a congestion control mechanism wherein retransmission timeout and duplicate acknowledgments (ACKs) are needed to detect congestion [14]. However, this congestion control mechanism is not enough to limit the congestion experienced in the network.

Congestion avoidance mechanisms are thus developed to complement the work that this congestion control mechanism accomplishes. With a congestion control mechanism, packet loss is inevitable since congestion is only detected when packets are lost due to buffer overflow or when a time-out occurs due to high delay. Congestion avoidance mechanisms, on the other hand, aim to detect incipient congestion. Thus, packet loss could be minimized while maximizing throughput and minimizing delay. There are several congestion avoidance mechanisms defined such as DECbit and Random Early Detection (RED), the latter being the more prominent one. Some simulations and papers have acknowledged the advantages of using RED [9, 10] but some authors refute these [14]. Some have extended RED knowing its capabilities and limitations. Some have combined RED with Explicit Congestion Notification (ECN) and compared this with Packet drop [5]. Some have combined RED and ECN with a preferential dropping [11, 12]. There are therefore many factors that could be studied to improve the overall treatment of congestion or incipient congestion. The type of congestion avoidance mechanism, its parameters, and congestion notification algorithms are among the things that could be investigated to improve network performance through metrics like throughput and delay.

User Datagram Protocol (UDP), an unreliable connectionless transport protocol, does not have a congestion control mechanism [14]. Unlike TCP, it does not respond to congestion notifications. It just uses up whatever

bandwidth is available. Dropping packets is then the solution in managing the buffer. However, in the upper layer, congestion indications are not ignored. Thus, even though UDP traffic is inherently 'greedy', it can still be studied in congestion avoidance issues. Some authors suggested penalizing these types of traffic since 'behaved' flows are the ones suffering from these 'greedy' flows [12]. On the other hand, some are not biased against these flows but are in favor of taking measures in controlling this type of flows [13].

1.2 Congestion

Congestion happens when too many packets are dropped frequently at the routers. This could be due to a fast sender sending too many packets, transmitting at a rate that a receiver cannot handle. Packets are queued in a limited, finite buffer and if the buffer becomes full, packets need to be dropped.

Congestion could also occur when too many users or applications are connecting to the same output. Again, since buffer space and capacity of the link are limited, the buffer could overflow, dropping packets in the process.

1.3 Congestion Avoidance Mechanisms

There are many mechanisms developed in congestion avoidance. As opposed to drop-tail which just allows packets to be dropped when the buffer overflows, active queue management is much recommended because it avoids congestion thus minimizing packet drops.

1.3.1 DECbit

The DECbit mechanism aims to avoid congestion at the router by setting a bit in the packets which experience congestion [14]. The receiver then sends an ACK to the source indicating that there is congestion. The source, which counts the number of packets with set congestion bit, will decrease its congestion window to 0.875 times the previous value when more than 50% of the total transmitted packets have set congestion bit.

1.3.2 Random Early Detection (RED)

Random Early Detection (RED) [10] is a congestion avoidance mechanism used for packet-switched networks. It is called Random Early Detection because it notifies the connections at random of incipient

congestion and anticipates congestion before it happens. RED is one of the most popular algorithms being used today to detect and avoid congestion. It is most commonly implemented in gateways since the gateways have a local view of the state of the connections. It allows occasional burst traffic by maintaining a regular, evenly spaced interval when marking packets, and avoids global synchronization by randomly choosing which connection to notify to decrease its window size in sending packets. In this way, simultaneous decreasing of windows will be avoided.

RED is designed to be used in conjunction with Transport Control Protocol/Internet Protocol (TCP/IP) in the transport layer but it could still be used when other protocols in the transport layer are implemented. This is because in RED, marking of packets can be implemented by dropping them rather than marking a bit in the packet header.

RED computes the average queue size as packets arrive in the gateways rather than the actual queue size to isolate cases of transient congestion. This means that RED allows occasional burst traffic and is not biased against it. After computing the average queue size, this value is compared to the minimum and maximum thresholds. If the average queue size is less than the minimum threshold, then the packets are queued with no marks. If the average queue size is greater than the maximum threshold, the packets are marked (a bit is set in the packet header) or the packet is dropped. If the average queue size is between the minimum and maximum thresholds, then the packets are marked with a probability proportional to the connection's share of the bandwidth.

The detailed RED algorithm is shown below:

INITIALIZATION:
    $avg = 0$;
    $count = -1$;
FOR EACH PACKET ARRIVAL:
    calculate the new average queue size avg:
        if the queue is not empty {
            $avg = (1 - w_q)avg + w_q q$; }
        else {
            $m = f(time - q\_time)$;
            $avg = (1 - w_q)^m avg$;
        }

if $min_{th} \leq avg < max_{th}$ {
increment *count*;
calculate probability p_a;
$p_b = max_p(avg - min_{th})/(max_{th} - min_{th});$
$p_a = p_b / (1 - count * p_b)$;
with probability p_a:
mark the arriving packet;
count = 0;
}
else if $max_{th} \leq avg$ {
mark the arriving packet;
count = 0;
}
else {
count = -1;
}

when queue becomes empty
q_time = *time*;

Saved Variables:
avg: average queue size
q_time: start of the queue idle time
count: packets since last marked packet

Fixed Parameters:
w_q: queue weight
min_{th}: minimum threshold for queue
max_{th}: maximum threshold for queue
max_p: maximum value for p_b

Other:
p_a: current packet-marking probability
q: current queue size
time: current time
f(t): a linear function of the time *t*

The average queue size is calculated as

$$avg = (1 - w_q)avg + w_q q$$

In calculating the average queue size, a low pass filter is needed and an exponential weighted moving average is used to implement it. This type of filter is used so that occasional bursts and transient congestion do not contribute too much to the increase in the average queue size.

In calculating the packet marking probability, the initial packet-marking probability, p_b, is determined by

$$p_b = max_p(avg - min_{th})/(max_{th} - min_{th})$$

This p_b is used in calculating the final packet-marking probability, p_a,

$$p_a = p_b / (1 - count * p_b)$$

This is the equation used so that the interval of marked packets is fairly evenly spaced to avoid clustering such that global synchronization is avoided.

Some guidelines had been followed in finding a value for the parameters. The weight in calculating the average queue size, W_q, should not be too large or too low. It must be greater than 0.001 [8] and must satisfy the following equation where min_{th} is the minimum threshold, and L is the allowable bursts of packets:

$$L + 1 + \frac{(1 - w_q)^{L+1} - 1}{w_q} < min_{th}$$

The minimum threshold, min_{th}, must not be too small such that the link utilization is not optimized. The maximum threshold, max_{th} must be set to at least twice min_{th} to account for the increase in the average queue size in one roundtrip time. It must also be greater than 0.1 [8, 10].

1.4 Congestion Notification Algorithms

The congestion notification algorithms should be carefully chosen to minimize or avoid global synchronization or bias against bursty traffic.

1.4.1 Packet Drop

Packet Drop is the basic way of notifying connections of congestion. However, studies have shown that this is not enough and that a more explicit indicator should be adopted.

1.4.2 Explicit Congestion Notification (ECN)

Explicit Congestion Notification is used in conjunction with a congestion avoidance mechanism (in a TCP/IP environment) which sets a bit in the packet header. Once the ACK is received and the source sees that this bit is set, it will halve its window size.

1.5 Parameter Setting

As mentioned before, there are many things that could be manipulated or could influence the network so that congestion is best controlled and avoided.

1.5.1 Type of Congestion Avoidance Mechanism and Parameters

As mentioned earlier, congestion control is not enough to address congestion problems. The type of congestion avoidance mechanism that will be used together with other factors (such as its algorithm and parameters) affect network congestion. RED, the more commonly used congestion avoidance mechanism, has a lot to offer in the area of network congestion. However, as mentioned earlier, it also has areas for improvement. The area that we are most interested in is the manipulation of its parameters. One of the studies on RED [5] suggests that the parameter max_p be investigated. Another study [14], on the other hand, suggests that the calculating probability algorithm be investigated.

1.5.2 Type of Congestion Notification Scheme

Most of the current experiments now explore schemes other than dropping packets. ECN is more commonly used. The congestion notification scheme is important in congestion control and avoidance since congestion management does not just end in the routers. End to end management is equally important since it is the sender who will be adjusting its transmission rate and transmitting packets over the network.

1.5.3 Type of Flows

The traffic mix affects the performance of the network. TCP flows are responsive to congestion indications. UDP flows, on the other hand, are unresponsive to congestion. One paper [11] categorizes traffic into three categories: non-adaptive (e.g., audio and video applications), robust (respond to congestion but uses up as much bandwidth as allowed), and fragile (responds to congestion but cannot adapt easily to available bandwidth and is sensitive to losses). In the same paper, Flow Random Early Drop (FRED), an extension of RED, is studied. It tries to eliminate RED's tendency to be 'unfair' since it does not distinguish between traffic types.

1.5.4 Traffic Load

Studies show that traffic load influences the Quality of Service (QoS) of computer networks. RED, in particular, does not operate optimally under different traffic conditions. A study, then, proposes to maintain high throughput and low delay while adaptively managing the queue [14].

1.5.5 Queuing Discipline and Scheduling

FIFO

In First-In-First-Out (FIFO) [14], packets are queued in the buffer as they arrive. Packets are serviced in the order of arrival in the router, i.e., packets that arrived first get serviced first. With drop-tail, packets are just dropped once the buffer is full. With other mechanisms, performance of FIFO queues will greatly improve.

Fair Queuing

Traffic flows get differentiated in fair queuing. Each flow is queued in separate buffers. Round Robin scheduling is implemented in the different queues. Round Robin is done by passing through all the queues one after the other in an ordered manner. Once the last queue is serviced, the first queue is serviced next and again, the same order is followed.

1.6 Value-Based Utility

The Value-Based Utility [6] is a tool, for measuring the level of satisfaction or dissatisfaction of users. Its definition is given by:

$$U_{i,QoS,m,\Delta t}(p,b) = G/N - (N-G)/N * p/q \qquad \text{Equation (1)}$$

where U is the utility of a flow i for the required QoS, taken at point m during the interval $\Delta t = (t_2-t_1)$

p is the target percentage of packets meeting the required QoS

b is the target QoS bound

G is the number of packets that satisfy the requirements of a certain flow

N is the total number of packets of a flow

q is the allowed percentage of error and is equal to *1 - p*

Utility is 1 when *G* equals *N*. This means that all packets of a certain flow satisfy all the requirements for that flow. This is what is referred to as $happiness_{max}$. This indicates that the user corresponding to that flow is satisfied to the highest level.

Utility is 0 when *G* equals *N*p*. This is what is referred to as $happiness_{min}$. This indicates that the user corresponding to that flow is satisfied but to the lowest level only, i.e., when the percentage of the packets satisfying the requirement is equal to the target percentage of packets meeting the required QoS.

Utility is between 0 and 1 when there is no target requirement percentage for meeting bound *b,* when *p* equals 0, or when the target QoS bound, *b* is not set. This could also happen when *G* is greater than *N*p*, i.e., when a certain flow *i* exceeds the target percentage of packets meeting the required QoS bound. In this case, the user is still happy and the degree of happiness varies from 0 to 1 depending on how much the target is exceeded.

Utility is negative when *G* is less than *N*p* which means that the percentage of packets that meet the requirement is less than the target expectation. Negative utility is bounded by $unhappiness_{max}$ when *G* equals 0 or when utility is *–p/q*. This also means that there are no packets that satisfied the target QoS bound.

Utility is shown in diagram form [6] in Figure 1-1:

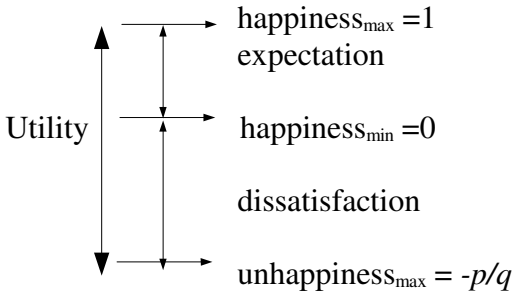

Figure 1-1. Utility in diagram form

The definition of utility in Equation (1) is not applicable for a deterministic requirement wherein *p* is always 1. In the above equation, when *p* equals 1, *q* equals 0 which would give an undefined *p/q* term. Moreover,

when *G* equals *N*, the above definition would give a utility of 1 or $happiness_{max}$. This should not be the case for a deterministic QoS requirement since when *p* is always equal to 1 and *G* equals *N*, this points to the fact that the QoS requirement is just satisfied and degree of satisfaction is only minimum. This also means that since it is expected that all packets satisfy the target QoS bound, then, when this happens, the degree of happiness is just the minimum. For a deterministic requirement, the definition of utility is expressed as:

$$U_{i,QoS,m,\Delta t}(p,b) = -(N-G)/N * \alpha'$$

where α' is some value greater than *p/q* of the known smallest *q* at point *m*

The Value-Based Utility could be used in traffic management when controlling the admission of connections, classifying packets, shaping and conditioning traffic, marking packets, scheduling, and discarding packets.

In admission control, the Value-Based Utility could be used to determine how many users are satisfied. If this number, say *T*, taken over the total number of users, say *W*, meets a certain percentage, *X*, then new connections could be admitted and the number of connections to be admitted could depend on how far or near *T/W* is to *X*. The Value-Based Utility could also be used to adjust the utility of users to give way to new connections. It could look at some number of users with the highest value of utility, and then adjust the resources of these connections such that these utilities will decrease while retaining a minimum level of satisfaction and making room for new connections.

In packet classification, the Value-Based Utility could be used to attach a label to each flow. Labels could be ‘highly satisfied’, ‘satisfied’, and ‘unsatisfied’. ‘Highly satisfied’ flows could be adjusted such that these become ‘satisfied’ flows and some, if not all, ‘unsatisfied’ flows could be adjusted to become ‘satisfied’ flows. Moreover, some, if not all, ‘satisfied’ flows could be promoted to become ‘highly satisfied’ flows as long as there are no more ‘unsatisfied’ flows.

In traffic shaping and conditioning wherein the transmission of packets is regulated and a connection's profile is checked, the Value-Based Utility could be used to slow down or to speed up transmission of packets by manipulating p, b or Δt or it could be used to police connections and see if these still adhere to the contract agreed upon with the network.

In scheduling, the Value-Based Utility could be used, for example, in priority scheduling, to choose which flow to service first depending on the degree of happiness or unhappiness of each flow. Flows that have lesser level of happiness could be serviced first with the happiest flow serviced last. Service weights or percentage of flows permitted to the network could also be adjusted to improve the utility of unhappy flows.

In packet marking or discarding, the Value-Based Utility could be used to classify which and when to mark or discard packets. This is important in congestion avoidance mechanisms such as RED. Since the Value-Based Utility could be used to know the state of each flow, packets belonging to a certain flow that need to be dropped or marked when implementing RED could still be queued or serviced to improve its happiness. Moreover, the Value-Based Utility could also be used to dynamically adapt or change the parameters critical to the implementation of RED. This is an area that needs to be thoroughly researched and is the area the researcher plans to study in depth.

As shown in the algorithm of RED, the parameters – minimum and maximum thresholds are critical for the implementation of the mechanism. This is where the Value-Based Utility will be used. The Value-Based Utility will be used to dynamically adapt these parameters to avoid congestion while maintaining or improving the satisfaction of some, or improving the state of unsatisfied users.

2.0 Statement of the Problem

RED treats all flows similarly even if flow requirements are different. This results in an insufficient queue management leading to dissatisfaction of some flows. A solution is needed to address this problem. Value-Based Utility offers flow identification based on requirements which can be adapted for RED. We believe that

with this additional information used to adapt and tune RED parameters, the problem of uniform treatment of flows will be mitigated and RED's performance improved.

3.0 Methodology

The initial phase includes designing and implementing Drop Tail and RED in Network Simulator 2 (NS2), a tool wherein buffer management strategies could be experimented on and modified. After implementation, simulation results will be compared based on packet loss. Similarly, RED with VBU management (RED-VBU) will be designed and implemented in NS2. Simulation results will also be compared based on packet loss, delay and throughput.

3.1 Simulation environment

All experiments will investigate the following factors unless specified otherwise.

3.1.1 Network Topology

Single-hop and double-hop models will be used. In the single-hop model (Figure 3-1), sources S_1, S_2, S_3,... S_n are connected to the router *R* with *O* as the output sink. Utilities could be measured and kept in both R and O. Utilities measured in O will serve as feedback to R. These will be used in recalculating the utilities in R. On the other hand, in the double-hop model (Figure 3-2), sources S1, S2, S3,... Sn are connected to the router, R1, like in the former model but this time, R1 is connected to another router, R2 with O as the output sink. In this model, utilities could be measured in R1, R2 and O. The utilities measured in O will serve as feedback to R2. R2 will then recalculate the utilities and in turn will serve as feedback to R1. Note that where the utilities will be measured is still subject to experimentation.

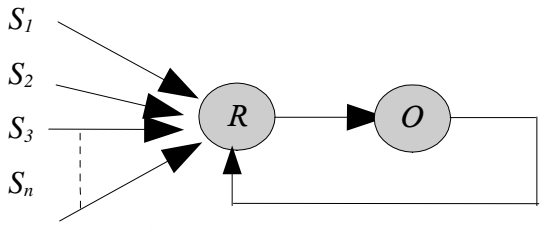

Figure 3-1. Single hop model

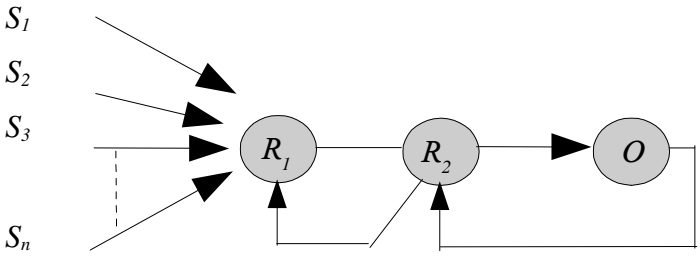

Figure 3-2. Double hop model

3.1.2 Traffic Source Models

In general, we will consider two types of traffic – bulk transfer and interactive traffic. We will be looking at some source models commonly used and available in NS2. Initially, we will look at the Exponential and Pareto distributions in generating the traffic sources. These distributions will be used in varying the size of packets and the transmission rate of each packet.

3.1.3 Network Load

Load will be varied to see the effects of the buffer management schemes. The schemes will be tested under *light, medium* and *heavy* load. This classification could be defined by looking at one of these things – the utilization, utility or the packet drops. A high utilization would mean that the buffer is almost always with packets that need to be serviced. The utilization, U, could be measured by counting the amount of time, B, that there is/are packet/s in the queue and dividing it by the measurement period, T. A high utility, $\boldsymbol{U}$, would mean that the flow is very happy. This means that the flow is satisfied with the service that it gets. High packet drops would mean that the buffer frequently overflows. Thus, load is considered *light* when the U is low, $\boldsymbol{U}$ is high and packet drops is zero. Load is considered *medium* when U, $\boldsymbol{U}$ and packet drops are 35-70%.

Load is considered *heavy* when U is high, $\boldsymbol{U}$ is low and packet drops is almost 100%.

3.1.4 Traffic mix

Flows will be grouped according to their requirements. Initially, the groupings used in [6] will be adapted. Flows will be grouped as follows : High Expectation Flows (HEFS) have a target, p, of 99%, Medium Expectation Flows (MEFS), 90% and the Low Expectation Flows (LEFS), 80%. HEFS are those flows which could tolerate 1% packet loss while MEFS could tolerate 10% packet loss and LEFS 20%. The number of sources with each type of flow will be varied.

3.1.5 Service Time

Service time, S, is the average amount of time that is needed for a packet to be serviced. S of each router will be deliberately set to values that will lead to congestion. This is needed to determine the effect of the schemes to be investigated. Setting S small would cause the buffer not to overflow or be empty at times. This is not the desired scenario since we are observing at the performance of the schemes while congestion is happening. S should then be slower than the arrival rate, λ, □of each packet. The experiments will be implemented with the service time kept constant while varying the buffer length.

3.1.6 Routers R_1 and R_2

Each router will have a single buffer that will be used for queueing. The length will be varied from large to small. This will show the effect of the schemes when there is congestion. Experiments will be designed to look at the effect of logically partitioning the buffer. There will be experiments using a single buffer with no partition and experiments using virtual buffers to accommodate each type of flow. These virtual buffers would allow differentiation of the flows so that each flow will be serviced according to its need. Since there are three types of flows that will be considered – HEFS, MEFS and LEFS, we will initially

use three virtual buffers. Each buffer will maintain RED's parameters – min_{th}, max_{th}, max_p and w_q. Upon implementing the experiments, the number of logical partitions will be investigated together with the length of each virtual buffer and the values of each RED parameter per virtual buffer.

The bandwidth of the link connecting R_1 and R_2 should be small to induce congestion in both R_1 and R_2. This would mean that S will be large or servicing would be slow since S is inversely proportional to the bandwidth. Again, S should be slower than λ.

3.1.7 Duration of the simulation

Preliminary experiments per scheme will be done to look at the duration of the simulation that will be used in all the experiments. Not all data will be used particularly the transient period associated with the startup phase. This will be removed so that the results will not be affected by the fluctuations during the start of the simulation.

3.2 Experiments

Majority of the experiments will use the single hop model and we shall initially investigate the Exponential and Pareto distributions as traffic source models. Some experiments will be implemented in the double hop model to be able to generalize the results. The experiments will be implemented under *light, medium* and *heavy* load. Traffic mix will also be investigated. Preliminary experiments will be simulated at different durations to extract that which gives the most stable results. Experiments will be designed and implemented using Drop Tail, RED and RED-VBU schemes.

3.3 Proposed Scheme : RED-VBU

NS2 script for RED will be modified by integrating VBU management of packet loss requirements in RED's algorithm. Initially, routers will each have three virtual buffers with each buffer assigned to a specific class. This scheme will investigate the sizes of each virtual buffer under a predefined buffer length. The experiments will be investigated under different factors shown in Table 3-1.

Factors	*Description*
time interval Δt	time interval in getting VBU measurements
handling packets	queuing, dropping or marking
RED parameters	minimum and maximum thresholds, max_p (maximum value for probability p_b), and w_q (queue weight)

Table 3-1. Factors affecting RED and VBU

In RED-VBU, we will compute the utility per flow in the router. The average queue size will be computed every packet arrival. Only after Δt will we decide on the appropriate way of handling the packets. The decision will now be based on the average queue size and the utilities in comparison to the thresholds. After the flow is serviced, the utility will be computed again in the output sink. The utility measured will then be sent back to the router. The previous utility will be compared to this feedback utility. Based on this comparison, the size of the virtual buffers and/or the values of RED parameters will be adjusted.

The last implementation of RED-VBU scheme will also be investigated on the double hop model to look at its scalability and consistency. Under the same simulation environment, Drop Tail and RED will also be implemented on the double hop model. Implementation in the double hop model will only involve managing packet loss requirements similar to the implementation in the single hop model.

3.4 Analysis of Results

Scripts will be written to reduce the data gathered in the simulations. This will remove the data that are not needed in the trace file extracted from the simulations. Xgraph, which is integrated in NS2, or GNUplot, which is an open source data plotting program, will be used to graph packet loss, delay and throughput.

Results of Drop Tail, RED and RED-VBU will be compared. Comparisons will be shown through graphs.

3.5 Schedule

Phase	*Month 1*	*Month 2*	*Month 3*	*Month 4*	*Month 5*	*Month 6*
Phase 1						
Phase 2						
Phase 3						
Phase 4						
Phase 5						
Phase 6						
Phase 7						
Phase 8						

Figure 3-3. Schedule

Phase 1 consists of performing preliminary experiments on Drop Tail using the single hop model implementing the different traffic source models, experiments on Drop Tail under different simulation environment, extracting the data that we will be using from the trace file and graphing packet loss, delay and throughput metrics.

Phase 2 consists of performing experiments on RED using the single hop model and graphing packet loss, delay and throughput metrics.

Phase 3 consists of designing experiments on RED using the single hop model.

Phase 4 consists of performing experiments on RED using the single hop model and graphing packet loss, delay and throughput metrics.

Phase 5 consists of comparing and analyzing of results using the graphs obtained.

Phase 6 consists of designing and performing experiments on Drop Tail, RED, and RED-VBU using the double hop model under the simulation environment that produced the most satisfactory RED-VBU scheme results using the single hop model.

Phase 7 consists of comparing and analyzing the results from Phase 6.

Phase 8 consists of making the documentation.

4.0 Bibliography

[1] M. Arpaci and J. Copeland, “An adaptive queue management method for congestion avoidance in TCP/IP networks,” in *Proceedings IEEE GLOBECOM 2000*, San Francisco, California, November 2000.

[2] U. Bodin, O. Schelen, and S. Pink, “Load-tolerant differentiation with active queue management,” *ACM SIGCOMM Computer Communication Review*, vol. 30, no. 4, pp. 4-16, July 2000.

[3] T. Bonald, M. May, and J. Bolot, “Analytic evaluation of RED performance,” in *Proceedings of INFOCOM*, pp. 1415-1424, 2000.

[4] B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, and L. Zhang, “Recommendations on Queue Management and Congestion Avoidance in the Internet,” Request for Comments (RFC) 2309, April 1998.

[5] W. C. Feng, D. Kandlur, D. Saha, and K. Shin, “A self-configuring RED gateway,” in *Proceedings of INFOCOM'99*, March 1999.

[6] C. Festin, “Utility-based buffer management and scheduling for networks,” Ph.D. Thesis, University of London, 2002, Ch. 2 & 3.

[7] V. Firoiu and M. Borden,“A study of active queue management for congestion control,” in *Proceedings of INFOCOM 2000.*

[8] S. Floyd. “RED: Discussion of Setting Parameters”. http://www.icir.org/floyd/REDparameters.txt. (Accessed August 2003).

[9] S. Floyd and K. Fall, "NS simulator tests for random early detection (RED) queue management," Lawrence Berkeley Laboratory, April 29, 1997.

[10] S. Floyd and V. Jacobson, “Random early detection gateways for congestion avoidance,” *IEEE/ACM Transactions on Networking,* vol. 1, no. 4, pp. 397-413, August 1993.

[11] D. Lin and R. Morris, “Dynamics of random early detection,” in *Proceedings of ACM SIGCOMM '97,* Cannes, France, pp. 127-137, October 1997.

[12] R. Mahajan, S. Floyd, and D. Wetherall, “Controlling high bandwidth flows at the congested router,” in *Proceedings of the ACM 9th International Conference on Network Protocols (ICNP)*, Nov. 2001.

[13] M. Parris and K. Jeffay, “A better-than-best-effort service for continuous media UDP flows,” NOSSDAV '98.

[14] L. Peterson and B. Davie. *Computer Networks: A Systems Approach*. 2nd ed. Academic Press, 2000.

[15] S. Reddy, “LRU-RED: An active queue management scheme to contain high bandwidth flows at congested routers,” in *Proceedings of Globecomm*, Nov. 2001.