A New Class-Based Traffic Queue Management Algorithm in the Internet

Ye Zhu
Department of Electrical and Computer Engineering, Cleveland State University
Cleveland, OH 44115, USA
[e-mail: y.zhu61@csuohio.edu]

Received September 14, 2009; revised November 7, 2009; accepted November 20, 2009; published December 31, 2009

Abstract

Facing limited network resources such as bandwidth and processing capability, the Internet will have congestion from time to time. In this paper, we propose a scheme to maximize the total utility offered by the network to the end user during congested times. We believe the only way to achieve our goal is to make the scheme application-aware, that is, to take advantage of the characteristics of the application. To make our scheme scalable, it is designed to be class-based. Traffic from applications with similar characteristics is classified into the same class. We adopted the RED queue management mechanism to adaptively control the traffic belonging to the same class. To achieve the optimal utility, the traffic belonging to different classes should be controlled differently. By adjusting link bandwidth assignments of different classes, the scheme can achieve the goal and adapt to the changes of dynamical incoming traffic. We use the control theoretical approach to analyze our scheme. In this paper, we focus on optimizing the control on two types of traffic flows: TCP and Simple UDP (SUDP, modeling audio or video applications based on UDP). We derive the differential equations to model the dynamics of SUDP traffic flows and drive stability conditions for the system with both SUDP and TCP traffic flows. In our study, we also find analytical results on the TCP traffic stable point are not accurate, so we derived new formulas on the TCP traffic stable point. We verified the proposed scheme with extensive NS2 simulations.

***Keywords*:** RED, quality of service (QoS), active queue management (AQM), utility, Internet traffic engineering

The preliminary version of the paper appeared in 2008 International Conference on Communications and Networking in China and won the best paper award from the conference.

DOI: 10.3837/tiis.2009.06.007

1. Introduction

Protocols for today's Internet have do handle overload and congestion, which in turn can be they caused by insufficient engineering of the network, by flash crowds, or by denial-of-service attacks. Traditionally, some control over network overload has been provided by the limited buffer availability at network nodes. Excessive traffic is dropped when buffer overflows, and packets "at the tail" of the buffer queue are dropped. RFC 2309 [1] has pointed out a number of shortcomings of drop tail, the traditional queue management. They are:

- "Lock out" phenomenon: This occurs when one or several flows occupy all the buffer space so that there is no room for other flows.
- Long periods of full queue status: The buffer is near full for a long period of time which cause the long end-to-end delay.

Active Queue Management (AQM) has therefore been recommended as a mechanism to control congestion and improve network performance. AQM solves the above problems by pro-actively managing the buffer, for example by dropping or marking packets before the queue becomes full. These dropped or marked packets will subsequently trigger a notification to "responsive" senders to slow down the transmissions of packets. In this way, the congestion can be avoided or alleviated. RFC 2309 also recommends one AQM mechanism, Random Early Detection (RED), was first proposed in [2]. The main advantages of RED are as following:

- RED can prevent global flow synchronization which may occur when drop tail queue management is used and cause low link utilization.
- RED can achieve low end-to-end delay and low buffer consumption.
- RED has no bias against bursty flows.
- RED can avoid consecutive packets drops because probabilistic early drop can prevent queue overflow.
- RED is fair in terms of bandwidth consumption for flows which are responsive, because more packets are likely to be dropped for flows consumes more bandwidth.

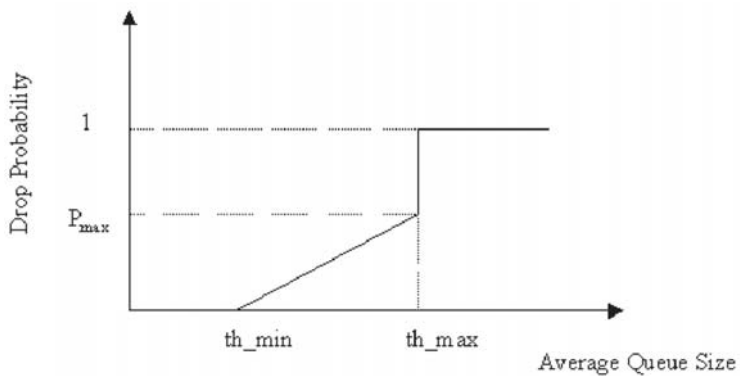

Fig. 1. RED Drop Curve

Fig. 1 illustrates how packets are handled in a node with RED queue management. The RED drop curve is as **Fig. 1**. An RED controller is described by five parameters: lower threshold (th_min), upper threshold (th_max), weighting factor in calculating average queue

size, buffer size and max drop probability(P_max) when average queue size is equal to upper threshold. When the average queue size is below the lower threshold, no packets are dropped. When the average queue sizes crosses the lower threshold, every incoming packet is dropped with a probability that increases linearly with the average queue size. When the average queue size crosses the upper threshold, all the incoming packets are dropped. The other details about RED can be found in [1], [2] and [3]. Many alternatives to RED are proposed in the literature, such as BLUE [4], SRED [5], REM [6], adaptive REM [7], AFD [8], and CHOKe [9].

While AQM, and RED in particular, unquestionably addresses the problems of drop tail management, it also has a number of drawbacks. First, non-responsive flows will not back off due to packets drop, which causes a fairness problem between responsive flows and irresponsive flows. Second, the large numbers of parameters involved in characterizing an RED controller make RED-based systems hard to tune. In [10], some guidelines for setting these parameters are given. But all these guidelines are based on experience.

In [11], the authors compare the performance of RED and of traditional drop tail in a testbed for RED consisting of real routers and PCs. Their test traffic includes ftp traffic, http traffic and UDP traffic. According to their test result, RED "does not exhibit much better performance than Tail Drop". In the case of UDP traffic, Tail Drop is "more aggressive than RED with regard to policing non-responsive flows". They also pointed out the parameters choice is an "inexact science" and that this problem needs more research.

Similar researches are described in [12]. In that experiment, a real network was established to simulate an enterprise or campus network with a single WAN link to an ISP. The test traffic consists of only http web traffic. Their major performance measure is the response time, a user oriented metrics. Their work shows that, under heavy load, RED has no significant advantages over traditional drop tail. They also conclude that the network performance is sensitive to the choice of parameters.

In the following, we will consider networks with a wide mixture of traffic flows, stemming from different applications, which in turn have different Quality-of-Service (QOS) requirements. These traffic flows differ from each other in many aspects, such as bandwidth requirements, dynamics, responsiveness to congestion control, or others. In such environment, RED performs poorly for a number of reasons:

First, RED operates on aggregates of flows, possibly over many classes of applications and over many protocols. Some non-responsive flows, which may be used by applications based on UDP without window control mechanism, will not respond to the congestion notification. Even more, they may increase their sending rates due to retransmission mechanisms in the application layer. Thus more buffer space will be occupied by these non-responsive flows. Consequently, responsive traffic is penalized and the network performance generally degrades.

Second, responsive flows may be treated unfairly as well. Because different responsive flows may have different dynamics, their response to congestion notifications may be different.

Finally, for responsive flows that have similar dynamics, again RED may behave sub-optimally. While RED effectively controls congestion in switches and routers, the effect of this control is perceived unevenly by different application classes. For example, if a packet loss happens on a loss-tolerant UDP video application, there may be no need for re-transmission, because one packet loss will not bring a perceivable video quality degradation. Moreover, maybe after re-transmission, the packet arrival time has passed the processing deadline for the packets and so it is just discarded if it is re-transmitted in this case. For many other applications, however, a single packet loss may significantly affect the perceived quality

of service. In interactive applications over TCP, for example, the loss of a single packet significantly affects transmission delays and throughput for subsequent packets. For a short life web request, for example, delay is a very critical performance metrics because this is a user-interactive application. If a packet loss happens on this web request, retransmission is surely required unduly delaying the response for the request. Hence, for different applications, a single packet loss can induce very different effects as perceived by the applications or end users.

There clearly is a need for a differentiated operation of RED, where flows are classified and treated according to the protocol and application class characteristic, which includes responsiveness, dynamics and perceived effect of certain loss rate. For example, the loss-tolerant UDP video application described above should be considered as an application running on UDP, and it is non-responsive, requiring no retransmission, and insensitive to single-packet losses. Alternatively, telnet is a TCP based, responsive applications, requiring retransmission and sensitive to packet losses.

Some of previous work has targeted a differentiated treatment to the packets in the queue.

One example is RIO proposed in [13]. Its purpose is to provide different levels of best-effort service. RIO can only work on TCP traffic. In RIO mechanism, traffic out of service profile and in service profile is differentiated by tags. The router has different drop policy for packets with a tag and packets without a tag. The RIO scheme can differentiate the traffic, but it does not give a way to achieve the optimal network utility.

Another example is Weighted RED (WRED) proposed by CISCO in [14]. WRED is targeted to provide different quality of service based on RED mechanism. WRED uses the IP precedence field in IP packets to differentiate packets. So for IPv4, there will be eight drop policies. Packets with higher precedence will be dropped with less probability. WRED also assumes that the source is using TCP.

The above two forms of differentiated RED are all aimed to provide different quality of service. According to [13], RIO "can allocate bandwidth to different users in a controlled and predictable way during network congestion". For WRED, according to [14], "packets with a higher IP precedence are less likely to be dropped than packets with a lower precedence."

But they still have some problems. First, they all assume the source is sending TCP traffic. They have not dealt with non-responsive traffic.

Second, they all need edge routers to do some co-operation so that the core routers can do differentiation. For RIO, edge routers tag the packets according to service profile. For WRED, edge routers assign precedences to packets.

Third, the differentiation provided by RIO and WRED is at the priority level. It is not sufficient to capture all the essential characteristic of application traffic to achieve optimality. For example, if packets of a delay sensitive telnet application and a delay insensitive ftp application are both out of service profile for RIO mechanism (or of the same precedence for WRED), then the same drop probability or policy for them has not the same effect as perceived by the applications. For telnet applications, the retransmission delay can be perceived by the end user while for ftp applications, the user may even not care if there is delay.

In this paper, we will present a method to take advantage of application specific information about dynamics (and so its response to RED activity) of the traffic and the effect that RED has on perceived application performance. Our method does not assume there are only TCP sources.

The remainder of this paper is organized as follows: Section 2 will introduce related work in both network congestion and real-time scheduling area. In section 3, we will describe the system model. A description of our scheme will be given in section 4. Section 5 will give our

simulation result. In section 6, we will conclude and give an overlook on future work.

2. Related Work

2.1 Handling Network Congestion

Network congestion avoidance has been studied for many years. Additive increase and multiplicative decrease mechanism was first proposed by Jain and Chiu [15]. This mechanism is only applied in congestion avoidance phase. The idea is that when the TCP sender gets an indication of congestion, it will cut its congestion window by a factor, such as two, thus cutting the window in half. If there is no congestion indication, the congestion window will increase by a constant, such as one. This mechanism is adopted in 4BSD TCP implementation according to [16]. Jacobson [16] also presents the other six new algorithms adopted in 4BSD TCP such as round trip time variance estimation, slow start and fast retransmit. More information can be found in that paper.

Chiu and Jain [17] have analyzed the additive increase and multiplicative decrease mechanism for TCP from efficiency, fairness, distributedness and convergence aspects. This paper has compared the additive increase/multiplicative decrease mechanism with the other three alternatives. They are additive increase/additive decrease, additive increase/multiplicative decrease and multiplicative increase/additive decrease mechanisms. It is analytically shown in this paper that the additive increase and multiplicative decrease mechanism "satisfies the sufficient conditions for convergence to an efficient and fair state regardless of the starting state of the network".

In Jacobson's [16], it is also pointed out that additional support is needed at gateway level. Random Early Detection (RED) has been originally proposed by Floyd [2] and is an example of such a gateway based scheme. The RED mechanism has been introduced in the introduction section. Further researches on RED include BLUE [4] and SRED [5].

In BLUE, the active queue management is not based on average queue length, but on packet loss and link utilization instead. Through simulations and experiments, the authors show that BLUE can reduce the packet loss rate and network delay in comparison with the traditional RED mechanism.

SRED's queue management relies on queue length. The queue occupation is stabilized by estimating the number of active flows. A concept called hit is introduced in SRED. A hit happens when the new packet and one of its recent preceding packets are from the same source or belong to the same flow. The hit is used in the estimation and finding out the "misbehaving flows". The authors use the simulation to verify the SRED mechanism.

In [18], a fluid based analysis on TCP flows and RED routers is presented. Their analysis and modeling lead to two differential equations which describe the dynamics of TCP flows through RED routers. The first differential equation describes the additive increase and multiplicative decrease mechanism. This equation has also taken into account the delay between the packet drop and source's knowledge of the packet drop. The second differential equation models the queue length change in the RED router when the router is congested. The queue length change results from two aspects. Since RED routers use average queue length to determine the drop probability, Misra et al. [18] have also modeled this averaging behavior of the RED routers. The averaging factor and sampling interval have been taken into account in the modeling.

Misra et al. [18] have also conducted the simulation using the wellknown NS simulator [19]. They claim their model agrees with the simulation results excellently.

Hollot et al. [20] present a control theoretical analysis of RED on the basis of the differential equations given in Misra et al. [18]. Through the linearization of the differential equations, the model becomes a linear system. In the analysis diagram, the RED router becomes a block with queue length input and drop probability output. After applying the stable conditions of linear feedback system, the authors present the stable condition for the TCP traffic over the RED router. The authors also describe stable point of the system by setting the first derivative in the differential equations to zero. The authors also analyzed the roles of the RED parameters and tradeoffs in the parameters choices.

They support their analysis by NS simulation, which is nonlinear in nature. Throughout the simulation, they show that by selecting RED parameters to satisfy the derived stable conditions, the queue length in the RED router can be stablized. In other words, no oscillation of the queue length will happen. They also do the simulations to show the impact of the RED parameters choices on the response time and robustness.

Based on the seminal work [18], a number of research efforts are carried out. Ren et al. [21] proposed the gental-RED which is proven more stable than RED. Li et al. [22] proposed a nonlinear dynamic model combined TCP and UDP under RED and investigated system stability.

In comparison with previous works, our approach is more general. The proposed AQM is designed to be class-based. Traffic belonging to different classes is controlled differently to maximize overall utility.

2.2 Realtime Scheduling

A number of scheduling schemes have been studied that provide various levels of timing isolation across multiple flows or classes of traffic.

Weighted Fair Queuing was first presented in Demers et al. [23]. The nonpreemptive version Weighted Fair Queuing algorithm is proposed for packet network scheduling.

In Weighted Fair Queuing, packets from different connections are stored in a FIFO queue. The scheduler computes the finish number for every packet when the packet arrives and associates every packets with their own finish numbers .

The scheduler maintains a priority queue. Every connection with packets in the router has one and only one entry in the priority queue. The entry records the information of the first waiting packets of a connection. The priority queue is ordered by the finish numbers of the entries. The packet which has the smallest finish number in the priority queue gets the chance to be transmitted.

The finish number is computed on the basis of the weight of the connection and if the associated packet is the first packet of a busy interval. The finish number calculation makes sure that the bandwidth is fairly shared between connections according to their weights.

The round robin scheduling algorithm is a well known algorithm and widely implemented in many operation systems. In round robin algorithm, every waiting job gets service in the round robin fashion.

The major difference between weighted round robin algorithm and traditional round robin algorithm is that in weighted round robin algorithm, different jobs can get different service time during a round. The major advantage of weighted round robin algorithm over the

weighted fair queuing algorithm is there is no need to maintain a priority queue in weighted round robin algorithm.

The weighted round robin algorithm is suitable to packet switch network and has already been used in ATM networks according to [24]. There are also several variants of weighted round robin algorithm such as Stop and Go algorithm proposed in [25], Hierarchical Round Robin algorithm proposed in [26] and Budgeted Weighted Round Robin algorithm proposed in [27].

In our model, the traffic isolation scheduling mechanism can be chosen from Weighted Round Robin or Weighted Fair Queuing as proposed or analyzed in [23], [28], [25], [29], [26].

3. System Model and Problem Definition

The system model is as in **Fig. 2**. In our scheme, we design a multi-class RED control block. It is composed of several RED queues. Different RED queues will have different RED parameters such as maximum and minimum thresholds, slope and weighting factor.

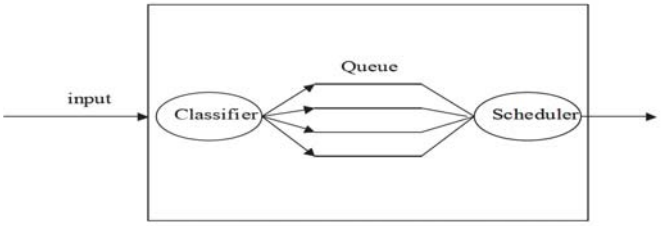

Fig. 2. System Model

So the whole picture of our scheme is as following: When the traffic enters the router, it will be classified into one of the RED queues in the multi-class RED control block. Different queues will have different bandwidth assignment. The bandwidth assignment is determined according to the result of our NLP optimization program. When the change of traffic arriving probability crosses a certain threshold, the optimization program will re-run and then the bandwidth will be re-assigned to different classes. Obviously, there is a tradeoff in selecting the threshold. Small threshold will cause more precise control, that is, the system can adapt to the traffic changes more quickly, but it will cause the transient phase happens more often.

In general, traffic from different applications can be classified by port numbers. For example, telnet packets, ftp packets, ssh packets, and http packets can be classified based on server-side port numbers. Bernaille *et al.* [31] proposed a technique to classify packets on the fly. Advanced classification algorithms such as [32] classify traffic by exploiting the distinctive characteristics of applications. In this paper, we assume the use of the previous proposed classifiers to focus on application-aware queue management schemes.

For traffic from different applications, the utility function will be a function of traffic loss rate due to the reason presented in the introduction section. We represent it by $U_{gi}(\delta_i)$. i means the ith class of traffic. δ_i here means traffic loss rate of the ith class of traffic. Different applications will have different utility functions.

The total utility will be the sum of the utility contributed by all the classes of traffic. We can express it as following:

$$U_{totoal} = P_1^a(1-\delta_1)U_{g1}\delta_1 + P_2^a(1-\delta_2)U_{g2}\delta_2 + \cdots + P_n^a(1-\delta_n)U_{gn}\delta_n \quad (1)$$

where P_i^a represents the arriving probability of *i*th class of traffic and *n* is the number of classes.

Now let's take into account of the network resource constraints. Suppose the network bandwidth is the bottle-neck right now. Assume the bandwidth assigned to the *i*th class traffic is c_i, c_i will be a function of traffic loss rate δ_i. We can express this as:

$$c_i = f_i(\delta_i) \tag{2}$$

Function f_i will depend on the application dynamics. It describes the relationship between bandwidth and traffic loss rate when the system is in the equilibrium state.We will further introduce the function and how to get this function in later sections.

Apparently, the sum of the bandwidth assigned to all the classes of traffic should be no greater than the total bandwidth available. So

$$\sum_{i=0}^{n} c_i \le C \tag{3}$$

where C is the link bandwidth.

Since δ_i is the traffic loss rate or drop probability for the i*th* class,

$$0 \le \delta_i \le 1 \tag{4}$$

for i = 0, 1, ... n.

Now our problem becomes to find the optimal traffic drop vector $[\delta_1, \delta_2, \cdots, \delta_n]$ to maximize the total utility expressed in Equation (1) under the constraints of Inequalities (3) and (4). It is a constrained optimization NLP problem. We can use Lagrange Multiplier and non-negative Kuhn-Tucker conditions to solve it. In our simulation we use Matlab to solve the optimization problem.

Note: Since up to now, we still have not touched the stable condition which can make the whole system stable, the constraints listed above is incomplete. We will discuss the stable condition constraints in the following sections.

4. The Scheme in Case of Precise Traffic Classification

In the following, we will first study one node case and then network case.

4.1 One Node Case

In this section, we will consider a simple case, where:

- Network: one node (i.e. router or switch)
- Traffic: two classes (one is TCP, and the other is Simple-UDP (SUDP) [1]).

We will first describe how to complete the constrained optimization problem in this simple case.

1) Differential Equations for SUDP and TCP Applications: In this subsection, we derive the differential equations of SUDP and TCP applications.

a) SUDP applications: They represent a type of video or audio applications which are based on UDP, and have the following end-to-end control behaviors:

- Once sender gets a packet loss indication, it will send β packets retransmission.

[1]We will describe its features later.

- The relation between playback buffer occupation at the receiver and the sender transmission rate is as in **Fig. A.10**. (Assume the sender will adjust the transmission rate according to the feedback from receiver about the playback buffer occupation)
- Assume the sender is sending CBR video, audio or other data. So at the receiver side, the playback buffer is drained at a constant rate *R*.

From the above behaviors, we can get the following differential equations to describe the dynamics of this class of traffic. Please refer to Appendix A for more details.

$$-d\lambda_{1i}(t+\tau)\frac{q_{max}}{\lambda_{max}} = -\frac{dt}{R} + (1-p(t)+\beta p(t))\lambda_{1i}(t)dt \tag{5}$$

where τ, q_{max}, and λ_{max} represents half RTT time, size of playback buffer, and maximum sending rate respectively. We use $p(t)$ and $\lambda_{1i}(t)$to denote the drop probability function for the class and sending rate function for *i*th flow.

$$\frac{dq(t)}{dt} = -c + \sum_{i=0}^{n}\lambda_{1i}(t) \tag{6}$$

where $q(t)$ represents the queue size function for the class.

b) TCP applications: The differential equations to describe TCP behavior have been derived in [18]. The equations are listed as follows:

$$\frac{dW_i(t)}{dt} = \frac{1}{R_i(t)} - \frac{W_i(t)W_i(t-R_i(t))}{2R_t(t)}p(t-R_i(t)) \tag{7}$$

$$\frac{dq(t)}{dt} = -c + \sum_{i=1}^{n}\frac{W_i(t)}{R_i(t)} \tag{8}$$

where $W_i(t)$ and $R_i(t)$ denote the congestion window size and round trip time functions for *i*th flow. Again we use $p(t)$ and $q(t)$ to denote the drop probability and queue size function for the class.

2) Derivation of Stable Conditions for SUDP System and TCP System: The importance of stability to RED systems relies on the following facts:

- According to [12], the performance of the RED system is the same as one of the drop-tail system, if the RED system is not stable, and the load is around 90-100%. The reason is that if the system is unstable, there are a lot of oscillations. The oscillations can cause more queue overflows and subsequent retransmission than the stable conditions.
- Stability will lead to small jitter and smaller buffer required at the receiver. To gain the advantages of the stability, we have to derive the stable conditions of systems. The steps of deriving these conditions are that first to describe the application traffic behavior into differential equations, then to obtain the transfer functions of the system, finally to use the control method to analyze the whole system.

a) The transfer functions of SUDP system: The transfer functions can be derived as below. Please refer to Appendix A for derivation.

$$P_{sudp}(s) = \frac{\frac{\lambda_{max}}{q_{max}}(\beta-1)\lambda_0}{s+\frac{\lambda_{max}}{q_{max}}(1+(\beta-1)p_0)} \tag{9}$$

$$P_{queue}(s) = \frac{N}{s} \tag{10}$$

b) The transfer functions of TCP system: They have been derived in [20]. They are listed as follows:

$$P_{tcp}(s) = \frac{\frac{R_0 C^2}{2N^2}}{s + \frac{2N}{R_0^2 C}} \tag{11}$$

$$P_{queue}(s) = \frac{\frac{N}{R_0}}{s + \frac{1}{R_0}} \tag{12}$$

Then we can get the system diagram in **Fig. 3**. We need to find out the range of RED router parameters to make the system stable. There are two ways to analyze this system:

- Multiple input and multiple output system: Since the phase problem is also included in this model, the analysis of this system will be complex and the system is a nonlinear system.
- A simplified way: First decouple the two loops. For loop1, we can think q_2 input to the $C_1(s)$ block is just an offset, if we can make sure the loop2 is working in stable condition. So we need to find out the parameter range to make the system stable. We adopt the simplified way for its simplicity and robustness.

c) SUDP loop: Given the transfer function of the RED control derived in [20], the transfer functions of RED control used in SUDP loop can be written as:

$$C_2(j\omega) = \frac{L_{red-dup}}{\left(\frac{j\omega}{K_{udp}} + 1\right)} \tag{13}$$

So the loop transfer function of SUDP system can be derived as follows:

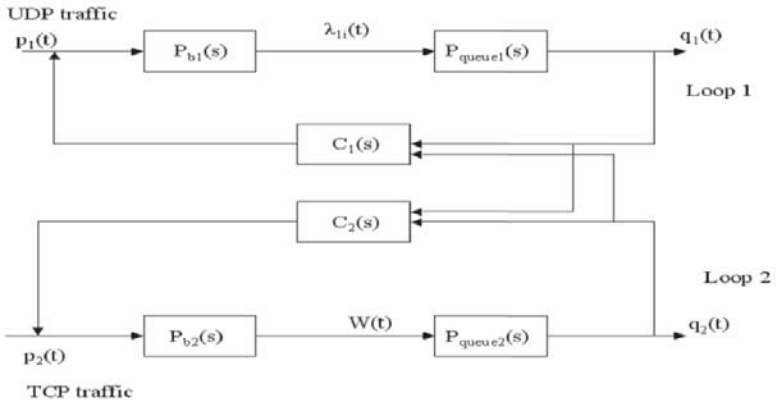

Fig. 3. UDP and TCP Control Model

$$L(j\omega) = P_b(j\omega) P_{queue}(j\omega) C_2(j\omega) e^{-j\omega\tau} = \frac{L_{red-udp} \frac{\lambda_{max}}{q_{max}} (\beta - 1)\lambda_0 N e^{-j\omega\tau}}{\left(\frac{j\omega}{K_{udp}} + 1\right)\left(j\omega + \frac{\lambda_{max}}{q_{max}}(1 + (\beta - 1))p_0\right) j\omega}$$

where $L_{red-udp} = \frac{P_{max-udp}}{th_{max-udp} - th_{min-udp}}$, $K_{udp} = \frac{\log_e(1-\alpha_{udp})}{\delta}$. The parameters α_{udp} and δ are the weighting factor and sample time for the class respectively. We use N, λ_0, and p_0 to denote the number of flows in the class, the sending rate and drop probability in equilibrium state respectively.

To keep the SUDP loop stable, we need to make sure $|L(j\omega)| < 1$ and $\angle L(j\omega) > -180°$. So

$$\frac{L_{red-udp}\frac{\lambda_{max}}{q_{max}}(\beta-1)\lambda_0 N}{\omega_g\sqrt{\frac{\omega_g^2}{K_{udp}^2}+1}\sqrt{\omega_g^2+\frac{\lambda_{max}^2}{q_{max}^2}(1+(\beta-1)p_0)^2}} < 1 \tag{14}$$

where $\omega_g = 0.1\min\left\{K_{udp}, \frac{\lambda_{max}}{q_{max}}(1+(\beta-1)p_0)\right\}$ and for $\forall\omega\in[0,\omega_g]$,

$$\angle L(j\omega) \approx \angle \frac{L_{red-udp}\frac{\lambda_{max}}{q_{max}}(\beta-1)\lambda_0 N e^{-j\omega\tau}}{j\omega\frac{\lambda_{max}}{q_{max}}(1+(\beta-1)p_0)} \tag{15}$$

$$\angle L(j\omega) = \angle\frac{1}{jw_g} - \omega_g\tau \approx -90° - 0.1\frac{-180°}{\pi} > -180°$$

d) TCP loop: For the TCP part, the range is given in Hollot et al. [20]. It is listed as follows:

$$\frac{L_{red-tcp}(R^+C)^3}{(2N^-)^2} \le \sqrt{\frac{w_g^2}{K_{tcp}^2}+1} \tag{16}$$

where $w_g = 0.1\min\left\{\frac{2N^-}{(R^+)^2C}, \frac{1}{R^+}\right\}$ and $L_{red-tcp}$ and K_{tcp} are defined similarly as those for SUDP loop. The parameter N^- and R^+ denote the lower limit of the number of TCP flows in the class and upper limit of round trip time.

e) Stable points: Having got the stable conditions of the system, we need to find out the stable point. The stable points can be got from the differential equations. Please refer to the appendix for details. For UDP traffic, the stable point is as following:

$$\frac{1}{R} = ((\beta-1)p_0+1)\lambda_0 \tag{17}$$

$$C_{udp} = N\lambda_0 \tag{18}$$

where C_{udp} represents the bandwidth assigned to the SUDP class.

For TCP traffic, the stable point is given out in paper [20] as following:

$$W_0^2 p_0 = 2 \tag{19}$$

$$W_0 = \frac{R_0 C_{tcp}}{N} \tag{20}$$

where W_0 and C_{tcp} represents the congestion windows size in equilibrium state and the bandwidth assigned to the TCP class.

Through simulation, we find Equation (19) is not accurate enough. So we re-examine the equilibrium state of TCP traffic and get the following equations to describe the stable point,

$$w_0 = p_0\left(\frac{(2n-2)(2n-1)(4n-3)}{6} - \frac{(n-2)(n-1)(2n-3)}{6} + 2(n-1)^2\right) \tag{21}$$

$$\frac{1}{p_0} = (n-1)\left(\frac{3}{2}n + 2\right) \tag{22}$$

$$W_0 = \frac{R_0 C_{tcp}}{N} \tag{23}$$

where v is the difference between maximum congestion windows size and minimum congestion window size in equilibrium state. The relation between ω_0 and p_0 can be simplified into one equation by combining Equation (21) and (22). Detailed derivation of stable points equations is available in Appendix B.

These stable points equations are correspondent to f_i in Equation (2).

3) *The Constrained Optimization Program:* The previous section gives out how to make the system stable. Now we can return to the optimization problem defined in model section.

With the addition of the system stable constraints (14) and (16), now we get a complete set of constraints of our optimization problem. To solve the optimization problem, we still need to know the arriving probability p_i^a in our object function (1). The arriving probability can be got from information collected by classifier or inferred from the queue size of different classes.

Our aim is to achieve maximum total satisfaction or utility. To achieve our aim, we need to make sure that the system will be stable at point where maximum satisfaction or utility is got. From the relation of stable point in above Equations (17), (18), (21), (22) and (23), it is obvious that we can change the bandwidth assignment to adjust the stable point so that the stable point is also the optimal point.

So after we determine the optimal drop probability for each class of traffic by solving the constrained optimization problem, we can allocate bandwidth to each class of traffic from the above stable point equations.

4.2 Network Case

Using differential equations to describe a class of application traffic will cause the aggregation error. How can we reduce the aggregation error to the equations more accurate? Our approach is to classify the traffic based on the geographic information because traffic which has similar routes will have similar traffic parameters such as round trip time for TCP traffic. Thus more classes, more accurate the equations are. Ideally if we classify every flow into one class, our scheme is similar to per-flow queuing scheme with optimization scheme. So depending on how precisely we want to get the optimality, we can decide how many classes we want to have in the system.

5. Performance Evaluation

5.1 Simulation Setup

We use topology shown in **Fig. 4** in our simulations. The bandwidth and delay of link between hosts and routers are 10 Mbit/s and 1 ms respectively. The bandwidth and delay of link between routers are 1 Mbit/s and 1 ms. We did extensive simulations on the proposed RED

control. Since the maximization of total utility is guaranteed by the optimization process in the proposed approach, our experiments below focus on stability and stable points. Due to the space limit, we show typical experiment results below. More experiment results are available at [30].

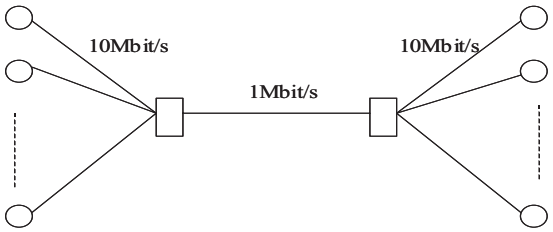

Fig. 4. Simulation Topology

5.2 Stability

We did extensive experiments on controlling different amount of traffic flows by the proposed RED control. Typical experiment results for different amount of traffic are shown in **Fig. 5**, **Fig. 6**, and **Fig. 7**. In **Fig. 5**, **Fig. 6**, and **Fig. 7,** the number of TCP and SUDP flows (denoted as N) is 10, 30, and 60 respectively. The experiment results of sixty TCP flows with sixty SUDP flows are shown in **Fig. 7**. In **Fig. 7(a)**, both average queue length and current queue length are shown. In **Fig. 7(c)**, both congestion windows size and slow-start threshold of a typical TCP flow are shown. We can observe under the proposed RED control, the stability can be maintained.

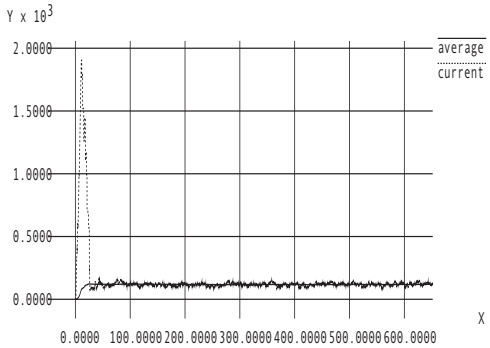

Fig. 5 (a) Queue Length (Y: Queue Length X: Time in Second, N=10)

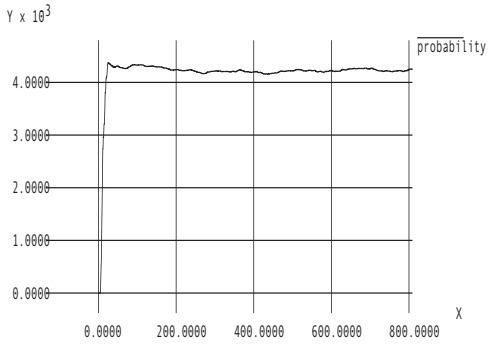

Fig. 5 (b) Drop Probability (Y: Drop Probability, X: Time in seconds, N=10)

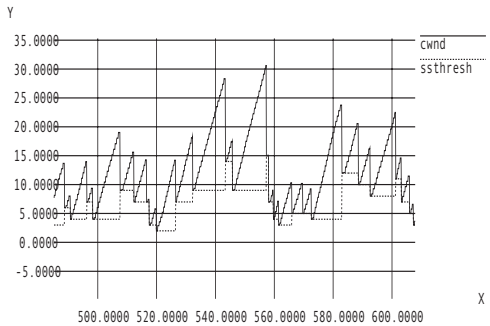

Fig. 5 (c) Congestion Window Size (Y: Window Size, X: Time in Second, N=10)

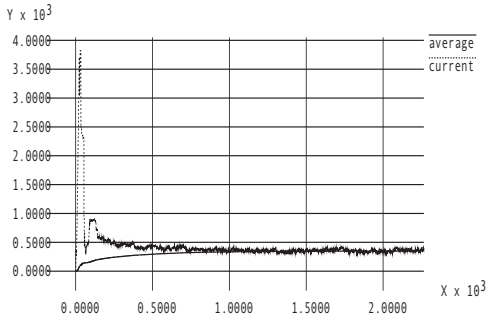

Fig. 6 (a) Queue Length (Y: Queue Length, X: Time in seconds, N=30)

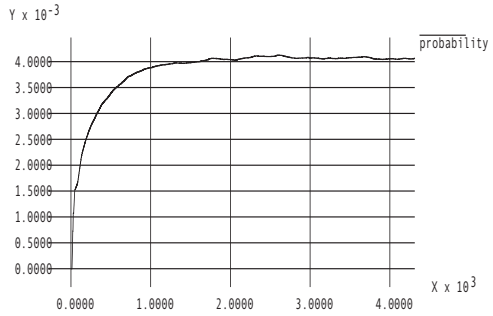

Fig. 6 (b) Drop Probability (Y: Drop Probability, X: Time in Second, N=30)

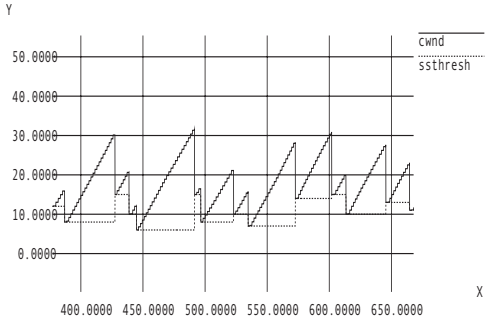

Fig. 6 (c) Congestion Window Size (Y: Window Size, X: Time in seconds, N=30)

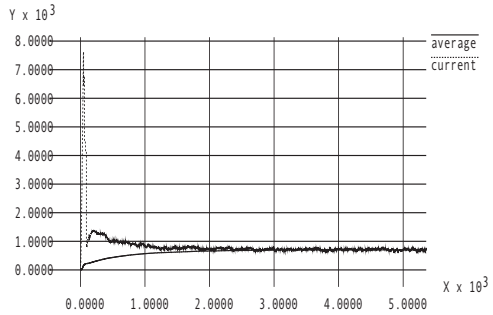

Fig. 7 (a) Queue Length (Y: Queue Length X: Time in Second, N=60)

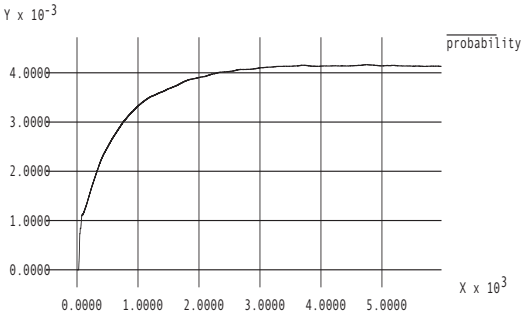

Fig. 7 (b) Drop Probability (Y: Drop Probability, X: Time in seconds, N=60)

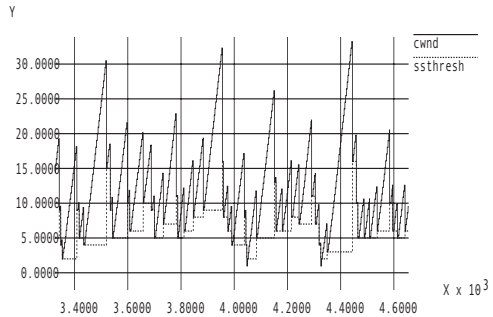

Fig. 7 (c) Congestion Window Size (Y: Window Size X: Time in Second, N=60)

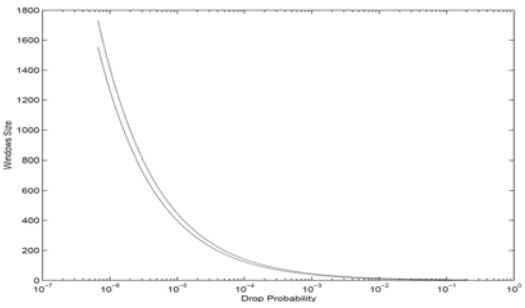

Fig. 8. Comparison of Window Size vs. Drop Probaility Relationship

From **Fig. 5**, **Fig. 6**, and **Fig. 7**, we can also observe that transient phase is longer when more traffic flows going through the system. In other words, it takes more time for the system to get into stable states. We beleive it is because that it takes more time to notify enough traffic flows to slow down by packet drops when lots of traffic flows are going through the system. So there are too many traffic flows classified into one class, it is necessary to split the class to reduce the length of transient phase.

5.3 Stable Points

Table 1. shows the difference among simulation result and expected results from our method and Hollot's method about drop probability in equilibrium state. **Table 2**. shows the difference among simulation result and expected results from our method and Hollot's method about queue size in equilibrium state.

Table 1. Stable Drop Probability Comparison

Flow Numbers	1	2	5	10	30	60
Simulation Result	0.004	0.0041	0.00417	0.0042	0.00415	0.00424
Our Method	0.005	0.005	0.005	0.005	0.005	0.005
Hollot's Method	0.0068	0.007	0.0066	0.0067	0.0064	0.0066

Table 2. Stable Queue Size Comparison

Flow Numbers	1	2	5	10	30	60
Simulation Result	10	22	58	117	357	710
Our Method	12.425	26.35	68.125	137.75	416.25	834
Hollot's Method	15.61	32.4	85.25	171.15	528.17	1044.72

The difference between our method and Hollot's method is about the relationship between windows size and drop probability in equilibrium state. So we compares these two methods in the **Fig. 8**. In **Fig. 8**, Hollot's curve is above our curve. We can also observe that when the drop probability in equilibrium state becomes smaller, the difference between these two curves become larger. We believe it is because in equilibrium state, the difference between maximum window size and minimum window size becomes larger when drop probability in equilibrium state becomes small.

6. Extension and Conclusion

In this paper, we propose an approach to maximize utility by optimizing RED control. To make our approach scalable, it is designed to be class-based. Traffic belonging to different classes is controlled differently to maximize overall utility. We verify the proposed approach through extensive experiments.

Our scheme can be extended to deal with the DDOS attacks. When the arriving probability of a certain class of traffic increases significantly, we can further split the class into mulitple classes. As we mentioned before, if there are more classes, we can get more utility but at the cost of more queue management overhead in normal case. For DDOS attack case, we can further split the classes until the attack flows are roughly isolated. Then this isolated DDOS traffic will get a large drop probability after calculation of our optimization problem because the utility function of DDOS traffic will be negative.

References

[1] 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," United States, 1998.

[2] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Trans. Netw.*, vol. 1, no. 4, pp. 397-413, 1993.

[3] M. May, T. Bonald, and J.-C. Bolot, "Analytic evaluation of RED performance," in *Proc. of INFOCOM 2000*, Tel-Aviv, Israel, Mar. 2000.

[4] W. Feng, D. Kandlur, D. Saha, and K. G. Shin, "BLUE: A new class of active queue management algorithms," Tech. Rep. CSE-TR-387-99, 15, 1999.

[5] T. J. Ott, T. V. Lakshman, and L. H. Wong, "Sred: Stabilized red," in *Proc. of INFOCOM*, pp. 1346-1355, 1999.

[6] L. Zhu, G. Cheng, and N. Ansari, "Local stability of random exponential marking," *IEE*

Proceedings-Communications, vol. 150, no. 5, pp. 367-370, 2003.
[7] L. Tan, G. Peng, and S. Chan, "Adaptive rem: random exponential marking with improved robustness," *Electronics Letters*, vol. 43, no. 2, pp. 133-135, 2007.
[8] R. Pan, B. Prabhakar, L. Breslau, and S. Shenker, "Approximate fair allocation of link bandwidth," *IEEE Micro*, vol. 23, no. 1, pp. 36-43, 2003.
[9] A. Tang, J. Wang, and S. Low, "Understanding choke: throughput and spatial characteristics," *IEEE/ACM Transactions on Networking*, vol. 12, no. 4, pp. 694-707, Aug. 2004.
[10] S. Floyd, "Red: Discussions of setting parameters," http://www.aciri.org/floyd/REDparameters.txt.
[11] M. May, J. Bolot, C. Diot, and B. Lyles, "Reasons not to deploy RED," in *Proc. of 7th. International Workshop on Quality of Service (IWQoS'99), London*, pp. 260-262, June 1999.
[12] M. Christiansen, K. Jeffay, D. Ott, and F. D. Smith, "Tuning red for web traffic," *SIGCOMM Comput. Commun. Rev.*, vol. 30, no. 4, pp. 139-150, 2000.
[13] D. D. Clark and W. Fang, "Explicit allocation of best-effort packet delivery service," *IEEE/ACM Trans. Netw.*, vol. 6, no. 4, pp. 362-373, 1998.
[14] CISCO Systems, Inc., "Distributed weighted random early detection," http://www.cisco.com/univercd/cc/td/doc/product/software/ios111/cc111/wred.pdf.
[15] R. Jain, K. K. Ramakrishnan, and D.-M. Chiu, "Congestion avoidance in computer networks with a connectionless network layer," pp. 140-156, 1988.
[16] V. Jacobson, "Congestion avoidance and control," *SIGCOMM Comput. Commun. Rev.*, vol. 25, no. 1, pp. 157-187, 1995.
[17] D.-M. Chiu and R. Jain, "Analysis of the increase and decrease algorithms for congestion avoidance in computer networks," *Comput. Netw. ISDN Syst.*, vol. 17, no. 1, pp. 1-14, 1989.
[18] V. Misra, W.-B. Gong, and D. Towsley, "Fluid-based analysis of a network of aqm routers supporting tcp flows with an application to red," in *SIGCOMM '00: Proceedings of the conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*. New York, NY, USA: ACM, pp. 151-160, 2000.
[19] Information Sciences Institute, "The network simulator," http://www.isi.edu/nsnam/ns/index.html.
[20] C. V. Hollot, V. Misra, D. F. Towsley, and W. Gong, "A control theoretic analysis of red," in *INFOCOM*, pp. 1510-1519, 2001.
[21] F. Ren, C. Lin, and B. Wei, "A nonlinear control theoretic analysis to tcp-red system," *Computer Networks*, vol. 49, no. 4, pp. 580-592, 2005. [Online]. Available: http://www.sciencedirect.com/science/article/B6VRG-4FS21DP-1/2/fc2a036a6d2a58b5e33e71be84e66f1b
[22] Z.-z. Li and Y.-p. Chen, "A control theoretic analysis of mixed tcp and udp traffic under red based on nonlinear dynamic model," in *ICITA '05: Proceedings of the Third International Conference on Information Technology and Applications (ICITA'05) Volume 2*. Washington, DC, USA: IEEE Computer Society, pp. 747-750, 2005.
[23] A. Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a fair-queuing algorithm," in *SIGCOMM*, pp. 1-12, 1989.
[24] H. Saran, S. Keshav, and C. R. Kamanet, "A scheduling discipline and admission control policy for xunet2," in *Proc. of the Fourth International Workshop on Network and Operating System Support for Digital Audio and Video*, pp. 89-101, 1993.
[25] S. J. Golestani, "A framing strategy for congestion management," *IEEE Journal on Selected Areas in Communications*, vol. 9, no. 7, pp. 1064-1077, Sep. 1991.
[26] C. Kalmanek, H. Kanakia, and S. Keshav, "Rate controlled servers for very high speed networks," in *Proc. Globecom*, pp. 12-20, 1990.
[27] I. R. Philp, "Scheduling real-time messages in packet-switched networks," Ph.D. dissertation, Champaign, IL, USA, adviser-Jane W. Liu, 1996.
[28] A. K. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control inintegrated services networks: the single-node case," *IEEE/ACM Transactions on Networking*, vol. 1, no. 3, pp. 344-357, June 1993.
[29] M. Shreedhar and G. Varghese, "Efficient fair queueing using deficit round robin," *SIGCOMM Comput. Commun. Rev.*, vol. 25, no. 4, pp. 231-242, 1995.

[30] Y. Zhu, "Optimizing RED Control to Maximize Utility," Technical Report, CSUECE-TR-08-01, Cleveland State University, http://academic.csuohio.edu/zhu_y/pub/trCSU0801.pdf
[31] L. Bernaille, L. Teixeira, R. Akodkenou, I. Soule, and K. Salamatian, "Traffic classification on the fly," *SIGCOMM Comput. Commun.* Rev. 36, 2, Apr. 2006.
[32] J. Erman, M. Arlitt, and, A. Mahanti, "Traffic classification using clustering algorithms," In *Proceedings of the 2006 SIGCOMM Workshop on Mining Network Data,* Pisa, Italy, September 11-15, 2006.

Appendix

Appendix A: SUDP Dynamics Model

The SUDP flow model is as in **Fig. A.9**. In **Fig. A.9**, the symbols definitions are as following:
λ_{1i} : the sending rate of the ith flow from sender
λ_{2i} : the arrival rate of the ith flow at the receiver buffer
q_b : buffer occupation at the receiver playback buffer
c: link speed
q: queue size at the RED router
$p(t)$: drop probability at the RED router

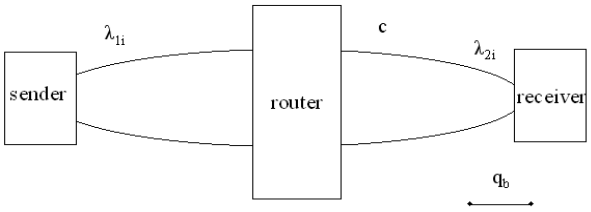

Fig. A.9. SUDP Model 1

The relationship between sender's transmission rate and receiver's playback buffer occupation is as **Fig. A.10**.

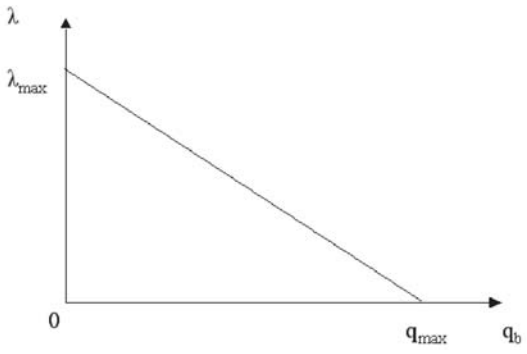

Fig. A.10. SUDP Model 2

So for this model, we can get:

$$\frac{d\lambda(t)}{dq_b(t)} = -\frac{\lambda_{max}}{q_{max}} \tag{A.24}$$

(1) PLAYBACK BUFFER EQUATION

From the point view of the playback buffer size, the size changes can be from three aspects. They are constant consumption of the playback buffer, β retransmission packets due to every packet loss and arrival of new packets respectively. So we can get the following equation:

$$dq_b = -\frac{dt}{R_i} + \beta dN_i + \lambda_{2i}(t)dt = -\frac{dt}{R_i} + \beta\lambda_{loss}(t)dt + \lambda_{2i}(t)dt$$

$\lambda_{loss}(t)$: arrival rate of loss indication Since the traffic is going through the RED router, so

$$\lambda_{loss}(t) = p(t)\lambda_{1i}(t) \tag{A.25}$$

$$\lambda_{2i}(t) = (1 - p(t))\lambda_{it}(t) \tag{A.26}$$

Note: actually there is some delay between drop probability and all λ_s. The delay is considered in the RED model.

Then,

$$dq_b = -\frac{dt}{R_i} + \beta p(t)\lambda_{1i}(t)dt + (1 - p(t))\lambda_{1i}(t)dt = -\frac{dt}{R_i} + (1 - p(t) + \beta p(t))\lambda_{1i}(t)dt$$

Together with Equation A.24 we can get:

$$-d\lambda_{1i}(t)\frac{q_{max}}{\lambda_{max}} = -\frac{dt}{R_i} + (1 - p(t) + \beta p(t))\lambda_{1i}(t)dt \tag{A.27}$$

Since the feedback from the receiver to the sender need to take time, we model this delay as half round trip time RTT). So we get:

$$-d\lambda_{1i}(t+\tau)\frac{q_{max}}{\lambda_{max}} = -\frac{dt}{R_i} + (1 - p(t) + \beta p(t))\lambda_{1i}(t)dt \tag{A.28}$$

τ: half RTT time

(2) RED BUFFER EQUATION

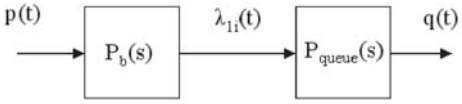

Fig. A.11. SUDP Behavior Model

At the point view of the buffer size at the RED router, the change of the buffer size can be from 2 aspects. They are constant departure from the router when congested and arrival from the sender. So we can get the following equation:

$$\frac{dq(t)}{dt} = -c + \sum_{i=1}^{n}\lambda_{1i}(t) \tag{A.29}$$

From the playback buffer equation and RED router equation, we model the UDP behavior dynamics as following:

$P_b(s)$ can be got from playback buffer equation.

$P_{queue}(s)$ can be got from the RED buffer equation.

(3) SUDP TRANSFER FUNCTION

For UDP loop, the equilibrium state is got when $\frac{d\lambda(t)}{dt} = 0$ and $\frac{dq(t)}{dt} = 0$

From the playback buffer equation and RED buffer equation, we can get the equilibrium state:

$$\frac{1}{R} = ((\beta - 1)p_0 + 1)\lambda_0 \tag{A.30}$$

$$c_{udp} = N\lambda_0 \tag{A.31}$$

We can simplify the UDP playback buffer equation and RED buffer equation as following:

$$-\frac{q_{max}}{\lambda_{max}}\frac{d\,\lambda_1(t)}{dt} = -\frac{1}{R} + (1 + (\beta - 1)p(t-\tau))\lambda_1(t-\tau) \tag{A.32}$$

$$\frac{dq(t)}{dt} = -c + N\,\lambda_1(t) \tag{A.33}$$

Then we do linearization, let

$$f(\lambda_1, q, p) = -\frac{1}{R} + (1 + (\beta - 1)p(t-\tau))\lambda_1(t-\tau) \tag{A.34}$$

$$g(\lambda_1, p) = -c + N\,\lambda_1(t) \tag{A.35}$$

At the equilibrium state (λ_0, p_0, q_0), we can get

$$\frac{\partial f}{\partial \lambda_1} = (\beta - 1)p_0 + 1 \tag{A.36}$$

$$\frac{\partial f}{\partial p} = (\beta - 1)\lambda_0 \tag{A.37}$$

$$\frac{\partial f}{\partial q} = 0 \tag{A.38}$$

$$\frac{\partial g}{\partial \lambda_1} = N \tag{A.39}$$

$$\frac{\partial g}{\partial q} = 0 \tag{A.40}$$

So around the equilibrium state, we can get

$$\delta\lambda_1(t) = -\frac{\lambda_{max}}{q_{max}}(1 + (\beta - 1)p_0)\delta\lambda_1(t-\tau) - \frac{\lambda_{max}}{q_{max}}(\beta - 1)\lambda_0\,\delta p(t-\tau) \tag{A.41}$$

$$\delta q(t) = N\delta\lambda_1(t) \tag{A.42}$$

After Laplace transform, we can get

$$P_b(s) = \frac{\frac{\lambda_{max}}{q_{max}}(\beta - 1)\lambda_0}{s + \frac{\lambda_{max}}{q_{max}}(1 + (\beta - 1)p_0)} \tag{A.43}$$

$$P_{queue}(s) = \frac{N}{s} \tag{A.44}$$

So the complete loop transfer function

$$L(j\omega) = P_b(j\omega)P_{queue}(j\omega)C_2(j\omega)e^{-j\omega\tau}$$

$$= \frac{L_{red-udp}\,\frac{\lambda_{max}}{q_{max}}(\beta - 1)\lambda_0 N\,e^{-j\omega\tau}}{\left(\frac{j\omega}{k} + 1\right)\left(j\omega + \frac{\lambda_{max}}{q_{max}}(1 + (\beta - 1)p_0)\right)j\omega}$$

Appendix B: TCP Traffic Stable Point

Through extensive simulations on the stable point, we find the stable point equation given out in Hollot et al. [20] is not accurate enough.

So we have tried to find out the reason why the stable point is not accurate. We think the inaccuracy is from two factors. One is about the first differential equation which describes the behavior of the window size given out in Misra et al. [18]. The equation is based on the additive-increase multiplicativedecrease behavior of TCP. But this equation has omitted the fact that when a timeout event happens, the congestion window size will decrease to one. In our simulation, we find really there are timeout events when the RED system is in the

equilibrium state.

The other one is about the deduction in Hollot et al. [20] of the stable point. From the angle of control theory, it is straightforward to find the stable point by setting change of queue size with time to zero, that is $\frac{dq}{dt} = 0$. But for a TCP flow in equilibrium state, the congestion window can change from the maximum to half the maximum when a packet drop occurs. Then the congestion window size will increase step-by-step to the maximum. So we can not just simply substitute the w with W_0 when finding the stable point equation.

Let's first look at the one flow case. Suppose now the RED router is table at drop probability p_0. So on the average, there will be a packet drop every $\frac{1}{p_0}$ packets. To simplify the analysis, we assume the packet drop happens every $\frac{1}{p_0}$ packets. Then the congestion window changes periodically as **Fig. B.12** in equilibrium state.

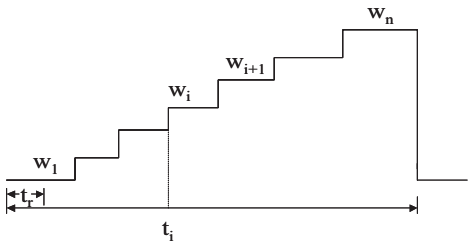

Fig. B.12: Congestion Window Dynamics in Equilibrium State

Here we assume W_0, the equilibrium state window size is the average of the window size over time in a period T. So $W_0 = \frac{1}{T}\int_0^T w(t)dt$. Since the system is in the equilibrium state, the queue size should not build up at the end of a period. So these $\frac{1}{p_0}$ packets should be digested in time $\frac{1}{p_0 c}$. In another word, $T = \frac{1}{p_0 c}$.

$\int_0^T w(t)dt$ is the area under the w(t). So it can be computed as $\sum_{i=1}^{n} w_i r_i$. r_i the round trip time in the ith interval. Here we assume that the r_i does not change in the ith interval for simplification.

Now let's look at the time t_i. Suppose the queue length at time t_i is q_i. So at the end of the ith time interval, these q_i packets will just leave. As we all know, in every round trip time, the congestion window increase by one in congestion avoidance phase. So in the ith interval, the sender will receive w_i acknowledges, which means w_i packets will be sent out from the sender. So the queue size at the end of ith interval will be around w_i. In the (i+1)th interval, we can assume the queue size is around $w_i + 1$. Then round trip time r_i is around $\frac{q_i}{c}$. (c is the bandwidth.)

The window size falls from the maximum to half maximum when the sender gets the duplicate acknowledges for Reno TCP. Then the window size will freeze about one round trip

time t_r before it begins to increase again. So t_r is around $\frac{w_n}{c}$. Since w_1 is equal to a half of w_n and, $w_{i+1} = w_i + 1$, $w_n = 2(n-1)$ and $w_1 = n-1$.

From above, we can get

$$\begin{aligned} W_0 &= \frac{1}{T}\int_0^T w(t)dt \\ &= \frac{1}{T}\left(\sum_{i=1}^{n} w_i r_i + t_r\right) \\ &= \frac{1}{\frac{1}{p_0 c}}\left(\sum_{i=1}^{n} w_i \frac{w_i}{c} + w_1 \frac{w_n}{c}\right) \\ &= p_0\left(\sum_{i=n-1}^{2(n-1)} i^2 + 2(n-1)^2\right) \\ &= p_0\left(\frac{(2n-2)(2n-1)(4n-3)}{6} - \frac{(n-2)(n-1)(2n-3)}{6} + 2(n-1)^2\right) \end{aligned}$$

So we get

$$W_0 = p_0\left(\frac{(2n-2)(2n-1)(4n-3)}{6} - \frac{(n-2)(n-1)(2n-3)}{6} + 2(n-1)^2\right) \quad \text{(B.45)}$$

Since T is composed of n round trip time and a retransmission time, we can have the following equation:

$$\begin{aligned} T &= \sum_{i=1}^{n} r_i + t_r \\ &= \sum_{i=1}^{n} \frac{w_i}{c} + \frac{w_n}{c} \\ &= \sum_{i=n-1}^{2(n-1)} \frac{i}{c} + \frac{2(n-1)}{c} \\ &= \frac{(n-1)\left(\frac{3}{2}n + 2\right)}{c} \end{aligned}$$

Since $T = \frac{1}{p_0 c}$, we can get

$$\frac{1}{p_0} = \frac{(n-1)\left(\frac{3}{2}n + 2\right)}{c} \quad \text{(B.46)}$$

Our simulations show that equations (B.45) and (B.46) are more accurate than the stable point equation given in Hollot et al. [20].

(1) MULTIPLE FLOWS CASE

When multiple flows share one link, according to [17], the multiplicative-decrease and additive-increase mechanism of TCP protocol will lead to a fair and stable state. Since our drop policy is fair to every flow in a certain class, the flows within one class of traffic will eventually get into a fair state too. Same as the one flow case, in the equilibrium state, the packets will only be dropped by the RED drop policy. There will be no queue overflow drop if we can make the system stable at the dynamic region (between maximum threshold and minimum threshold). Thus we can think n flows sharing a link whose bandwidth is c as one flow occupying a link whose bandwidth is $\frac{c}{n}$. Since equations (B.45) and (B.46) are irrelevant with flow bandwidth, they are also suitable for multiple flows case. If we follow the steps in the one flow case, we will also get these two equations for multiple flows case.

Ye Zhu is an Assistant Professor in Department of Electrical and Computer Engineering at Cleveland State University. He was a research assistant in the NetCamo research group and received his Ph.D. in Electrical and Computer Engineering Department from Texas A&M University. He received his B.Sc. from Shanghai JiaoTong University and his M.Sc. from Texas A&M University. His current research interests include QoS, traffic engineering, anonymous communication, network security, and peer to peer networking. He published his research results in premier journals such as ACM TISSEC and IEEE TPDS and prestigious conferences such as ICDCS.