



## Providing QoS to TCP and Real Time Connections in the Internet

V. VENUGOPAL REDDY, VINOD SHARMA and M.B. SUMA\* vinod@ece.iisc.ernet.in  
*Electrical Communication Engineering Department, Indian Institute of Science, Bangalore-560012, India*

Received 10 January 2003; Revised 20 June 2003

**Abstract.** We consider a system where streaming and/or real time applications, using the UDP protocol pass through a bottleneck router. Several persistent and non persistent TCP connections may also be passing through that router. Using the rate control at the UDP sources and choosing the RED parameters appropriately, we obtain pre-specified delays, little delay jitter, no packet loss, 100% channel efficiency, pre-specified throughput for the TCP and UDP streams and little rate fluctuation. We verify our control scheme by extensive simulations. Results are extended to a tandem of queues.

**Keywords:** Internet, real time applications, QoS, rate control

### 1. Introduction

Internet was designed to transmit non-real time data traffic. But recently efforts are being made to transmit real time voice and video also on the Internet (see RFCs [4,6]). However, the Quality of Service (QoS) requirements for these services are quite different from that of the traditional data services (e.g., FTP, e-mail). For example, voice and video can tolerate a little packet loss ( $\sim 1\%$ ) but require upper bounds on delay and delay jitter ( $\leq 200$  msec). The traditional data applications are delay tolerant but not loss tolerant. Therefore tremendous efforts are being made to change the Internet architecture to satisfy the QoS requirements of different services in a cost – effective way [4,6].

Considering the various services using TCP and the real time applications, e.g., IP-Telephony, real-time video, which use UDP (along with, probably RTP, RTCP), certain QoS can be provided for these services if the TCP connections can be guaranteed some minimum throughput and the real-time applications guaranteed upper bounds on the end-to-end delay, delay jitter and probability of packet loss and a minimum (average) throughput (see [8,15]). In addition the network resources should be efficiently utilized.

In this paper we suggest a control scheme for the rate control of real time traffic and window control of the TCP traffic using the current active congestion control mechanisms, e.g., RED (Random Early Detection [12]) implemented in the Internet routers. Various practical codecs for voice and video allow layered coding which can be used for sending voice or video at variable rates. The information about this can be sent via RTP

\* Her work was supported by the DRDO-IISc program on Mathematical Engineering.

to the receiver. Our scheme guarantees a fixed workload in the queue of the bottleneck router. Thus, the channel is fully utilized, there is no packet loss, the delay is at a fixed desired level and there is negligible delay jitter. Thus this is a popular control objective in recent network architectures (see [24]). However, this seems to be the first work where this control objective is being realized in the presence of TCP connections. In addition to the above objectives, we further ensure a guaranteed pre-specified share of throughput for the various TCP and UDP connections. Our control scheme does not require any changes to the Internet protocols or architecture. Also, no per-flow processing is required at the routers and the rate controller at the source requires minimal information.

In this paper we are considering long-lived (persistent) and short-lived (non-persistent) TCP connections depending upon the file size they transmit. Most of the connections in the Internet are short-lived, lasting a few seconds or less. However, there are some very long-lived (lasting several minutes or more) connections transferring large files (e.g., several megabytes) [27]. One more model of TCP, called TCP ON-OFF is also becoming relevant today. In this case a TCP connection is used to transmit several files. This is used in HTTP 1.1 protocol for web browsing. Also, now many web browsers want to use a TCP connection to upload several files. After a file is transmitted, there may not be any more packets to transmit till a new request comes. This makes a TCP ON-OFF model suitable. A performance analysis of a router with non-persistent and TCP persistent ON-OFF connections is provided in [13]. In the set up of our present paper, TCP ON-OFF models have been included in [29]. In the present paper we limit ourselves to just long-lived and short-lived TCP connections.

Now we mention some relevant literature. The TCP has been analyzed in [2,3,5,10,12,17–19,25] (and the references therein). Non-persistent TCP connections have been studied in [7,9]. The rate control of real time streams has been considered in [11,16,21,29,30]. However except [29] the other studies do not try to provide any specific end to end QoS guarantees. We propose control schemes at the routers to provide QoS to all connections passing through them.

The Intserv [6] and Diffserv [4] architectures in Internet are two frameworks proposed to provide QoS. These contain general guidelines on how one may obtain QoS for different connections. In addition to admission control, per flow processing, extensive signalling (in [6]) these require various scheduling policies at the routers (e.g., priority, generalized processor sharing, weighted round robin) and reservation of resources. Our proposal here shows how it may be possible to provide QoS with minimal changes in the Internet: we use FCFS at the routers, no new modifications to the protocol, very little signalling support, no buffer management, no per-flow processing and only one queue per output link. Furthermore it can be possibly used in conjunction with [4] and/or [6] and provide specific QoS guarantees.

The paper is organized as follows. Section 2 explains the model and the notation. Section 3 provides the control scheme to provide QoS. Section 4 presents the simulation results.

## 2. The model and notation

We motivate our model by first discussing the general traffic patterns and characteristics which have been observed by many measurement studies (see, e.g., [27]) on real networks. The average number of packets in a TCP flow is around 20. Most of the flows transmit small files (a few Kbytes) but a few carry very large files (several megabytes). This is in conformity with the heavy-tailed file size distributions found in the Internet. The long files can take several minutes to transmit while short files can be transmitted in a fraction of a second. Also, the traffic statistics can be taken as stationary up to a 5 minute interval [27]. Thus it is reasonable to limit ourselves to a time scale of  $\sim 5$  minutes. During that time the TCP connections transmitting long files can be modelled by persistent TCP connections which always have packets to send and will not terminate. Since such connections will arrive rarely, one can assume them to be a fixed number in our model. Also, if a connection has a long file to send, it will try to make its packets as large as permitted by the network. It will have thousands of packets to transmit, (e.g., a 10 Mbyte file with packet size of 1000 bytes will have 10000 packets). Therefore, for these connections a QoS measure can be their throughput in number of packets successfully transmitted per second. The short-lived TCP connections may have only a few (say  $\leq 20$ ) packets to transmit and their packet lengths can also be small. These connections may be ON for a fraction of a second and then terminate. These short-lived connections make the majority of TCP connections on a link. Hence on a time scale of several minutes, we can model them as a stream (Poisson can be a good assumption) of TCP connections arriving at random times which have small files to transmit and after transmission they terminate. Their file sizes and packet lengths will be random in length (for a particular connection the packet lengths will be constant but it will change from connection to connection). An appropriate QoS parameter for these connections is the mean down-load time of the file.

The number of UDP flows and the fraction of traffic they carry is still comparatively small  $\sim 5\%$ . However it is likely to increase. In addition to other applications, the interactive/streaming audio/video is carried in UDP connections. This audio/video traffic has the most stringent QoS requirements. Thus, in this paper, we will limit ourselves to only such applications while discussing the UDP connections. Such traffic can last upto several minutes (especially say IP telephony, video conferencing). Also, since the connection arrival rate of this traffic is very low, on our time scale, we can consider a fixed number of UDP connections which send their traffic as CBR or VBR streams. Their QoS requirement is an upper bound on end to end delay, delay jitter and fraction of packets lost.

The above considerations, motivate the models studied in this paper. First we study a model with a fixed number of persistent (long-lived) TCP connections and a fixed number of UDP connections. We explain our scheme to provide their QoS requirements. In section 3.4 we will also include the non-persistent (short-lived) TCP connections. Let a real time application (e.g., a real time voice or video) be passing through a route with one bottleneck router. We consider the router with buffers only at the output ports. The

packets generated by the real time application share a buffer at an output port with  $N \geq 0$  other TCP connections passing through the router. The real time application may be generating packets as a CBR (Constant Bit Rate), ON-OFF, Poisson or an MMPP (Markov Modulated Poisson Process) stream. Initially, the TCP connections are assumed to be persistent i.e., they always have packets to send. In section 3.4 we will also consider non-persistent (short lived) TCP connections. Let the packet lengths of the real time application, which uses the UDP protocol and that of TCP connections form *iid* sequences with general distributions. Each sequence is assumed to be independent of the other sequences. Let  $s$  and  $s_T(i)$  be generic service times of the UDP and the  $i$ th TCP connection. All the packets are stored in the same queue and are served in First-In-First-Out (FIFO) fashion. This is a typical scenario in the Internet. Later on we will also consider a multiple number of UDP connections and a tandem of routers.

In the above scenario one can compute the mean delays and the throughputs of different TCP and UDP connections from [25]. However, these may not provide the desired QoS to different connections. To achieve this objective we will suggest a queue management and a rate control scheme. Ideally, the control schemes should be such that, the different data streams experience fixed delays within their required upper bounds, there is no delay jitter, no packet loss, each application obtains at least its required minimum throughput and there is minimal throughput fluctuation. Also, this should be done such that minimal changes are required in the current Internet and no per-flow processing is done in the core routers.

We try to achieve these objectives by rate control at the UDP and RED control at the router. The router will need the mean packet lengths and propagation delays for different connections. The mean packet lengths for different connections can be estimated at the router itself. Often (especially, for TCP connections) the packet size of a connection is fixed. Thus, it can be easily obtained at the router in the beginning itself. The new routers will have such capabilities. Also, this information along with the needed throughput (and other QoS requirements) can be provided by the users at the connection setup time. The propagation delays of a TCP connection can be estimated from the RTTs (Round Trip Times) received at the TCP source. This estimate will be required at the router. If we do not provide the information about the propagation delays of the TCP sources to the router, we could take them to be zero in our control scheme. The effect of it will be that the TCPs will get less throughput than specified and as a result the UDPs will get more than their minimum required throughput. However the other QoS requirements (on delay, delay jitter and packet loss) will still be satisfied.

We use the following notation. The maximum window size of the  $i$ th TCP connection is  $W_{\max}(i)$  and its total propagation delay is  $\Delta(i)$ . The mean throughput obtained/desired (in number of packets/sec) for the UDP and the  $i$ th TCP stream are  $\lambda_U$  and  $\lambda_T(i)$ .

### 3. The control scheme

In this section we develop the overall control scheme which will provide the objectives mentioned above. To motivate this scheme we first explain in section 3.1 the interaction of rate control schemes with the TCP window flow control. This motivates the overall control scheme described in section 3.2. In section 3.3 we extend the control scheme to a system with multiple UDPs and multiple routers. Section 3.4 incorporates short lived TCP connections. Section 3.5 discusses modifications to the control scheme to make it more realistic and also discusses other practical considerations. In section 4 we will show the effectiveness of this scheme via simulations.

#### 3.1. Preliminaries

To understand the interaction between the rate controlled schemes and the TCP window control, we start with the assumption that the buffer length at the queue is infinite, there is no rate control at the UDP stream and no RED control at the router. Such a system has been studied in [25]. Since there is no packet loss, the TCP window lengths will tend to  $W_{\max}(i)$  and stay there from then onward. It is shown in [25], that the UDP gets its required throughput  $\rho (= \lambda_U E[s])$  if it sends packets at rate  $\lambda_U$ . Also, all the TCP connections together get the throughput  $1 - \rho$  as long as the propagation delay of at least one of the connections is not too large (so that the queue never becomes empty). Also, the throughputs obtained by the different TCP connections depend on their parameters  $W_{\max}(i)$ ,  $E[s_T(i)]$ ,  $\Delta(i)$ . The mean delay of each stream is roughly same and is determined by  $\rho$  and the above TCP parameters. Similar conclusions hold even if there is RED control at the router [25].

To obtain the extra flexibility in being able to control the delay, delay jitter and the throughputs of different streams, we consider controlling the rate of the UDP stream also. Rate control instead of window control is a more natural choice for the UDP stream. Consider the objective of trying to keep the workload in the queue at a fixed level, say at  $V_d$  bits,  $0 < V_d < B - \delta$ , where  $B$  is the buffer length in bits and  $\delta$  is appropriately chosen. This ensures that the waiting times of all packets are fixed (specified by  $V_d$ ), there is little delay jitter, no packet loss and 100% channel efficiency (since  $V_d > 0$ ). By appropriately deciding the  $V_d$  we can control the waiting times to  $V_d/\mu$ , where  $\mu$  is the service rate of the queue in bits/sec. Of course to realize this objective, one needs an appropriate rate control scheme.

Suppose we have succeeded in obtaining a control scheme which keeps the workload at  $V_d$  (a suitable rate control algorithm will be described in section 3.2). Let  $\lambda$ ,  $\lambda_T(i)$  be the throughputs (in packets/sec) of the UDP and the  $i$ th TCP stream. Let  $E[q_T(i)]$  be the mean (stationary) number of TCP packets in the queue. Then  $W_{\max}(i) - E[q_T(i)]$  is the mean number of TCP/ack packets propagating at any time. By Little's law,

$$\lambda_T(i)\Delta(i) = W_{\max}(i) - E[q_T(i)], \quad \frac{\lambda_T(i)V_d}{\mu} = E[q_T(i)].$$

Hence

$$\lambda_T(i) = \frac{W_{\max}(i)}{\Delta(i) + V_d/\mu}, \quad E[q_T(i)] = \frac{W_{\max}(i)V_d}{\mu\Delta(i) + V_d}. \quad (1)$$

Also, since the queue is never empty,

$$\lambda = \frac{\mu - \sum_{i=1}^N \lambda_T(i)E[s_T(i)]}{E[s]}. \quad (2)$$

Thus we observe that once the  $V_d$  is specified, the throughputs received by the UDP and the various TCP streams also get fixed. However the difference now (as compared to the system described above without rate control) is that the TCP throughputs get fixed while the UDP now gets the left over throughput. The other difference is that now we have achieved the following QoS requirements: fixed, desired delays, no delay jitter and no packet loss in the router. And this has happened by only applying rate control at the UDP sources based on the delay information of their packets. This information is easily available to the UDP sources if they use the RTP, RTCP protocols (see section 3.5 for more details). If we also want to provide the different connections their required throughputs, we need to apply control at the bottleneck router. The router may require the information about the parameters of the TCP connections passing through it. We explain the overall control scheme in the next section.

### 3.2. The control scheme

We describe in this section the complete control scheme which provides the required throughput to the UDP and TCP streams, there is a fixed pre-specified delay for each stream, minimal delay jitter, 100% channel utilization, no packet loss and low throughput fluctuation after initial transients.

The rate control scheme for the UDP which provides the desired  $V_d$  will be explained later on. First we explain how we can provide the required throughputs in the scheme explained at the end of section 3.1. Observe that  $W_{\max}(i)$ ,  $\Delta(i)$ ,  $E[s_T(i)]$  are fixed and will not be allowed to be changed (if that is also allowed, we can enlarge the class of controls in an obvious way). But from (1) we observe that the throughput  $\lambda_T(i)$  can be changed if we change the window size of the TCP. Since  $W_{\max}(i)$  is not allowed to be changed, we can drop some packets of the TCP to change its window size. In that case, if  $E[W(i)]$  denotes the average window size of the  $i$ th TCP connection, we should replace  $W_{\max}(i)$  in (1) with  $E[W(i)]$ . Instead of actually dropping packets, if we could provide the explicit congestion notification (ECN) (see [20]) to the TCP streams, then we will be able to change the window size without dropping the packets.

There is one undesirable effect of varying the window size of the TCP connections. This makes the TCP sources bursty and as a result the queue length and the resulting controlled UDP rates oscillate. For real time video and voice this deteriorates the QoS even when the mean throughput is sufficient. Therefore, in using this control policy we should ensure that the probability of TCP packet dropping/markings is small  $\sim 0.01$ . Also, this oscillatory effect will be small (because of averaging effects) if there are a

large number of TCP connections passing through the router. This is the likely scenario in the Internet. We will verify these observations via simulations in section 4.

In the following we use the RED algorithm [12] for packet dropping/ECN. One could easily use other such algorithms but this algorithm is being widely studied and has also been deployed in many current routers. If the average queue length (in number of bits – we are using RED in byte-mode) in the queue is  $\hat{v}$  then RED drops incoming packets with probability  $p(\hat{v})$ , where

$$p(\hat{v}) = \begin{cases} 0, & \text{if } \hat{v} < T_{\min}, \\ \frac{(\hat{v} - T_{\min})p_{\max}}{T_{\max} - T_{\min}}, & \text{if } T_{\min} \leq \hat{v} \leq T_{\max}, \\ 1, & \text{if } T_{\max} < \hat{v}, \end{cases}$$

and the  $T_{\min}$ ,  $T_{\max}$  and  $p_{\max}$  are appropriately chosen constants. Also, the average queue length at the time of the  $n$ th arrival is

$$\hat{v}_{n+1} = (1 - \beta)\hat{v}_n + \beta v_n,$$

where  $\beta$  is a small constant (the usual recommended value is  $\leq 10^{-3}$ ). The  $p(\hat{v})$  and  $\hat{v}_n$  provided above are approximations to the actual values (see [12] for details).

The overall control algorithm is as follows. Decide  $V_d$  based on the max delay requirement of the UDP stream. Let  $\lambda_U$ ,  $\lambda_T(i)$  be the desired throughputs of the different streams in packets/sec. From (1) we obtain the  $E[q_T(i)]$  and the desired  $E[W(i)]$ . Calculate  $\lambda$  from (2). If  $\lambda \geq \lambda_U$ , we can obtain all the required throughputs; if not, the link can not provide the desired throughputs. One may then have to reduce the throughput requirements for some of the streams so as to obtain a feasible throughput set. Once that happens, from the desired  $E[W(i)]$ , we compute the appropriate probability of packet loss  $p_i$  using the relationship

$$E[W(i)] = \sqrt{\frac{8(1 - p_i)}{3p_i}} + 1 - 2.5.$$

This is a modification of the formula provided in [19] and matches better with the relationship obtained between  $E[W(i)]$  and  $p_i$  via simulations. Once the  $p_i$ 's are fixed, then along with  $V_d$ , we decide on the RED parameters to be used for each connection (or we could partition TCP connections into classes based on their similar throughput requirements and parameters  $\Delta(i)$ ,  $E[s_T(i)]$  and select the RED parameters for each class. The coarser the classification, the less control we have on the QoS guarantees). We do not obtain unique RED parameters. But we can use some guidelines for fixing the RED parameters (these are not critical for our schemes). For example, we take  $T_{\min} < V_d$ , fix  $p_{\max}$  (say 0.1) and then accordingly find  $T_{\max}$ . This along with specifying the rate control algorithm to be described below will complete the control algorithm. It is desirable to have  $p \leq 0.1$  to make time-outs rare in the TCP connections. Otherwise the TCP behaviour is not well understood. Also, as explained above, a low  $p$  is desired to keep the rate fluctuations of the UDP low.

Finally, we explain the rate control algorithm we have used. This is a variation of the LMS algorithm usually used in the adaptive signal processing [14]. We have also used several other algorithms in [28] but this provides the best results. The desired throughput of the UDP is  $\lambda_U$  while the desired workload is  $V_d$  (in bits). We update the rates of the UDP at time instants  $nT$ , where  $T$  is an appropriate constant. Let the workload at time  $nT$  be  $V_n$ . After, observing  $V_n$ , we decide the UDP rate to be  $\lambda_{n+1}$  (in bits/sec). The cost function to be minimized is taken as

$$J(\lambda) = E \left[ \frac{\alpha'_1}{2} (\lambda - \lambda_U E[s])^2 + \frac{\alpha'_2}{2} \left( \frac{V_{n+1} - V_d}{T} \right)^2 \right],$$

where  $\alpha'_1, \alpha'_2$  are some positive constants. The iterative algorithm used to minimize the cost function is steepest descent. Therefore,

$$\lambda_{n+1} = \lambda_n - \alpha \frac{\partial}{\partial \lambda} J(\lambda)|_{\lambda=\lambda_n},$$

where  $\alpha$  is a small positive constant. Since  $J(\lambda)$  is not available, we replace it with  $(\alpha'_1/2)(\lambda - \lambda_U E[s])^2 + (\alpha'_2/2)((V_n - V_d)/T)^2$ . Also since  $(\partial V/\partial \lambda)|_{V=V_n}$  is not known, we replace it with  $(V_n - V_{n-1})/(\lambda_n - \lambda_{n-1})$ . To maintain the random oscillations within limits we truncate this derivative approximation to  $\pm 1$ . Thus, finally, the algorithm becomes

$$\lambda_{n+1} = \lambda_n - \alpha_1 (\lambda_n - \lambda_U E[s]) - \alpha_2 (V_n - V_d) \gamma_n, \quad (3)$$

where  $\alpha_1 = \alpha \alpha'_1$ ,  $\alpha_2 = \alpha \alpha'_2 / T^2$  and  $\gamma_n$  is

$$\gamma_n = \begin{cases} \min \left( 1, \frac{V_n - V_{n-1}}{\lambda_n - \lambda_{n-1}} \right) & \text{if } \frac{V_n - V_{n-1}}{\lambda_n - \lambda_{n-1}} \geq 0, \\ \max \left( -1, \frac{V_n - V_{n-1}}{\lambda_n - \lambda_{n-1}} \right), & \text{otherwise.} \end{cases} \quad (4)$$

The relative values of  $\alpha'_1, \alpha'_2$  provide the weightage given to the rate and the workload errors. We have kept  $\alpha'_1$  larger in our simulations which provides smoother rates for the UDP. Also smaller  $\alpha_1, \alpha_2$  provide less oscillations but the rate of convergence to the stationary value slows down (we have proved convergence of (3) using standard stochastic approximation techniques; however in this paper we will limit ourselves to demonstrating the effectiveness of this algorithm via extensive simulations).

### 3.3. Multiple UDPs and multiple routers

In this section we explain the rate control scheme when there are multiple real time applications sharing the queue. We will also develop our scheme when there are multiple routers involved. That will show that our control scheme can in principle be implemented in a general network scenario although of course scalability issues will arise.

We consider the case when there are  $M$  UDP sources sharing the bottleneck router. For source  $i$  let  $s(i)$  be its generic packet length and let  $\lambda_U(i)$  be the desired throughput

in packets/sec. If  $D(i)$  is the the maximum allowed delay for connection  $i$ , then take  $V_d = \min_i D(i)/\mu$ . Now at time  $nT$  the rate for user  $i$  is given by

$$\lambda_{n+1}(i) = \lambda_n(i) - \beta(i) \left[ \alpha_1 \left( \sum_{j=1}^M (\lambda_n(j) - \lambda_U(j)E[s(j)]) \right) + \alpha_2(V_n - V_d)\gamma_n \right], \quad (5)$$

where  $\beta(i) = \lambda_U(i)E[s(i)]/(\sum_{j=1}^M \lambda_U(j)E[s(j)])$ . Observe that  $\lambda_n = \sum \lambda_n(i)$  satisfies (3) and the change in the  $i$ th UDP's rate is proportional to the rate it desires. Everything else stays as for a single UDP case.

Finally, we consider the case of a tandem of queues. We limit ourselves to two queues. Generalization to more than two queues is immediate. Various TCP and UDP connections may be passing through the first queue only, the second queue only or both the queues. Let at queue  $j$  the link speed be  $\mu_j$  and the desired workload  $V_{d,j}$ .  $V_{d,j}$  will be decided by the delay requirements of different connections passing through the two queues. There may not be a unique specification of  $V_{d,j}$  and different considerations can be used in deciding the  $V_{d,j}$ . For brevity, here we assume that we have decided upon the desired workloads  $V_{d,1}$  and  $V_{d,2}$ . If the probability of packet loss of TCP connection  $i$  at queue  $j$  is  $p_i(j)$  then its overall packet loss probability is

$$p_i = 1 - (1 - p_i(1))(1 - p_i(2)). \quad (6)$$

The mean window size  $E[W(i)]$  depends upon  $p_i$  as for the single queue case. The overall propagation delay of connection  $i$  is  $\Delta(i)$ . The throughput  $\lambda_T(i)$  of TCP connection  $i$ , if it passes through both the queues is obtained from (1) if we replace  $W_{\max}(i)$  by  $E[W(i)]$  and  $V_d/\mu$  by  $V_{d,1}/\mu_1 + V_{d,2}/\mu_2$ . If it passes through only one of the queues (say queue  $j$ ), then  $V_d/\mu$  is replaced by  $V_{d,j}/\mu_j$  (assuming these queues are the only bottleneck routers on its path – otherwise other routers on its path also need to be considered).

Therefore, for TCP connection  $i$ , for desired throughput  $\lambda_T(i)$ , we first obtain the  $E[W(i)]$  needed from (1). This provides the  $p_i$  needed. If connection  $i$  passes through both the queues, its packet drop probability  $p_i(j)$  in queue  $j$  is obtained such that (6) is satisfied.

Now let us discuss the rate control algorithm for UDP connection  $i$ . Let at time  $nT$ ,  $V_n(j)$ , be the workload in queue  $j$ . Based on  $V_n(j)$  only, using algorithm (3) (or (5)), we obtain  $\lambda_n(i, j)$ . If UDP  $i$  passes through only queue  $j$  then its rate is  $\lambda_n(i) = \lambda_n(i, j)$ . If it passes through both the queues then we specify  $\lambda_n(i) = (\lambda_n(i, 1) + \lambda_n(i, 2))/2$ . This is one possible compromise value for  $\lambda_n(i)$ . We show in section 4 via simulations that it works very well.

### 3.4. System with non-persistent TCPs

So far we have been considering a fixed number of persistent TCP connections. These approximately model the TCP connections which are sending long files. However, in a practical system, there are many short-lived TCP connections also, i.e., a source has a

small file to transmit. Now we include such TCP connections. For simplicity we will explain our system only for a single router.

Consider a bottleneck router on which an output link is being shared by  $N$  persistent TCP connections,  $M$  UDP connections, as in section 3.3, and in addition there is a Poisson stream of non-persistent TCP connections. The arrival rate of the non-persistent TCP connections is  $\lambda_N$ . These connections arrive, transmit a file of a random size (taken *iid*) and then these connections are closed. For this system also we try to keep the queue size at  $V_d$ . We denote a generic file size in packets by  $L_N$ .

If the file size is small then there is no point in trying to control its window size and hence we will let it transmit its file in the normal way. Thus, if the file is of size  $l$  packets where  $l$  is not large then the number of cycles (round trip times) required to transmit this file will be (assuming  $W_{\max}$  is large enough)

$$\bar{N}(l) = \min\{n: 1 + 2 + 2^2 + 2^3 + \dots + 2^n > l\} \quad (7)$$

because there is no packet loss in our scheme. If the propagation delay of this connection is  $\Delta$ , then since the queue length will be  $V_d$  bits, each cycle requires  $V_d/\mu + \Delta$  time. Thus, the download time of the file will be  $\bar{N}(l)(V_d/\mu + \Delta)$ . This is the minimum download time for the TCP connection if the queue length is kept at  $V_d$ . Thus the  $V_d$  should be small enough that this time is acceptable.

When the file of a non-persistent TCP connection is large, then if its window size is allowed to grow as in the last paragraph, it may eventually consume too much bandwidth. Thus, in that case, we may want to use RED control on this connection. Unlike for persistent connections, because of slow start phase and a finite file size, the number of cycles required to send a file of size  $l$  is now more complicated to compute. If the RED is dropping packets with probability  $p$  let  $\bar{N}(l, p)$  be the mean number of the cycles needed to download this file. Then, the mean download time is  $\bar{N}(l, p)(V_d/\mu + \Delta)$ . Thus for a desired mean download time we may be able to compute  $p$ .

From the distribution of file sizes of the non-persistent connections, using  $\bar{N}(l)$  and  $\bar{N}(l, p)$  we can compute the mean download time  $E[S_N]$  of the non-persistent connections. The number of the non-persistent connections  $N_N$  alive at a time is random. Under stationarity, using Little's law,

$$\lambda_N E[S_N] = E[N_N].$$

The mean bandwidth consumed by a non-persistent connection is  $E[L_N s_N]/E[S_N]$ , where  $s_N$  is a typical packet size of non-persistent connections. Thus the total mean BW (bandwidth) requirement of non-persistent connections is  $E[L_N s_N]E[N_N]/E[S_N] = \lambda_N E[L_N s_N]$ . This implies that the total mean BW consumed by non-persistent connections does not depend upon the  $V_d$ , their propagation delays, the RED parameters and the other persistent TCP and UDP connections involved. Thus, to a first order approximation, unlike for persistent connections, using RED on the non-persistent TCP connections will not affect the throughput and other requirements of the persistent TCP and UDP connections. On the other hand, it can increase the download time of a non-

persistent connection. Therefore, we should not employ RED control on non-persistent TCP connections.

If we do not employ RED control on non-persistent connections, then the question is how do they affect the QoS requirements of other connections. Since these connections take away  $\lambda_N E[L_N S_N]$  bandwidth the rest of the link bandwidth may not be sufficient to provide the QoS to the non-persistent and UDP connections. Of course with the left-over BW we can work as in section 3.2 and provide whatever QoS can be provided but now we have one more degree of freedom. We may deny admission to some of the non-persistent TCP connections, i.e., with probability  $p_a$  we may not admit such connections. Then the BW non-persistent connections require is  $(1 - p_a)\lambda_N E[L_N S_N]$ . Connection admission control can also be applied on the persistent TCPs and UDPs (by admission control here, one may only mean that the bottleneck router under consideration may not accept that connection and then the connection may follow another route).

We will see in section 4 that including non-persistent TCP connections increases the oscillations in the queue length (and hence delay jitter) significantly. Thus even though all other QoS parameters can be satisfied, delay jitter could be of concern. More recently [26] we have shown that if non-persistent TCP flows are isolated (say via Weighted Round Robin) then we can again obtain low delay jitter for the other connections.

### 3.5. Some practical considerations

The control algorithm in sections 3.2–3.4 implicitly makes some assumptions which have practical implications. In this section we try to relax these assumptions. We will also make some other comments related to practical networks.

(i) *Delayed feedback to UDPs.* Equation (5) assumes that the queue measurement  $V_n$  is immediately available to the different UDPs. It is more realistic to assume that the UDP  $i$  gets the information  $V_n$  with a delay of  $\delta(i)$ , i.e., at time  $nT + \delta(i)$ , where  $\delta(i)$  are some constants which can be different for different UDPs. We will show in the next section, that even with delayed information (upto some practically useful limits) to different UDPs, the algorithm provides good QoS to different users. Although we have not tried in simulations, if  $\delta(i)$  are not constant but involve some randomness, the algorithm will continue to provide good results.

Another practically realistic situation is that the packets of the UDPs reach the router after some delay. However for our purposes this delay could be absorbed in  $\delta(i)$  considered above.

(ii) *Computations at the source.* Computation of rates in (5) could be done at the router and the rates  $\lambda_n(i)$  sent to the different UDPs. However it requires some signalling support. To reduce/eliminate this, this computation can be done at each of the UDP sources. Then the UDP source  $i$  needs  $V_n$  and  $\lambda_n(j)$  for all  $j$  if we use (5). To reduce this dependence, we change (5) to

$$\lambda_{n+1}(i) = \lambda_n(i) - \bar{\alpha}_1(i)(\lambda_n(i) - \lambda_U(i)E[s(i)]) - \bar{\alpha}_2(i)(V_n - V_d)\bar{\gamma}_n(i), \quad (8)$$

where  $\bar{\alpha}_1(i)$  and  $\bar{\alpha}_2(i)$  are some suitable small constants and  $\bar{\gamma}_n(i)$  is defined as in (4) but with the denominator changed to  $(\lambda_n(i) - \lambda_{n-1}(i))$ .

With the modification proposed in the above paragraph, now UDP source  $i$  needs only  $V_n$  and  $V_d$ . Instead of  $V_n$ , the workload at the router at time  $nT$ , we could consider the end-to-end delay of packets of source  $i$  at time  $nT$ . An estimate of one way end-to-end delay of a UDP packet can be obtained by time stamping at the source and the destination. Various issues involved in the accuracy of such an estimate are discussed in [1]. From these it emerges that one way end-to-end delay can be obtained with an acceptable accuracy with a fixed offset (due to clock synchronization offset at the source and the destination). Thus if there is only one bottleneck router on the route of a UDP,  $V_n$  can be estimated with a fixed bias which needs to be included in  $V_d$  also. Then the effect of this bias gets cancelled in  $V_n - V_d$ . The end-to-end delay information of a packet (we only need for a packet at time  $nT$ ) can be included in the RTCP receiver reports in the application-defined RTCP packets [22]. We will see in section 4 that the modified equation (8) along with the delayed feedback information to the UDP sources still provides all the QoS guarantees obtained in section 3.3 from (5).

Next there is the question of  $V_d$ . We have mentioned in section 3.3 that  $V_d$  could be decided based on the end-to-end requirements of real time connections. Then,  $V_d$  may change from time to time and the users need to be informed about it. It will be desirable to fix  $V_d$  which will be acceptable most of the time. Observe that we are concerned about the bottleneck routers only and that real time applications can generally tolerate a few hundred msec ( $\sim 250$  msec) of end-to-end delay. Thus, making  $V_d$  such that the queueing delay at a bottleneck router is  $\sim 20$  msec can be acceptable. If the link speed  $\mu$  is 10 Mbps and the average packet size (of all the packets arriving) is 500 bytes, for a delay of 20 msec,  $V_d$  will be 50 packets. This requires modest buffer size but is large enough to ensure that the queue will not be empty.

Once a  $V_d$  is fixed, this can be known at the users apriori. A usual scenario is that the routers at the access networks are the bottleneck routers while in the core there is not much queueing. Then, our algorithm can be used at the access networks with the aim of keeping  $\sim 20$  msec delay.

*(iii) Route changes of a connection.* In IP-Internet the path followed by different packets of a connection can be different. This causes problems in providing QoS in any scheme. Thus, route changes should be minimized. Initially, the routing algorithms in Internet were using hop-count as the cost of a path. Then a route usually changes only if there is a link failure. This anyway will be required. But even when link costs are dynamically changed (based on say the congestion on a link), Internet provides the option of source routing by which one can fix a route for a particular connection.

*(iv) Signalling requirements.* With the above mentioned changes, a UDP connection can run its rate control algorithm without problem. For TCP connections, one needs to setup the RED parameters at a bottleneck router. These depend upon the probability of loss/markings  $p$  selected for that TCP. But  $p$  depends upon the propagation delay  $\Delta$ , the mean packet size  $E[s_T]$  and the desired throughput  $\lambda_T$ .  $\lambda_T$  and  $E[s_T]$  can be provided

by the TCP connection at the time of connection setup (actually packet size will in fact be fixed and not random). However  $\Delta$  needs to be informed to the router. If the TCP connection is sending a small file (short-lived) then as discussed in section 3.4, there is no need to use RED on its packets. If the file is long (persistent connection) then, after a few slow start cycles, the user will have an estimate of its round trip time (RTT) which can be sent to the bottleneck router to set the RED parameters for this TCP. Infact, the router might have a few fixed RED parameters to choose from (per-class processing) and may choose the best fit for the new TCP connection.

(v) *With Intserv and Diffserv architectures.* One can think of our algorithm, as a way to convert a bottleneck router (which will have variable, unpredictable delays, packet losses and throughputs for different connections) into a box with predictable, controllable fixed delay, with no packet loss and which provides a predetermined throughput to each connection. Thus it can be used to provide a reliable per-hop behaviour (PHB) in a Diffserv-capable network. The advantage of using our scheme as compared to others (based on weighted fair queueing, weighted round robin, etc.), is that it is much easier to implement (single queue, no buffer management, FCFS), requires no traffic shaping, marking (except via RED) and provides QoS guarantees.

We can also use our scheme with Intserv. The RSVP can be used for signalling while instead of resource reservation one can do RED control at the routers for the TCP and rate control at the sources for the UDP.

#### 4. Simulations

We have done extensive simulations for our scheme for one and two queues in tandem. The UDP flows were taken as CBR, Poisson, or MMPP with the rates controlled by our algorithms. We obtained very good match between the QoS requirements and the simulation results in all the cases. These are reported in [28]. Here we provide a representative sample for lack of space. We used ns-simulator, version 2.1b6 of UCB/LBNL. All simulations were run for 200 seconds, much longer than required to attain stationarity. We used  $\alpha_1 = 0.08$ ,  $\alpha_2 = 0.2$  and  $T = 100$  msec unless stated otherwise. Also  $\beta$  used in the RED algorithm is 0.002.

First we provide simulation results to show the following interesting facts mentioned in section 3:

1. For the infinite buffer system our algorithm converges very fast and the workload and the rate processes have virtually no oscillations.
2. Once RED is introduced, we can change the rates of TCPs and UDPs to the desired level but the workload and rate processes oscillate substantially.
3. As the number of TCP connections increase, by the averaging effect the oscillations reduce considerably.

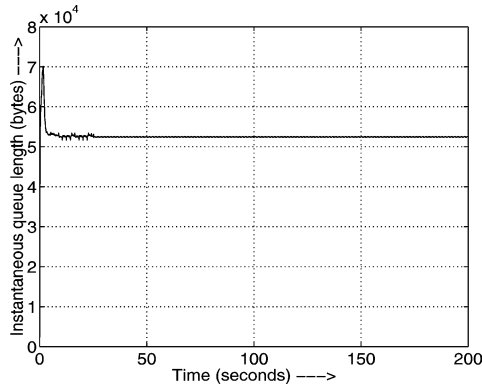


Figure 1. Queue length.

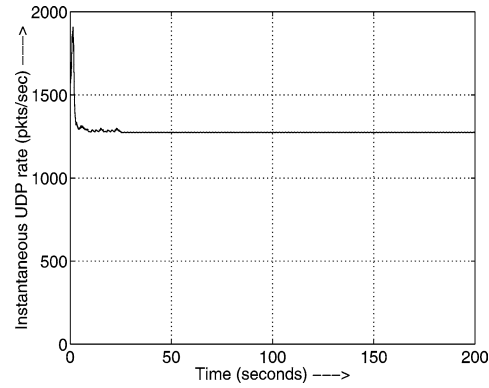


Figure 2. UDP rate.

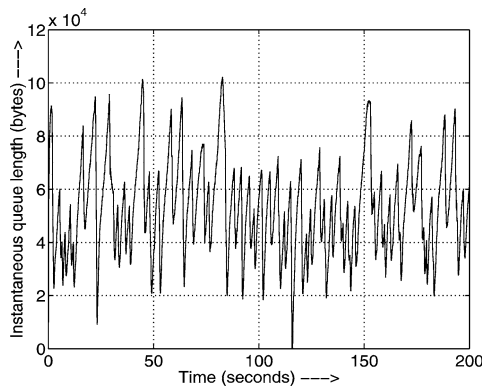


Figure 3. Queue length.

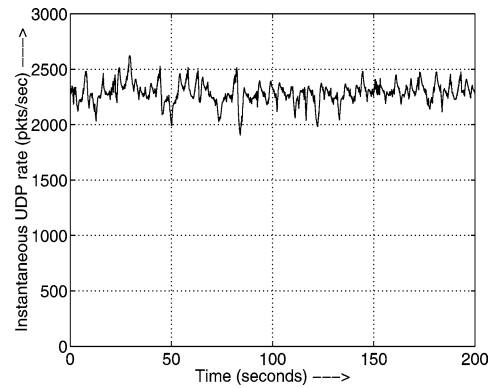


Figure 4. UDP rate.

To show these facts, we simulated a single queue with 3 Mbps link speed,  $V_d/\mu = 140$  msec, 1 TCP and 1 UDP connections, with fixed packet lengths of 750 bytes and 100 bytes, respectively. The TCP parameters are  $\Delta = 10$  msec and  $W_{\max} = 50$ . The UDP is sent as a CBR stream with the rate dictated by the algorithm. Figure 1 shows the plot of the instantaneous queue length and figure 2 of the UDP rate for the infinite buffer queue with no RED control. As mentioned, now we have no control over the UDP and TCP throughputs and using (1), we obtain  $\lambda_T = 333.33$  pkts/sec and  $\lambda_U = 1250$  pkts/sec. In fact for this case we take  $\alpha_1 = 0$  since the UDP rate is decided by  $V_d$  and the TCP parameters. Suppose our QoS requirement for  $\lambda_U$  is 2250 packets/sec. Then,  $\lambda_T = 200$  packets/sec. To obtain these throughputs while keeping  $V_d/\mu = 140$  msec, we need  $E[W] = 30$  and the corresponding packet loss probability is  $p = 0.0025$ . We employed RED with parameters  $T_{\min} = 30,000$  bytes,  $T_{\max} = 9,29,988$  bytes and  $p_{\max} = 0.1$ . Figures 3 and 4 provide the plots for the corresponding queue length and the UDP rate. One observes heavy oscillations in these plots. Figures 5 and 6 show the curves when there are 10 TCPs and each TCP losses packets with probability 0.0025. Now to retain the throughputs of each TCP to be 200 packets/sec, the different system parameters are

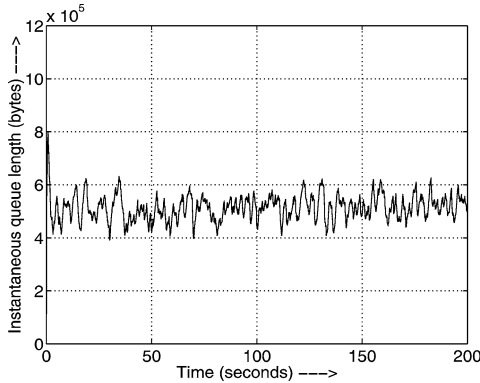


Figure 5. Queue length.

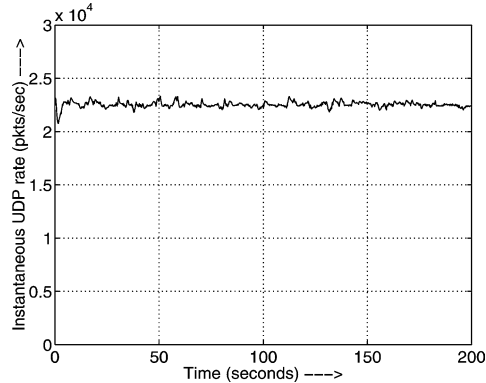


Figure 6. UDP rate.

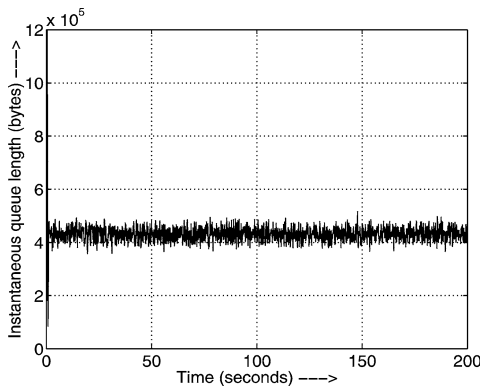


Figure 7. Queue length 1.

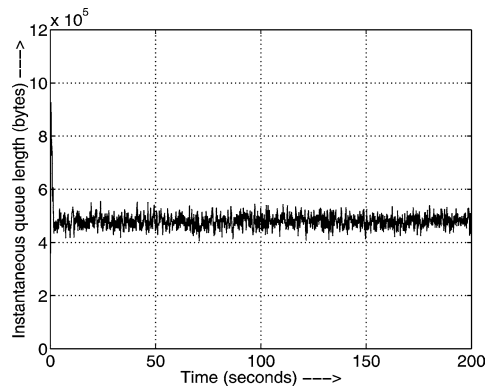


Figure 8. Queue length 2.

as follows. Link speed is 30 Mbps,  $\lambda_U$  is 22,500 pkts/sec and each TCP throughput is 200 pkts/sec. All the other parameters ( $\Delta(i)$ ,  $W_{\max}(i)$ ,  $V_d$ ) remain as before. The oscillations in this case are much less than in figures 3 and 4.

Next we consider the example of a tandem of two queues. The first queue has the BW of 50 Mbps and the second of 55 Mbps. There are 80 TCP connections and 7 UDP connections in the system. We classify the TCP connections into six classes. Class 1 consisting of 20 TCPs passes through both the queues. Classes 2, 3 and 4 consisting of 10 TCPs each, go through only the first queue. The classes 5 and 6 consisting of 15 TCPs each pass through only the second queue. The packet lengths of the six classes are 750, 1000, 1200, 1500, 750, 1000 bytes, respectively. Their propagation delays are respectively 30, 25, 20, 10, 10, 20 msec. Their throughput requirements are 50, 75, 100, 150, 100, 200 pkts/sec. The  $W_{\max}$  for all the connections are 50 packets each. The desired throughputs of the 7 UDPs are 2000, 4000, 3000, 4800, 2500, 3500, 4875 pkts/sec and their packet sizes are 80, 90, 100, 100, 100, 110, 120 bytes. The UDPs 1 and 2 pass through only queue 1 and are CBR streams. The UDPs 3, 4 pass through both the queues and are MMPPs. The UDPs 5, 6, 7 pass through only the second queue and are

Table 1  
Throughputs (pkts/sec) and UDP delays (sec), two queue case.

TCP classes	$\lambda_T$	$\lambda_U$		Delay
		UDP	Throughput	
1	53,52,50,50,46,50,49,47,50,51	UDP1	1997	0.069
		UDP2	3989	0.069
2	78,72,71,72,72,76,74,71,78,76	UDP3	3137	0.138
3	104,98,102,90,105,95,97,102,93,95	UDP4	5019	0.138
4	146,145,144,151,151,158,147,142,158,155	UDP5	2614	0.069
5	100,101,98,98,101,102,104,93,100,95	UDP6	3504	0.069
		UDP7	4880	0.069
6	186,204,198,198,198,207,204,186,197,193			

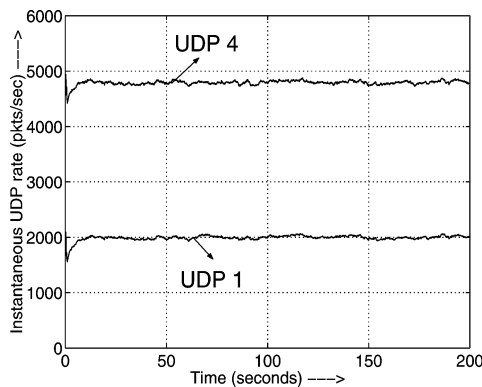


Figure 9. (Rate) UDPs 1, 4, two queue case.

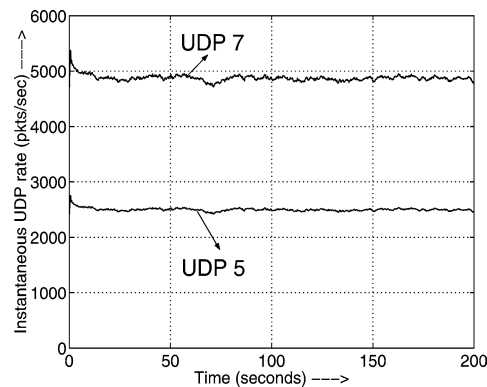


Figure 10. (Rate) UDPs 5, 7, two queue case.

respectively MMPP, CBR, CBR. The means of ON and OFF periods of the MMPP UDP streams are 10 msec each. The desired waiting times in the two queues are 70 msec each.

From (1),  $E[W(i)]$  for the 6 TCP classes are 8.5, 7.125, 9, 12, 8, 18 and the corresponding  $p_i$ 's are 0.0217, 0.0283, 0.0199, 0.0126, 0.0238, 0.0063. The RED parameters are:  $T_{\min} = 2,00,000$  bytes,  $p_{\max} = 0.1$  for each class and  $T_{\max} = 1294470, 1039222, 1393467, 2084913, 1381720, 4664288$  bytes for the 6 classes. RED is not employed on the UDP streams. For class 1 RED is employed at the first queue only.

The throughputs obtained from simulations for all the TCPs are provided in table 1. The throughputs and the mean delays for the UDPs are also in table 1. We see a good match to the QoS desired in each case. The mean delay for each TCP also, although not recorded here, is very close to the theory. The instantaneous queue lengths of the two queues are provided in figures 7 and 8. The rates of UDP 1 and 4 are in figure 9 and of UDP 5 and 7 are in figure 10. We see very little oscillations in the queue lengths and the UDP rates. The same holds for the other UDP rates which are not provided here.

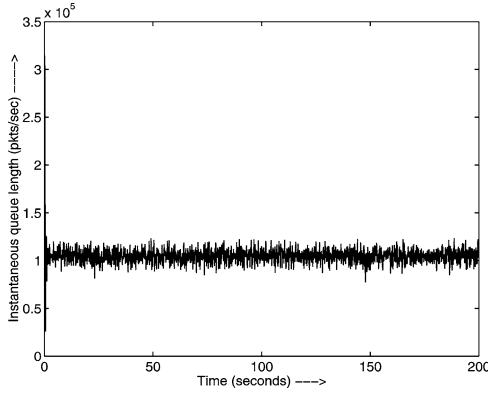


Figure 11. UDP rates for decentralized control.

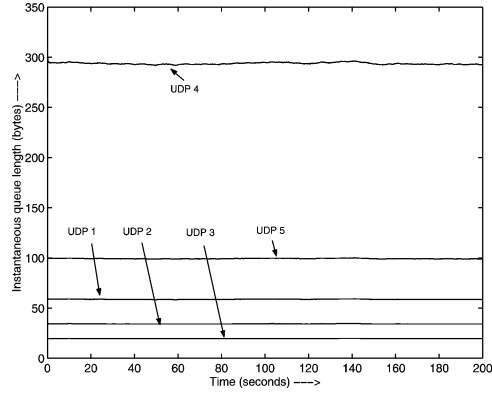


Figure 12. Queue size for decentralized control.

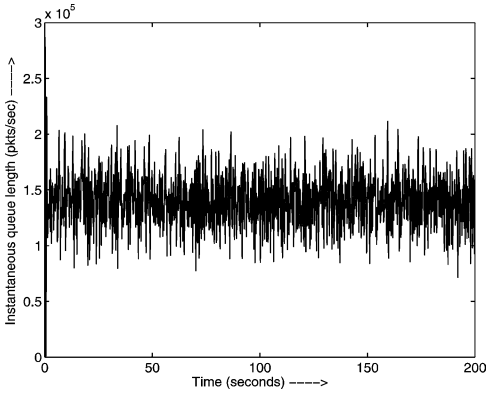


Figure 13. One queue, UDP rates with non-persistent TCPs.

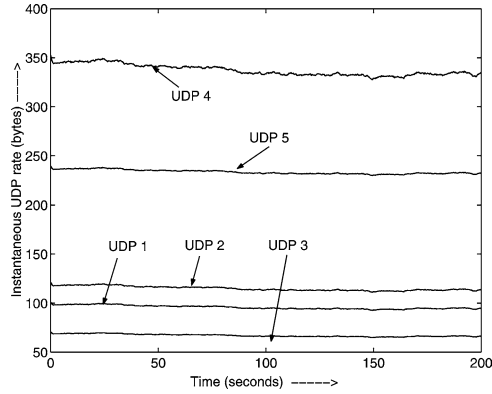


Figure 14. One queue, queue size with non-persistent TCPs.

Next, we provide an example to show that if the centralized algorithm developed in section 3.3 is replaced by the algorithm with feedback delays to the UDPs and computations at sources, as explained in section 3.5 we can still obtain very good QoS. We limit ourselves to the single router case. There are five UDP connections and 25 TCP connections. The UDP connections are generating traffic as CBR streams. Their packet sizes are respectively 100, 90, 80, 1000, 1500 bytes. The desired throughputs are 60, 35, 20, 300 and 85.5 packets/sec. The first three streams might have been generated by real time voice sources and the last two by video sources. The TCP connections are divided into four classes. The first class has 10 TCPs and rest have five TCPs each. The propagation delays of the four classes are respectively 20, 15, 30 and 25 msec. The packet sizes are 500, 750, 1000, 600 bytes. The  $W_{max}$  of each connection is 50. The desired throughputs of TCPs in the four classes are 100, 150, 175, 200 packets/sec. The link speed is 24 Mbps and  $V_d = 35$  msec. To obtain the desired throughputs of the TCPs, we choose the RED parameters for the four classes as follows:  $p = 0.061, 0.043, 0.021, 0.020$ ;  $T_{min} = 30.79$

Table 2  
One queue with decentralized control.

TCP classes	$\lambda_T$	$\lambda_U$		Delay
1	104, 99, 103, 100, 99, 98, 97, 105, 107, 102	UDP1	61	0.034
2	158, 151, 150, 158, 164	UDP2	35	0.034
3	172, 181, 188, 174, 182	UDP3	20	0.034
4	211, 207, 198, 208, 201	UDP4	306	0.034
		UDP5	101	0.034

Table 3  
One queue with non-persistent TCPs.

Persistent TCP classes	$\lambda_T$	$\lambda_U$		Delay	Non-persistent TCPs	Mean download time
1	132, 133, 132, 139, 138, 141, 135, 135, 136, 129, 134	UDP1	96	0.0338	file size 10	0.145
2	110, 109, 109, 110, 108, 103, 108, 103, 109, 105	UDP2	116	0.0337	file size 20	0.229
3	82, 78, 76, 78, 80, 80, 86, 79, 77, 81	UDP3	67	0.034	file size 100	0.481
		UDP4	338	0.0338	file size 150	0.320
		UDP5	233	0.0342		

for each class,  $T_{\max} = 101.82, 130.59, 238.06, 243.17$ . We simulated these system with the centralized algorithm in section 3.3, but with delayed feedback to the UDPs and computations at each source (as in (8)). The results for all three cases are almost identical but we report only the last case (with UDP delayed feedback and (8)). The five UDPs have the feedback delays of 0, 5, 10, 15, 25 msec, respectively. The  $\bar{\alpha}_1(i) = 1.28 \times 10^{-4}, 6.72 \times 10^{-5}, 3.4 \times 10^{-5}, 6.4 \times 10^{-3}$  and  $2.7 \times 10^{-4}$ ,  $\bar{\alpha}_2(i) = 3.2 \times 10^{-5}, 1.6 \times 10^{-5}, 8.4 \times 10^{-6}, 1.6 \times 10^{-3}$  and  $6.84 \times 10^{-4}$  for  $i = 1, 2, 3, 4, 5$ . The throughputs for the TCPs and the UDPs and mean delays of the UDPs are provided in table 2. The queue-length process is shown in figure 12 and the UDP rates are in figure 11. We observe that even in this scenario, we obtain very good QoS.

Finally, we provide an example which includes non-persistent TCP connections also. Now the linkspeed is 32 Mbps. The packet sizes of UDPs are 50, 90, 80, 1000 and 1500 bytes and their desired throughputs are 100, 120, 70, 350 and 238.4 packets/sec, respectively. There are three persistent TCP classes, each with ten TCPs. The packet sizes of TCP classes are 500, 600 and 1000 bytes and their throughput requirements are 125, 100 and 75 packets/sec with  $\Delta$  of 20, 25 and 30 msec, respectively. The packet drop probabilities  $p$  in the RED algorithm are 0.0298, 0.036 and 0.0475 with  $T_{\min}$  81.02 packets while the  $T_{\max}$  for the three classes are 316.6, 276 and 228.84 packets. The non-persistent connections arrive as a Poisson process with rate 60 connections/sec. Their propagation delays are kept 5msec while the packet sizes are 40, 800, 500 and 1500 bytes. The file sizes take values 10, 20, 50 and 30 packets with equal probabilities.

We keep  $V_d$ ,  $\bar{\alpha}_1$  and  $\bar{\alpha}_2$  as in the above example.  $W_{\max}$  for the non-persistent connections is 5. From our theoretical calculations the download times are 0.14, 0.22, 0.49 and 0.32 secs for the four file sizes mentioned above. We are using the decentralized algorithm (8) and the feedback delays for the UDP streams are as in the above example. The values from the simulations are provided in table 3. We see a good match between theory and simulations. The persistent TCP throughputs and the UDP throughputs and mean sojourn times are also provided in table 3. The queue length process and the UDP rates are shown in figures 13 and 14. We see that the UDP rates are almost constant close to the desired values, but there are more oscillations in the queue length process than in figure 11. This is of course expected.

## References

- [1] G. Alens and S. Kalidindisu, A one-way-delay metric for IPPM, Internet RFC 2679 (1999).
- [2] E. Altman, K. Avrachenkov and C. Barakat, A stochastic model of TCP/IP with stationary losses, in: *TCP Workshop*, INRIA, France, September 2000.
- [3] F. Baccelli and D. Hong, TCP is max-plus linear and what it tells on its throughput, INRIA Technical Report (August 2000).
- [4] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang and W. Weiss, An architecture for differentiated services, IETF RFC 2475 (December 1998).
- [5] T. Bonald, Comparison of TCP Reno and TCP Vegas; efficiency and fairness, *Performance Evaluation* 36/37 (1999) 307–322.
- [6] R. Braden, D. Black and S. Shanker, Integrated services in the Internet architecture: An overview, IETF RFC 1633 (June 1994).
- [7] N. Caldwell, S. Savage and T. Anderson, Modelling TCP latency, in: *Proc. of IEEE INFOCOM 2000*.
- [8] H.J. Chao and X. Guo, *Quality of Service Control in High-Speed Networks* (Wiley, New York, 2002).
- [9] G. De Veciana, T.J. Lee and T. Konstantopoulos, Stability and performance analysis of networks supporting elastic services, *IEEE/ACM Trans. Networking* 9 (2001) 2–14.
- [10] V. Firoiu and M. Borden, A study of active queue management for congestion control, in: *Proc. of IEEE Conf. INFOCOM 2000*.
- [11] S. Floyd, M. Handley, J. Padhye and J. Widmer, Equation based congestion control for unicast applications: The extended version, Preprint (March 2000).
- [12] S. Floyd and V. Jacobson, Random early detection gateways for congestion avoidance, *IEEE/ACM Trans. Networking* 1 (1993) 397–413.
- [13] A. Gupta and V. Sharma, A unified approach for analyzing persistent, non-persistent and ON–OFF TCP sessions in the Internet, submitted.
- [14] S. Haykin, *Adaptive Filter Theory*, 3rd ed. (Prentice-Hall, Englewood, Cliffs, NJ, 1996).
- [15] O. Hersent, D. Gurle and J.-P. Petit, *IP Telephony: Packet-Based Multimedia Communication Systems* (Addison-Wesley, Reading, MA, 2000).
- [16] S. Jacobs and Eleftheriadis, Real time dynamic rate shaping and control for Internet video applications, in: *Workshop on Multimedia Signal Processing*, Princeton, NJ, 23–25 June 1997.
- [17] R.J. La, V. Anantharam and J. Walrand, Analysis and comparison of TCP Reno and Vegas, in: *Proc. of IEEE Conf. INFOCOM*, 1999.
- [18] V. Misra, W.B. Gong and D. Towsley, Fluid based analysis of a network of AQM routers supporting TCP flows with applications, in: *Proc. of ACM SIGCOMM*, 2000, pp. 1714–1726.
- [19] J. Padhye, V. Firoiu, D. Towsley and J. Kurose, Modelling TCP throughput: A simple model and its empirical validation, in: *Proc. of ATM SIGCOMM'98*.

- [20] K. Ramakrishnan and S. Floyd, A proposal to add explicit congestion notification (ECN) to IP, IETF RFC 2481 (January 1999).
- [21] R. Rejaie, M. Handley and D. Estrin, Layered quality adaption for Internet video streaming, *IEEE J. Selected Areas Commun.* 18 (2000) 2530–2543.
- [22] H. Schulzrinne, S. Castner, R. Frederick and V. Jacobson, RTP: A transport protocol for real time applications, Internet RFC 1889 (1996).
- [23] V. Sharma, Modelling, analysis and control of TCP and real time connections for quality of service, in: *Internat. Conf. on Stochastic Modelling and IV Internat. Workshop on Retrail Queues*, Cochin, India, December 2002.
- [24] V. Sharma and J. Kuri, Stability and performance analysis of rate-based feedback flow controlled ATM networks, *Queueing Systems* 29 (1998) 129–159.
- [25] V. Sharma and P. Purkayastha, Performance analysis of TCP connections with RED control and exogenous traffic, in: *Proc. of IEEE Conf. GLOBECOM*, 2001.
- [26] V. Sharma and Suma M.B., Providing QoS to real time and TCP connections via rate control of UDP, in: *Proc. National Communications Conference (NCC)*, Bangalore, India, 2004.
- [27] K. Thompson, G.J. Miller and R. Wilder, Wide-area Internet traffic patterns and characteristics, *IEEE Networks* (November/December 1997) 10–23.
- [28] V. Venugopal Reddy, Providing QoS to TCP and real time connections in the Internet, M.E. thesis, ECE Department, IISc, Bangalore (January 2002).
- [29] V. Venugopal Reddy, V. Sharma and M.B. Suma, Providing QoS to TCP and real time connections in the Internet, Technical Report TR-PME-2002-01, DRDO-IISc Program on Math. Engrg. (January 2002).
- [30] D. Wu, Y.T. Hou and Y.-Q. Zhang, Scalable video coding and transport over broad-band wireless networks, *Proc. IEEE* 89 (2001) 6–20.