A SIMPLE APPROACH TO PROVIDE QOS AND FAIRNESS IN INTERNET

Vihang Kamble and Vinod Sharma

Department of Electrical Communications
Indian Institute of Science
Bangalore, India

ABSTRACT

We propose an approach to provide QoS in Internet for real and non-real time applications. This is a modification of an approach suggested in [13], [14] and simplifies the overall procedure. It also removes the signaling requirements needed in [13], [14]. Using RED control and priority and WRR schedulers at core networks we obtain minimal delay and delay jitter for real time applications and specified minimum throughput for TCP connections. We also show that using admission control our scheme can provide QoS in dynamic traffic conditions and also fair and efficient use of resources. Its implementation in DiffServ architecture is specifically described.

1. INTRODUCTION

In the past few years the Internet has grown at a rapid pace. Originally the Internet was designed to transmit non-real time data traffic. Such traffic requires error free transmission of all packets but it is quite tolerant to random, variable delays. However presently Internet is expected to support several real time applications also. These applications have different Quality of Service (QoS) requirements: they can tolerate some packet loss but have stringent end-to-end delay requirements.

The QoS demands of different applications are generally expressed in terms of lower bounds on throughput and upper bounds on delay, delay jitter and packet loss [6]. It is desirable that QoS to different applications should be provided with minimal changes in the current Internet. Efforts have been made to suggest new architectures like IntServ ([3]) and DiffServ ([2]) to support the QoS for new applications. IntServ can provide granularity in QoS but suffers from the scalability problem. On the other hand DiffServ is scalable but there is no QoS guarantee for individual flows. Therefore, there is a need to suppliment the DiffServ architecture with schemes which can be implemented in the core routers that can help ensure QoS to individual flows without requiring per-flow processing. We suggest one solution to this problem.

Our approach can work in the current Internet also (without DiffServ support). In particular, using only RED control [5] and commonly used priority and weighted round robin (WRR) scheduling algorithms in the core routers, we ensure very little delay and delay jitter for real time applications and a certain minimum throughput for TCP traffic. Furthermore, this requires minimal extra signaling support and utilizes the resources efficiently.

Another important question is the fairness of allocation of network resources to different users. One common definition of fairness is Max-Min criterion ([1]). In [7] Kelly et al. show that the linear increase and multiplicative decrease algorithm converges to a proportionally fair point. In our approach we will use admission control to ensure that there are minimum network resources. We will show that our approach is (approximately) proportionally fair in utilizing extra network resources.

Our approach to QoS is a modification of the one suggested in [13], [14]. The scheme suggested in [13], [14] works quite well but has two problems. It needs extra signaling support to provide information about queue length at the core routers to UDP sources at regular interval. Also the end-to-end delay jitter may become intolerable for CBR traffic in case it passes through several congested routers. In this paper we modify the scheme to remove both of these shortcomings without sacrificing the performance and several good features of the approach. It also simplifies the overall procedure. Furthermore, we study its performance in dynamic traffic conditions and in Diffserv architecture. We also provide an admission control procedure. These issues had not been addressed in [13], [14].

Now we mention some relevant literature. TCP with RED control at the routers has been studied in Floyd and Jacobson [5] Misra et al.[10] and Sharma and Punyaslok [12]. In [8] Kunniyur and Srikant have suggested a Adaptive Virtual Queue (AVQ) to provide QoS. [16] also discusses the fairness of AIMD with heterogeneous round trip times. Other related papers providing QoS are [4] and [9].

The difference between our approach and other approaches is that only in our approach (to the best of our knowledge) we try to provide end-to-end QoS to individual applications without per-flow processing at the core routers.

The paper is organized as follows. Section 2 describes the network model. Section 3 describes the QoS approach suggested in [13], [14] and then the modifications made in this paper. Section 4 shows that this approach works in dynamic network conditions and is approximately "proportionally" fair. Section 5 describes an admission procedure which can be used in our setup to ensure that QoS to different applications can be guaranteed. Section 6 demonstrates the implementation of the procedure in DiffServ domain. Section 7 verifies via NS-simulations the various claims made in the paper.

2. MODEL

We start with the following simple model of one bottleneck router (multi-router case is described in [15], [13], [14] and demonstrated via simulations in Section 7). There are several UDP and TCP streams passing through the router with different QoS requirements. The UDP streams may be sending voice or multimedia traffic. The voice streams are CBR (Constant Bit Rate) streams. The multimedia streams may be generating VBR (Variable Bit Rate)

0-7803-8674-4/04/$20.00 ©2004 IEEE

traffic. Both of these applications can tolerate a little packet loss but have stringent upper bounds on end-to-end delay and delay jitter. The QoS of different TCP connections can be generally satisfied if they are provided certain minimum mean throughput (which can differ for different TCP connections).

TCP connections can be grouped into three categories: Persistent (TCP-P), Persistent ON-OFF (TCP ON-OFF) and Non-Persistent (TCP-NP) (see [13] for details). If a TCP connection is sending a long file (say using FTP) which will last for several minutes, this can be classified as a persistent connection. If a TCP connection is being used for web browsing using HTTP1.1, then it will possibly transfer several different files (web pages) with idle (think) times in-between. Also, a connection may last for several minutes. This will be a TCP ON-OFF connection. For TCP ON-OFF the download time of a file is the QoS parameter. To upper bound the download time, we will need to lower bound the average throughput of this connection. If a TCP connection lasts for a few seconds, it will be called TCP-NP. Such connections may not require any explicit QoS. It was found in [13], [14] that TCP connections in different categories should be handled differently.

3. OUR APPROACH TO PROVIDE QOS

In [13] we studied the model provided in section 2 and proposed an approach to QoS. In this paper we modify the proposed approach. The modification simplifies the overall approach and reduces the extra signaling requirements. Furthermore, in later sections we discuss the admission control and fairness issues in this setup and describe its implementation in DiffServ. These issues were not discussed in [13], [14].

At the core of the approach in [13] to provide QoS was to control the queue length to an appropriate level at a core router by rate control of the UDP sources. Whenever the queue length increases, the rates at various UDP sources are decreased and whenever the queue length decreases the rates are increased. To be able to do this requires signaling support to provide queue length information to the sources. In the present paper we will achieve the same objective without rate control at the UDP users. This reduces the protocol complexity and also eliminates the need of extra signaling.

Once the queue length at a router is made approximately constant, at a desired level we can achieve many objectives at the same time. Firstly by keeping the queue length a bit away from being zero and also much lower than the buffer size, we can ensure that the queue is never empty (thus full link utilization), there is no (or very little) packet loss, the delay is at a desired tolerable level (can be decided based on QoS requirements of real-time traffic) and there is very little delay jitter. Furthermore, we showed in [13], [14] that the different TCP-P and TCP-ON-OFF connections, in this scenario can be provided the mean throughput they desire, which can be different for different TCP-connections. This is done simply by using a RED algorithm to make the average window size of *ith* TCP connection $EW(i) = \lambda_T(i)(\triangle(i) + V_d/C)$ where $\lambda_T(i)$ packets/sec is the desired throughput, V_d is the queue length in bits fixed at the router, C is the total BW available for the TCPs and $\triangle(i)$ is the delay in the rest of the network (assuming, say only one router on the route is a bottleneck router - for more general scenario see [13], [14]). Although this is an approximation, extensive simulations in [14] show that it is a good approximation. In general the throughput of a TCP depends not only on its own parameters but also on that of other traffic at that node, but this decoupling of throughput happens because of the queue length being made constant.

The throughput for TCP described above is for TCP-P and TCP-ON-OFF. For TCP-NP the duration of the connection will be small enough such that the RED control will not have much impact. Also TCP-NP traffic may not really need any QoS. Furthermore the number of TCP-NP connections will keep varying, causing oscillations in the queue length. Because of these reasons, in our approach, at the router the CBR, TCP-ON-OFF and TCP-P connections share one queue (Q_1) with the TCP connections RED controlled while TCP-NP connections are in another queue (Q_2), without any RED control. The two queues share the link BW via WRR (Weighted Round Robin).

The traffic intensity of real-time VBR traffic can change at random times by random amount and it also requires very low delay and delay jitter. If this traffic is also sent through the first queue it would cause very large oscillations in queue length (and hence large delay jitter). Therefore, it is sent through the second queue with higher priority over the TCP-NP traffic. The bandwidth (BW) assigned to the second queue is at least equal to the sum of the peak rate requirements of all VBR connections passing through it. It is shown in [13] that giving higher priority to VBR connections does not affect the throughput of the TCP-NP connections but decreases the delay and delay jitter of VBR traffic drastically.

The approach is shown to provide the QoS to different real and non-real time applications quite well. However, it has two shortcomings. First as explained above, the UDP connections require extra signaling support to get queue length information from different congested routers. Secondly although the delay jitter at each router is quite low, if a CBR connection has to pass through several congested routers, the end-to-end delay jitter may become intolerable.

Due to these problems we suggest a few modifications. Now we give the UDP-CBR traffic priority over the TCP traffic in Q_1. Also, we do not employ the rate control scheme on the UDP sources. This approach gets rid of the signaling requirement and also makes the delay and delay jitter seen by the UDP traffic at the router almost zero if the total CBR BW requirement is less than (say half of) the BW given to Q_1. Furthermore, it comes at no cost to the TCP traffic. This is because, as observed before, giving priority to UDP does not decrease the mean throughput of TCP traffic (and in fact also its mean delay in the queue). For TCP, only mean throughput matters.

TCP traffic in Q1 is RED controlled as before. In the modification suggested, the UDP-CBR has been given priority over the TCP traffic. It takes the bandwidth required by it, and the TCP connections get the remaining bandwidth (this is reversal of what happened in [13], [14]). Now the question is how to make the different TCP connections share this BW according to their requirements. Also, the average queue length will no longer be the desired V_d.

In the usual setup of Internet, if the TCPs and UDPs are sharing a bottleneck link controlled by the RED algorithm then we get a fixed average queue length. This average queue length is decided by a fixed point equation and it depends upon the parameters of different UDP and TCP connections sharing the queue. If the average queue length is not V_d, our previous scheme will not give the different TCPs their required throughputs. In the following we provide a way such that the average queue length will be V_d and the different TCPs get their desired throughputs.

Suppose the average queue length in Q_1 has somehow got fixed at V_d. Now, the total (average) RTT of TCP connection i will be $V_d/C+\Delta(i)$, where $\Delta(i)$ is its delay in the rest of the network and C is the total BW available to Q_1. Thus, to obtain its desired throughput $\lambda_T(i)$ packets/sec, we will need its average window size $E[W(i)] = (V_d/C+\Delta(i))\lambda_T(i)$. The needed $E[W(i)]$ will be obtained by using RED algorithm on it which will drop its packets with probability p_i, where p_i is obtained via (3) below. This way one can specify the RED parameters for all the TCP-P and TCP-ON-OFF parameters such that at V_d each connection i will experience a packet loss p_i. If we operate the RED algorithms for all the connections, according to these parameters, under steady state (fixed point) the average queue length will be V_d and the different TCP connections will get their required throughputs. We will verify these claims via simulations.

We will show in section 6 that the TCP connections can be divided into classes, each class having one RED algorithm. Thus we will not need per-flow processing at the core routers.

In the following sections we study this system further. In Section 5 we will provide an admission control algorithm which will ensure that a connection is not admitted on a router unless there is sufficient BW on the required link. In Section 4 we will show that if a link has more BW than the minimum required, then the extra BW will be shared "fairly" among the different TCPs. Next we will consider admission control. This is required to ensure QoS. Finally, in section 6 we will discuss implementation of this scheme in DiffServ.

4. QOS IN DYNAMIC TRAFFIC CONDITIONS

In the Internet the number of TCP and UDP connections will vary with time. Also, the rate of arrival of TCP-NP will change with time. The QoS requirements of new arrivals will be possibly different from the existing connections. In this changing scenario we need to study how the scheme proposed in the previous section will behave.

If an output link on a core router in the Internet does not have much traffic at a time, it is possible that both the queues at the router will be small and may keep becoming empty. Then at that time average queue lengths measured by the different RED algorithms (especially when $T_{min}(i)$, the RED parameter as specified in (4) below, are kept same) may be less than $T_{min}(i)$ and hence they may not drop any packets. Then the router will behave in the normal way and all the connections will get their required throughputs and delays. Only when a router starts seeing congestion, its average queue lengths will increase and then our QoS mechanisms will start having impact, Furthermore, our admission control (to be discussed in the next section) will ensure that the throughputs of different connections or delays and delay jitter of real time connections will satisfy their QoS requirements. Since the real time applications have priority they will get the rates (throughputs) they need. In the following we discuss the scenario when more link BW is available than needed to satisfy the minimum QoS requirements. In that case we show that the excess BW between different TCP connections is shared in a "proportionally" fair way. This is shown by considering two cases separately.

Case 1:Let at sometime a TCP-P or a TCP-ON-OFF connection closes. Now extra bandwidth is available for the remaining TCP connections in Q_1 of the router. Therefore the average queue length decreases (at least initially) and hence the RED algorithm will drop the TCP packets with reduced probability. This increases their mean window sizes, causing their throughputs to increase. Hence their QoS requirements will still be met. Below we will study the fairness issue.

Case 2: Next consider the situation when a UDP-CBR connection closes. Now extra bandwidth becomes available in Q_1, which will be given to TCP connections. A similar situation will prevail when Q_1 gets some extra bandwidth from Q_2 (either because a VBR connection closes or TCP-NP traffic intensity decreases).

In Case 1 the total BW, C available to TCP connections remains same, only the number of TCP connections changes. In Case 2 the number of TCP (P and ON-OFF) connections may remain same but C changes. We analyze the two cases separately.

4.1. Case 1

Let $E[V_T]$ be the queuing delay in the router. We can write $E[V_T] = V_d/C$ if the average queue length at that time is V_d. From Little's law, for ith TCP connection,

$$\lambda_T(i) = E[W(i)]/(\Delta(i)+E[V_T]). \tag{1}$$

Differentiating w.r.t $E[V_T]$

$$\frac{\partial\lambda_T(i)}{\partial E[V_T]} = \frac{\partial E[W(i)]}{\partial E[V_T]}\frac{1}{(\Delta(i)+E[V_T])} - \frac{E[W(i)]}{(\Delta(i)+E[V_T])^2}. \tag{2}$$

From ([11]) (the last term in the following equality is our correction term)

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

where p_i is the probability of packet loss of the i^{th} connection by the RED algorithm. If p_i is small it is approximately $\sqrt{8/3p_i} - 2.5$.

The dropping probability for i^{th} flow by the RED algorithm is

$$p_i = p_{max}(i)\frac{(CE[V_T]-T_{min}(i))}{(T_{max}(i)-T_{min}(i))} \tag{4}$$

where C is the total link speed available to the TCP connections in Q_1 and $T_{min}(i), T_{max}(i), p_{max}(i)$ are the RED parameters specified for connection i. From equations (3) and (4),

$$\frac{\partial E[W(i)]}{\partial E[V_T]} = -\sqrt{\frac{2}{3p_i}}\frac{1}{(E[V_T]-\frac{T_{min}(i)}{C})}. \tag{5}$$

Let $\bar{T}_{min}(i) = T_{min}(i)/C$. From (1) and (2)

$$\frac{\partial\lambda_T(i)}{\partial E[V_T]} = -\frac{\lambda_T(i)}{2(E[V_T]-\bar{T}_{min}(i))} - \frac{1.25}{(\Delta(i)+E[V_T])(E[V_T]-\bar{T}_{min}(i))} - \frac{\lambda_T(i)}{(\Delta(i)+E[V_T])}. \tag{6}$$

For simplicity, we take $T_{min}(i) = T_{min}$ for all TCP long lived connections i passing through the router (the RED algorithm can be configured to do this). Then the solution of the above differential equation is is

$$\sqrt{(E[V_T]-\bar{T}_{min})}[\lambda_T(i)(\Delta(i)+E[V_T])+2.5] = K(i) \tag{7}$$

where $K(i)$ is a constant which can be determined from the initial setting of E$[V_T]$. This result shows how the throughputs of existing TCP connections change when $E[V_T]$ changes and can be used to study the throughput in the above mentioned scenarios. From equation 6, if $\tilde{T}_{min}$ and $E[V_T]$ are comparable (it is desirable since it ensures that $p_i s$ are small) the first two terms dominate, giving

$$\frac{\partial \lambda_T(i)}{\partial E[V_T]} \approx \frac{-1}{(\triangle(i)+E[V_T])(E[V_T]-\tilde{T}_{min})}\left(\frac{E[W(i)]}{2}+1.25\right).$$

If $E[W(i)]$ is sufficiently greater than 2.5 then using equation 1 we can say that the change in the throughput is approximately proportional to the $\lambda_T(i)$ (we call this proportional fairness). We support this statement by simulation results in the section 7. If $E[W(i)]$ is comparable to 2.5 then there is a component of $\frac{\partial \lambda_T(i)}{\partial E[V_T]}$ which is proportional to λ_T and a component which is inversely proportional to propagation delay.

4.2. Case 2

We now present the effect of change of bandwidth on the throughput of different TCP-P or TCP-ON-OFF connections. Here the TCP traffic does not change but the bandwidth available to the TCP changes. Differentiating (1) w.r.t C, we get

$$\frac{\partial \lambda_T(i)}{\partial C} = \frac{\partial E[W(i)]}{\partial C}\frac{1}{(\triangle(i)+E[V_T])} - \frac{\partial E[V_T]}{\partial C}\frac{E[W(i)]}{(\triangle(i)+E[V_T])^2}. \tag{8}$$

Using (3) and (4), we can show (see [15])

$$\frac{\partial \lambda_T(i)}{\partial C} = -\sqrt{\frac{2}{3p_i}}\frac{E[V_T]}{(CE[V_T]-T_{min})(\triangle(i)+E[V_T])} - \sqrt{\frac{2}{3p_i}}\frac{C}{(CE[V_T]-T_{min})(\triangle(i)+E[V_T])}\frac{\partial E[V_T]}{\partial C} - \frac{E[W(i)]}{(\triangle(i)+E[V_T])^2}\frac{\partial E[V_T]}{\partial C}. \tag{9}$$

$\frac{\partial E[V_T]}{\partial C}$ can also be computed using similar methods. The details are provided in [15]. We have solved this ODE numerically in [15] for specific network scenarios and compared with proportional fairness. The two curves match quite well. This indicates that even in this case we obtain fairness in utilization of BW .

5. ADMISSION CONTROL

Although admission control is not implemented in the Internet today, it is essential to provide QoS guarantees. In our setup, we propose that admission control be used on connections/applications which demand QoS, i.e., TCP-P, TCP-ON-OFF, CBR and VBR but not on TCP-NP. Also, if a long-lived TCP connection does not really care about the QoS then it can be admitted as TCP-NP. TCP-NP traffic will be always admitted and will be routed as best effort traffic in Internet (with possibly a good congestion sensitive routing).

There is also the question of deciding at the time of connection setup how to classify a new TCP connection into TCP-P, NP or ON-OFF. One way could be based on the application it is supporting: Web browsing, streaming and FTP applications can be classified a long lived with classification TCP ON-OFF in the first two cases and TCP-P for the last case. An email application would be TCP-NP even if it may be sending a long file.

Next we explain how we are going to implement admission control for each of the classes. Whenever a new connection request comes, it will declare its QoS requirements. For CBR we only ask for the rate in bps it requires. For VBR traffic we require the maximum throughput in bps it needs. For TCP-P we require the minimum mean throughput (in packets/sec) needed and the packet length of the connection. A TCP-ON-OFF will specify the maximum mean download time of a file and the packet length. This can be translated into the minimum throughput it requires during its ON time. We do not need anything from the TCP-NP.

Whenever a new connection request comes, which is not TCP-NP, the network will try to find a route from the source to the destination such that on all the core routers on the way there is sufficient BW for that connection. A core router will decide whenever a request reaches it if it has the required resources (the procedure for this will be described below). If a route is found on which each of the routers accepts the request, the new connection will be accepted by the network; otherwise not.

If Q_1 or Q_2 has more BW than it needs for its minimum QoS requirements, the average queue length (calculated by RED algorithms) in Q_1 will become less than V_d. Then the mean window sizes of TCP connections will increase. This will increase their throughputs causing full utilization of link (unless it is a very light traffic situation). However the average queue length $\hat{V}_d$ at that time will still be less than V_d. This is because as $\hat{V}_d$ approaches V_d, the drop probabilities of TCPs will increase and will approach their minimum throughput requirement. Thus, to know if an output link at a router has extra BW or not (in either of the queues) it is sufficient to see the average queue length $\hat{V}_d$ in Q_1 at that time. This will be available at a router via the RED control algorithms. When a new request comes and $\hat{V}_d \geq V_d$, the request will be denied. When $\hat{V}_d < V_d$ we need to decide if the extra BW is sufficient to provide QoS to the new request, without compromising the QoS of existing connections. In the following we discuss this issue.

Suppose the new request is by a CBR source needing $\hat{C}$ bits/sec. If $\hat{V}_d < V_d$, we would like to check if there is sufficient BW available for the new request. $\frac{\partial EV_T}{\partial C}$ computed in [15] could be helpful here. But this required detailed information about traffic flows. Thus we consider a simpler solution. Since CBR traffic will usually be supporting voice, the BW required by one CBR connection will be very small and will not affect the QoS of other connections in Q_1 in any significant way. Therefore, we can admit a new CBR request if $\hat{V}_d \leq V_d - \varepsilon$ for some specified small $\varepsilon > 0$; otherwise not.

Next suppose a new TCP connection arrives requesting a minimum throughput of λ_T packets/sec and a packet length of s bits. Let $\hat{V}_d$ be less than V_d. By Little's law, the new connection will add (approximately) $\lambda_T \hat{V}_d/C$ more packets to the queue where C is the BW available to all the TCP connections in Q_1. If $\hat{V}_d + (\lambda_T \hat{V}_d/C)s \leq V_d$, then accept the request otherwise not. The actual addition of packets to the queue will be more than $\lambda_T \hat{V}_d/C$ because it will correspond to the new mean queue length. However it will be somewhat compensated by the fact that with increased average queue length the throughputs of the existing TCP connections will decrease (but will still remain higher than their minimum requirements).

If the new request is from a TCP-ON-OFF, it will only specify its mean throughput requirement λ_T during its ON period (which will be determined by the maximum mean download time for its files). Its long term mean throughput requirement will be less than this. However since the user may not know its ON-OFF statistics, we do not demand it at the time of connection setup. Thus, admission decision for such a connection will be made based on using this λ_T in the procedure described in the last paragraph. Although it is a conservative decision, these extra margins do not add up with time. This is because every new decision by the router is based on its $\hat{V}_d$ at that time which depends on the actual long term throughput requirements of the existing TCP-ON-OFF connections and not on their throughput demands during their ON periods.

Next we consider a new VBR request. If the router link is not in congestion (indicated by average queue length being less that T_{min}), accept the request; if the total peak rate of existing VBR connections on the router is greater than a prespecified threshold (can be specified to be a fraction of link speed), reject the request. Otherwise, if C_p is the peak rate requirement of the new connection, we compute the effect of admitting it on existing TCP connections in Q_1 by our proportional fairness result. If $\lambda(i)$ is the current throughput in bps of connection i and λ bps is the total throughput obtained by all TCP connections in Q_1, then connection i will lose atmost $\lambda(i)C_p/\lambda$ bps throughput. If $\lambda(i)(1-C_p/\lambda) \geq \lambda_T(i)$for each i, admit the new connection otherwise not.

The admission control scheme provided above will require some signaling support. In IETF there are discussions [17] about providing such support.

In [15] we have verified the effectiveness of the admission control scheme suggested here. Due to lack of space these results are not included in this paper.

6. DIFFSERV IMPLEMENTATION

In this section we explain the implementation of our scheme in the DiffServ domain. This shows that our scheme can be implemented in the current Internet and also if DiffServ architecture is available. We consider the cases of TCP traffic and UDP traffic separately.

At the edge routers one can use traffic markers, shapers etc. as suggested in DiffServ for the CBR and VBR Traffic. For both of these traffic we will use EF (Expedited Forwarding). Although in DiffServ there is only one DS-Point 101110 assigned to EF class, we can use an unused DS-Point (e.g., 111111) to accommodate CBR and VBR classes. For CBR and VBR traffic the 'out of profile' packets can be dropped. At the core routers CBR and VBR traffic is treated as explained before.

For TCP traffic that requires QoS, we use AF. We will classify our TCP connections into different AF classes. We just need the packet classifier. Packet conditioners are not required for the TCP traffic. These will unnecessarily cause extra delay and distort the window flow control mechanism of the TCP + RED. The TCP classes are needed to use different RED algorithms on them. Thus we do the classification based on their mean window size (packet loss) requirements. In AF PHB we can have only 12 classes. Let W_i $i = 1, 2, ..., N$ be the values of the mean window size quantization levels, with $W_1 = 1$ and $W_i, i \geq 2$, to be decided below. If EW required for a new connection is such that $W_i < EW \leq W_{i+1}$ then we will assign it the window W_{i+1}. This ensures that a user gets its minimum throughput. However for efficiency reasons we do not want to overprovision by a large margin. We divide the range in such a way that $W_{i+1} = 1.25W_i$. This makes sure that the over provisioning of the bandwidth is constant in every partition with a maximum of 25%. This is consistent with our proportional fairness concept.

For TCP-NP we use the "default" PHB which is nothing but the "Best Effort" approach. We do not use any profiler or conditioner for this traffic.

The DiffServ architecture also mentions about admission control although no specific recommendations are given. Thus the approach suggested by us can be included in the DiffServ architecture. The admission control, will also require additional signaling support. This is being developed in more recent Internet charter [17].

We have simulated this system in NS. Due to lack of space these results are not included in this paper but are available in [15]

7. SIMULATIONS

In this section we present simulation results for the scheme proposed in this paper. The simulations were done using NS-2.6. Due to lack of space we present only a few results. Specifically, results related to admission control and DiffServ implementation are not provided in this section; these are available in [15].

Briefly we mention the network simulated. We consider a system with two routers in tandem. The first link has a bandwidth of 28Mbps in which Q_1 utilizes 17Mbps. The second link has a link speed of 35 Mbps in which Q_3 (the first queue of second router) takes 16 Mbps. There are 3 classes of TCP-P connections. Class 1 passes through only the first router, class 2 passes through both the routers and class 3 passes through only the second router. There are 5 TCP connections in each class. The packet sizes of these TCP classes are 400, 700, and 1000 bytes. The Δs for the 3 classes are 15, 10 and 25 msec. We have 3 classes of TCP-ON-OFF connections each class consisting of 3 connections and the three routes as above. Their packet sizes are 800, 1000 and 400 bytes. The off time for TCP-ON-OFF connections is exponentially distributed with mean 1sec. For a particular connection, the file size when a new ON period starts could be 20, 200, 2000 packets with equal probability. $\Delta = 20, 5$ and 30msec. The desired values for the throughput for TCP-P and TCP-ON-OFF are shown in Table 1.

The network has six UDP-CBR connections, two connections each following the three possible routes. The rate requirements of the UDP-CBR are 100, 141.9, 120, 300, 70, 263.3 pkts/sec. The packet sizes of the six UDP classes are respectively 50, 1000, 90, 1500, 80 and 900 bytes. There are 3 VBR connections, one taking each routs. The traffic of each VBR connection is modeled as an MMPP (Markov Modulated Poisson Process) with two states: in state 0 the mean sojourn time is 0.001 sec and in state 1 it is 0.01sec. Their packet sizes are 100 bytes and their rates (during state 1) varied between 500 pkts/sec to 1500 pkts/sec. The TCP-NP connections arrive as a Poisson process with mean 72 connections/sec. For these connections, $W_{max} = 5$ and Δ = 5msec. They are divided into four classes. The packet sizes of the four classes are 40, 800, 500 and 1500 bytes respectively. The corresponding file sizes are 10, 20, 50 and 30 packets. A new TCP-NP connection belongs to any of these classes with equal probability.

We use RED control on TCP-P and TCP-ON-OFF connections with the parameters in Q_1, T_{min}= 37.15, T_{max}= 446.3, 309.8, (for TCP-P) 384.72, 216.9 pkts (for TCP-ON-OFF). and the RED parameters for the Q_3 are T_{min} = 24.8, T_{max}= 157.1, 107.2, (for

TCP-P) 46.8, 124.3 pkts (for TCP-ON-OFF). The instantaneous workload for Q_1 is shown in Fig 1 and for Q_3 in Fig 2. One observes that the queue lengths in Q_1 and Q_3 oscillate around the desired queue lengths.

From Table 1 one observes that the different TCP-P and TCP-ON-OFF connections are getting the desired throughputs. Also the mean download times of TCP-ON-OFF, as shown in Table 2 are close to the desired values.

In [15] the performance of this modified scheme is compared to that in [13], [14]. It is found that UDP-CBR and UDP-VBR delays and delay-jitters are negligible in the present case compared to the approach in [13], [14]. But, the TCP throughputs and download times in the present approach are comparable to that in [13], [14].

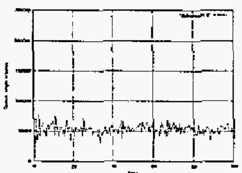

Fig 1. Workload in Q_1.

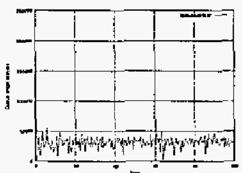

Fig 2. Workload in Q_3.

Table 1:Throughputs (pkts/sec)

TCP classes	TCP throughput	
	Theory	Sim
1	250	232.4, 258.85, 245.22, 264.19, 245.07
2	125	126.6, 125.66, 132.34, 140.81, 133.93
3	75	70.58, 61.56, 56.64, 71.43, 75.72
TCP-ON-OFF		
1	129.99	121.21, 118.68, 118.36
2	88	85.71, 86.02, 84.22
3	157.5	155.26, 166.1, 170.36

Table 2: Download times (secs) of TCP-ON-OFF

Down load times	file size	theory	sim.
TCP 1	20	0.133	0.191
TCP 1	200	1.33	1.33
TCP 1	2000	13.33	14.4
TCP 2	20	0.2	0.22
TCP 2	200	2	2.25
TCP 2	2000	20	20.36
TCP 3	20	0.1	0.121
TCP 3	200	1	1.02
TCP 3	2000	10	9.46

7.1. Fairness In Non-stationary scenario

In this simulation we consider a system with one bottleneck router. Here our aim is to see the performance of our scheme under dynamic traffic conditions and also to check for fairness. There are twelve TCP-P connections sharing the bottleneck link with four UDP connections in Q_1. TCP-NP is present in Q_2. The bottleneck link speed is 13.4556 Mbps for Q_1. Q_2 has been given a bandwidth of 12 Mbps. The TCP-P traffic consists of two classes, class 1 has seven connections and class 2 has five with packet lengths 500 and 1000 bytes respectively. The Δs are $15ms$ and $25ms$. The QoS requirements are : $E[V_T] = 0.035$s, $\lambda_T(1)$ = 150 pkts/s, $\lambda_T(2) = 85$ pkts/s. There are four UDP-CBR connections with packet lengths of 600,400,1000,1500 bytes. The throughput of UDP-CBR are 125, 312.5, 350, 121.3.

TCP-NP connections arrive as a Poisson process with rate 70 per seconds. The file size of a connection is chosen with uniform distribution over the four options: 10 packets with packet size 40 bytes, 20 packets, 800 bytes, 50 packets, 500 bytes and 30 packets with 1500 bytes.

In the simulation every 100 seconds TCP-P, TCP-ON-OFF and/or UDP-CBR are changing. We compare the actual throughput with the predicted value in Fig 3 for class 1 and Fig 4 for class 2. The theory generally follows the simulation results quite closely.

Actual and desired throughputs vs queue length

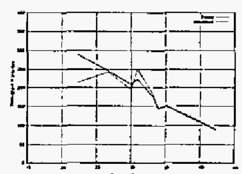

Fig 3. Class 1

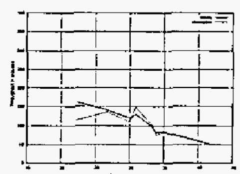

Fig 4. Class 2

8. REFERENCES

[1] D. Bertsekas and R. Gallager. Data Networks. Prentice-Hall, Englewood Cliffs.

[2] S. Blake, D. Black, M. Carlson, E. Davies, W. Wand, Weiss, An architecture for differentiated services, IETF RFC 2475, Dec 1998.

[3] R. Braden, D. Black, S. Shanker, Integrated services in the Internet architecture: an overview, IETF RFC 1633, June 1994.

[4] N. Christin and J. Liebrherr, A QoS architecture for quantitative service differentiation, IEEE Commun. Magazine, Vol. 41, June 2003, pp.38-45.

[5] S. Floyd and V. Jacobson, Random early detection gateways for congestion avoidance, IEEE/ACM Trans. Networking, Vol. 1, 1993, 397-413.

[6] G. Huston, Internet Performance Survival Guiding QoS strategies for Multiservice Netowrks, J. W., N. Y. 2000.

[7] F. Kelly,A.K.Maulloo and D.K.H. Tan, Rate control for communication networks: Shadow prices, proportional fairness and stability, Journal of the Operational Research Society, Vol. 49 1998.

[8] S. Kunniyur and R. Srikant. Analysis and design of an adaptive virtual queue algorithm for active queue management. In Proceedings of ACM Sigcomm, pages 123 134, San Diego, CA, August 2001.

[9] R. R-F. Liao and A. T. Campbell, Dynamic core provisioning for quantitative differentiated services, IEEE/ACM trans. Networking, Vol12, 2004, 429 - 442.

[10] V. Misra, W.B. Gong and D.Towsley, Fluid based analysis of network of AQM routers supporting TCP flows with applications, Proc. ACM SIGCOMM, 2000, 1714-1726.

[11] J. Padhye, V. Firoiu, D. Towsley and J. Kurose, Modelling TCP throughput: A simple model and its empirical validation, Proc. ATM SIGCOMM 98.

[12] V. Sharma and Punyaslok, Performance analysis of TCP connections with RED control and exogenous traffic, IEEE GLOBECOM, 01.(Detailed version in Queueing systems, Vol. 48, 2004, 193–235).

[13] V. Sharma and M.B. Suma, Providing QoS to Real Time and TCP Connections Via Rate Control of UDP, National Comm. conf.(NCC), Bangalore, Jan 2003.

[14] V. Venugopal Reddy, Vinod Sharma and Suma M.B., Providing QoS to TCP and real time connections in the Internet, Queueing Systems, Vol. 46, 2004, 457–476.

[15] Vihang Kamble, M.E. thesis, ECE depart. Indian Insttiute of Science, Bangalore, June 2004.

[16] M.Vojnovic,Le Boudec, C. Boutremans. Global fairness of additive-increase and multiplicative-decrease with heterogeneous round-trip times INFOCOM 2000.

[17] Next Steps in Signaling (nsis), IETE Charter, http://www.ietf.org/html.charters/nsis-charter.html.