High Performance DiffServ Mechanism for Routers and Switches: Packet Arrival Rate Based Queue Management for Class Based Scheduling

Bartek Wydrowski and Moshe Zukerman

ARC Special Research Centre for Ultra-Broadband Information Networks,

EEE Department, The University of Melbourne,

Parkville, Vic. 3010, Australia

{ b.wydrowski, m.zukerman }@ee.mu.oz.au

Abstract. This paper introduces a technique for applying packet arrival rate based queue management to class based scheduling algorithms. This enables a DiffServ architecture with very low packet latency, loss, and high link utilisation. Simulation results demonstrate that the proposed technique outperforms the current weighted random early drop (WRED) and weighted fair queue (WFQ) architecture.

1 Introduction

At the core of the Internet's Differentiated Services (DiffServ) architecture are the packet scheduling and queue management algorithms in routers or switches. Today's premier DiffServ architecture consists of a weighted fair queue (WFQ) with weighted random early drop (WRED) queue management. However, literature has shown that performance of packet arrival rate based congestion control, such as REM [5] or GREEN [8], significantly outperforms packet backlog based techniques such as drop-tail or RED. In this paper we form a basis for a high performance DiffServ architecture by applying rate-based queue management to packet scheduling algorithms. In the following subsections we give an overview of the area and show why rate-based control with packet scheduling is desirable.

1.1 Congestion Control Overview

Asides from the physical capacity of the network, the key design component that determines the quality of service of packet networks is load control. Load control determines how many packets are allowed onto each link of the network, who gets to send them and when. This controls the bandwidth, latency and jitter experienced by users.

There are a number of load control mechanisms, characterised by the amount of connection state information stored in the network. The range goes from connection admission control schemes, such as RSVP, through to stateless congestion control such as TCP, which is a subset of a more general macro-economic like system [7]. Diffserv occupies a middle ground, where individual connections are controlled on a connectionless/stateless basis from the perspective of the network, but aggregates of

E. Gregori et al. (Eds.): NETWORKING 2002, LNCS 2345, pp. 62-73, 2002.

flows, i.e. classes, receive pre-configured treatment at links, which require per-class information. This results in a good compromise between system complexity and control of performance.

Congestion control on an Internet with *Diffserv* is performed by two independent and concurrent mechanisms: (1) a closed-loop control mechanism controls the transmission of packets onto the end-to-end source to destination paths, and (2) open-loop packet scheduling algorithms, enforce statically pre-configured prioritisation and allocation of bandwidth at each link.

The closed-loop congestion control system consists of source algorithms controlled by link algorithms. The source algorithm is any protocol which transmits onto the Internet (e.g. TCP, UDP, RTP etc.) and is not necessarily responsive to congestion. The link congestion control algorithm is sometimes called queue management or active queue management (AQM). Examples of AQM algorithms include drop-tail, Random Early Drop (RED) [3] (and variants: WRED [1], GRED [14] etc.), Random Exponential Marking (REM) [5], Blue [4] and GREEN [8]. The AQM algorithm signals congestion to the source by packet marking, namely, explicit congestion notification (ECN) or packet dropping.

The open-loop scheduling algorithms determine which packet to send next [6]. They decide the order of packet transmission based on the order of packet arrival and the packet priority class. DiffServ uses a 3 bit code in each packet to identify the class. Scheduling of the packet controls the order of transmission as well as the relative bandwidth allocation to each class. Examples of scheduling disciplines include First In First Out (FIFO), Round Robin (RR), Priority Scheduling (PS) and Weighted Fair Queueing (WFQ).

1.2 Need for Scheduling: Classless vs. Class Based Differential Service

A number of papers [5] [7] have proposed an architecture for differentiated services without explicit packet classes. Instead, sources differentiate their demand for bandwidth by utility functions, $x = U(p)$, which determine the source's transmission rate x based on the current network price p. The price p, is determined by the end-to-end congestion level, and is communicated to sources from the AQM algorithm by packet marking or dropping. In fact, the network functions as a macro-economic system, where links sell their bandwidth and sources purchase it, based on their utility function. Sources which require more bandwidth than others, simply send more, suffering a higher price p. In such a system, scheduling algorithms are redundant because the allocation of bandwidth is determined solely by the macro-economic process. It has been shown that such a system maximises the aggregate of the utilities of all the sources [7].

If maximising the aggregate utility of the system is the only criteria, this system is sufficient. However, no guarantees can be made about the amount of bandwidth actually allocated to each source, because the current 'market' of all sources on the network determines this. A real network will consist of a subset of sources which require a minimum rate guarantee, and a subset which are satisfied by their 'market-share'. Since the network administrator is not aware of all of the utility functions of all the flows traversing the network, it is not possible to configure the utility functions of sources to guarantee their minimum rates in a competitive environment.

Many real applications require minimum rate guarantees. For example, an office with a set of voice-over IP telephones, an interactive online game, or video-conferencing, all require a guaranteed amount of bandwidth from the network at any time, regardless of the background traffic. If a subset of sources needs a guaranteed minimum rate from a link, the macro-economic system is not sufficient. A flow isolation mechanism, which removes the flows needing guarantees out of the competitive macro-economic environment that contains other flows with unknown utility functions, is essential to guarantee minimum rates. DiffServ with packet marking and link class-based scheduling does this by guaranteeing minimum capacity to flow subsets.

In practice, IP router manufactures have recognised this need for scheduling algorithms. However, the existing architecture for congestion control in a class-based environment remains crude. Until now, the closed-loop congestion control, or queue management, within the scheduling mechanism has been based on backlog measuring techniques such as drop-tail, RED or WRED. In this paper, a technique for implementing high performance rate based congestion control in a scheduler such as WFQ is introduced.

1.3 Need for Rate Based Control: Rate vs. Backlog Based Congestion Control

Broadly, there are two paradigms of congestion control algorithms, characterised by the way they observe congestion. Backlog based (BB) control, like droptail, RED and WRED, measures the number of packets in the buffer to determine the severity of congestion. Arrival rate based (RB) control schemes, such as REM or GREEN, measure the packet arrival rate.

In general, AQM algorithms signal congestion to the source algorithms by varying the rate of packet dropping or ECN marking, P. For BB control, P is a function of the backlog size $b(t)$ at time t, $P(t)=f(b(t))$; where $f(x)$ is a positive and increasing function for $x > 0$ and $f(0)=0$. For RB control, P is typically driven by an integration process, which sums the excess demand, such as $P(t+1) = P(t) + \Delta P \times (x(t) - u \times c(t))$, where $x(t)$ and $c(t)$ are the arrival and service rates at time t respectively, ΔP is the gain of the control which affects the stability and convergence, and u controls the target utilisation. Although BB congestion control is simpler to implement, it has some inherent limitations not present in RB control.

The backlog (queuing) process $b(t+1)=[b(t) + x(t) - c(t)]^+$, cannot observe long term arrival rates $x(t) < c(t)$ as if $x(t) < c(t)$ for a sufficient period of time, then b(t) reaches zero. Once $b(t)=0$, and if $x(t)$ continues to be less than $c(t)$ and b(t) remains zero, we can say nothing about how close $x(t)$ is to $c(t)$ by observing the state of $b(t)$. Therefore, by observing $b(t)$ the sources cannot be provided with feedback about the level of $x(t)$ to control their transmission rate, as $b(t)$ stays at zero and provides no information about $x(t)$. Given that a positive feedback signal P is required to control the source at some steady rate $x(t)$ where $x(t) < c(t)$, and $P(t)=f(b(t))$, the backlog must be positive, $b(t) > 0$, for P to be positive. This shows how BB control posits the existence of backlog and backlog is necessary for the control process itself.

Backlog is undesirable because it creates packet latency and delay jitter. Furthermore, delay in the congestion control system loop pushes the network towards instability, increasing the likelihood of buffer overflow and under-utilisation. Of course, some

backlog is necessary to achieve a desired utilisation of a link with a non-deterministic arrival process, however this is at worst equal to, but typically far less than, the backlog created by BB control such as drop-tail or RED [8].
Unlike the BB schemes, the RB control mechanism can observe *x(t)* directly. In a steady state situation, where the input process is stationary, the amount of backlog kept can therefore be only the minimum required to achieve the desired utilisation. It is not the intention of this paper to give a thorough performance comparison of different congestion control strategies, only to indicate some of the reasons why it is desirable to have a RB control strategy. For more background, the reader is referred to [5] [8].
Now that we have presented the need for (1) class based scheduling algorithms and (2) RB control, the algorithm which combines the two is presented in Section 2 and its performance evaluation is presented in Section 3.

2 Algorithm Background

RB AQM operates in symbiosis with a scheduler. Our proposed design of RB AQM applies to a work conserving WFQ like scheduler. A work conserving scheduler is never idle if there are any packets in any queue. A WFQ like scheduler, such as RR and many variants of WFQ, allocates a portion of service time to each queue during an interval of operation. The scheduler is interfaced to by the enqueue and dequeue functions, which accept and provide the next packet for queuing or transmission respectively.

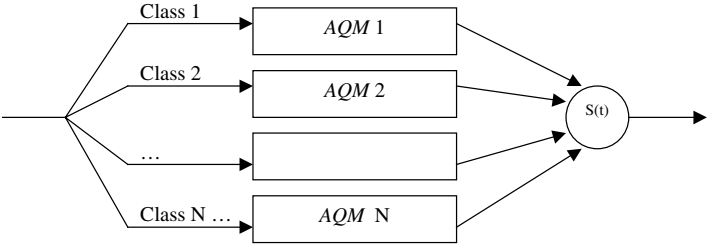


Fig. 1. RB AQM architecture in a class based scheduler

As shown in Fig. 1, each queue in the scheduler is managed by a separate instance of an AQM algorithm. The AQM algorithm decides which packets to drop or ECN mark. Packet marking/dropping gives the source algorithm a feedback signal which controls its transmission rate and avoids queue overflow or excessive backlog. Traditionally, this would be performed by BB control, such as drop-tail or RED queue. RB control directly replaces these algorithms. In general, RB AQM is any process which determines the packet marking/dropping rate, *P(t)*, from at least the packet arrival rate *x(t)* and capacity *c(t)*. Typically, the process for *P(t)* is an integrator of excess demand [10], $P(t+1) = P(t) + \Delta P \times (x(t) - u \times c(t))$, however, other functions are possible, motivated by better convergence or stability (eg: REM, GREEN).

$$P_i(t+1) = AQM(c_i(t), x_i(t), P_i(t), ...) \qquad 1 \geq P_i(t) \geq 0 \tag{1}$$

The distinctive issue, faced by RB AQM in a class-bases scheduler, is that the capacity available to each class i, denoted c_i, and the packet arrival rate for that class, denoted x_i, need to be known. In work conserving scheduler, such as WFQ, where unused capacity in one class is redistributed to other classes, the capacity available to each class is time-varying and depends on, and affects, the traffic in other classes. This paper enables RB AQM by presenting a technique for calculating and controlling c_i, the capacity allocated to each class. Class Dimensioning, or controlling the number of users per class is beyond the scope of this paper.

A basic algorithm is introduced in Subsection 2.1 which results in a functional work conserving RB system, where each class is guaranteed its minimum share, M_i. However, the capacity above the minimum is not distributed with any notion of fairness. Instead, the classes with the most aggressive traffic win the slack capacity. In Subsection 2.2, we present a notion of proportional fairness, and a mechanism to enforce it.

2.1 Basic Algorithm

2.1.1 Capacity Estimation

Consider a stream of packets scheduled by a work-conserving WFQ scheduler, of N classes. Let B be the vector representing the sizes (bits) of the H packets that have been served most recently. The order of the elements of vector B are in reverse order to their service completion times. In other words, B_0 is the size of the most recently served packet, B_1 is the size of the previous packet and so on. Finally, B_H is the size of the oldest packet packet in B. Similarly, we define the vector C, of H elements, such that C_j is the class (Cj $\in$ {1, 2, 3, … N}) of the packet represented by B_j, j = 1, 2, 3, … H.

Let $S(t)$ be the physical capacity of the link at time t. When $S(t)$ is time varying, such as with Ethernet, DSL, or radio, it can be estimated from the last packet's transmission time. The scheduling algorithm, such as WFQ, may guarantee minimum rates to each class. Let W be a vector whose element W_i corresponds to the share of capacity that each class i is guaranteed. For a WFQ scheduler, W_i corresponds to the service quantum for class i.

In a work conserving scheduler, the actual capacity available to a class depends on the traffic in other classes as well as on the minimum rate allocation W. Without apriori knowledge of the traffic, the future capacity available to a class, can only be estimated from the previous capacity. Let the identity function $I(j,i)$ be:

$$I(j,i) = \begin{cases} 1 & \text{if } C_j = i \\ 0 & \text{otherwise.} \end{cases} \tag{2.1}$$

The estimate class capacity, $S_i(t)$, is calculated from the portion of server time allocated to class i by the scheduling mechanism in the past H packets:

$$S_i(t) = \frac{\sum_{j=0}^{H} B_j(t) \cdot I(j,i)}{\sum_{j=0}^{H} B_j} S(t) \qquad \text{where} \quad i < N. \tag{2.2}$$

Note reduced complexity techniques such as exponential averaging could be employed to compute (2.2).

2.1.2 Capacity Allocation

The minimum service rate guaranteed by the WFQ scheduling mechanism, M_i, is given by:

$$M_i(t) = \frac{W_i}{\sum_{j=1}^{N} W_j} S(t) \cdot \tag{3}$$

The capacity allocated to each class is therefore also bounded by the minimum rate enforced by the WFQ scheduling policy. The capacity allocated to class i, denoted $c_i(t)$, is:

$$c_i(t) = Max(M_i(t), S_i(t)). \tag{4}$$

Notice that $c_i(t)$ is the capacity allocated to class *i*, not the capacity actually consumed by class *i*. The capacity not consumed by the class to which it is allocated, may be used by other classes. If for example, no class *i* packets arrive, $s_i(t)$ will be 0, and $c_i(t)=M_i(t)$. Although in this case no capacity is consumed by class *i*, if a burst of class *i* packets were to arrive, $M_i(t)$ capacity is guaranteed. Note (4) is evaluated at each update of the AQM process (1), which at the maximum rate, is at every enqueue event.

2.2 Extended Fair Share Algorithm

The algorithm in 2.1 is extended here to enforce a notion of proportional fairness. The fair allocation enforcement applies only to bottlenecked classes, where $x_i(t) \geq c_i(t)$. Classes which are not bottlenecked at the link, $x_i(t) < c_i(t)$, need no enforcement of fairness, since their rate is below their fair capacity and their bandwidth demand is satisfied. We define a fair allocation of capacity to a bottlenecked class i, $F_i(t)$, as:

$$F_i(t) = \frac{W_i}{\sum_{j=all\ \ bottlenecked\ classes} W_j} (S(t) - \sum_{j=all\ non-bottlenecked\ classes} x_j). \tag{5}$$

In the extended algorithm, the capacity of non-bottlenecked classes is given by (4), and for bottlenecked classes, the capacity is given be (5). Notice that the sum of $c_i(t)$ for non-bottlenecked by (4) and $F_i(t)$ by (5) may be more than $S(t)$. However, the non-bottlenecked classes do not utilise their allocated capacity $c_i(t)$, and the aggregate arrival rate is controlled below the capacity $S(t)$.

3 Implementation and Transient Performance Evaluation

3.1 Implementation

For class i, RB control was implemented in a WFQ scheduler with a variation of GREEN as the AQM algorithm, as follows:

$$P_i(t) = P_i(t) + \Delta P_i(t) \cdot U(x_i(t) - u_i \cdot c_i(t)). \tag{6.1}$$

where

$$U(x) = \begin{cases} +1 & x \geq 0 \\ -1 & x < 0 \end{cases} \tag{6.2}$$

and

$$\Delta P_i(t) = \max(abs(x_i(t) - u_i \cdot c_i(t)), k). \tag{6.3}$$

where u_i controls the target utilisation and hence also the level of queuing, and k is a constant which limits the minimum adjustment to $P_i(t)$, to improve convergence. The values of $P_i(t)$, $x_i(t)$ and $c_i(t)$ are updated with every class i packet arrival. The pseudo-code for the WFQ scheduling algorithm used is:

```
pkt* wfq.deque()
{
while(TRUE)
{
        for I = 1 to N {
                if (class[I].nextpkt.size < S[I])
                {
                        S[I] = S[I] - class[I].nextpkt.size();
                        return (class[I].dequeue);
                }
        }
        for I = 1 to N {
                if (S[I] < MaxS );
                        S[I] = S[I]  + W[I];
        }
}
}
wfq.enque(pkt *packet)
{
        class[packet.class].enque(packet);
}
```

Fig. 2. Low jitter WFQ scheduler

This particular WFQ variant minimizes the jitter of higher priority classes, lower class number. The *wfq.deque* function is invoked when the link is ready to transmit the next packet and the *wfq.enque* function is invoked when a packet is received for transmission onto the link. A packet queued in a higher priority class will always be

served next, so long as the class's work quantum, *W*, has not been exceeded. Note that the function *class[I].nextpkt.size* returns the size [bits] of the next packet in class *I*, or infinity if there are no packets left in the class. The constant *MaxS* controls the maximum burst size allowable to be transmitted in a class that has been previously idle.

3.2 Performance Evaluation

The system was simulated using Network Simulator 2 [9]. Three scenarios simulated are presented in this paper. All scenarios used the same network topology, as depicted in Fig. 3. For Scenarios 1 and 2, the Diffserv managed link *X* has a 1 Mbps capacity and it is 2Mbps in Scenario 3. Multiple TCP or UDP sessions are aggregated to form the traffic of each of the four classes presented to the link. All data packets are 1000 bytes. We will now describe each simulation scenario and the results.

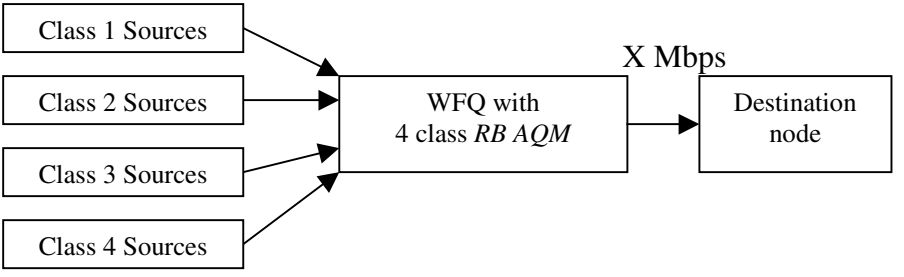


Fig. 3. Overview of Simulation Topology

Scenario 1A and 1B: TCP Traffic

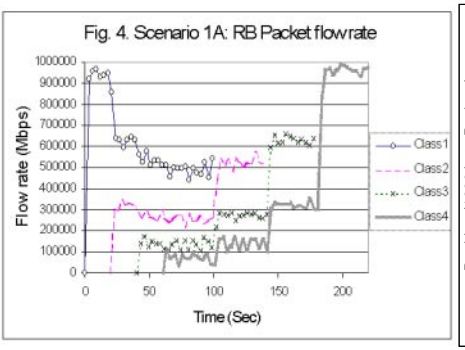
The traffic of this scenario consists only of TCP sources. Scenario 1A uses RB and WFQ with the fairness enhancement (5). Scenario 1B uses WRED and WFQ. The flow rates of traffic in each class and the total number of packets backlogged for all classed was measured. The parameters for this scenario are listed in Table 1.

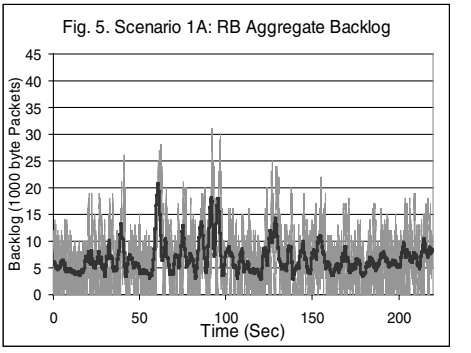
Table 1. Simulation Parameters for Scenario 1

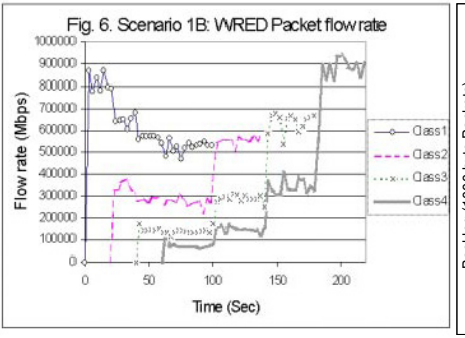
Class	u_i *Utilisation*	W_i	Sources	Start (sec)	Stop (sec)
1	0.93	8001	8 TCP 40ms RTT	0	100
2	0.93	4001	8 TCP 40ms RTT	20	140
3	0.93	2001	16 TCP 40ms RTT	40	180
4	0.93	1001	16 TCP 40ms RTT	60	220

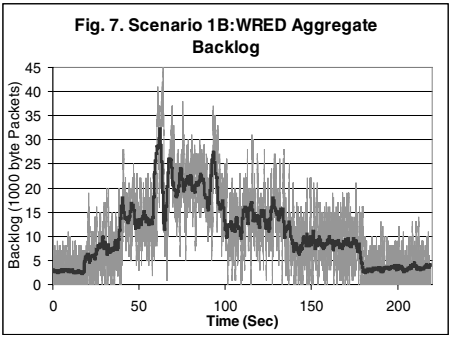
The WRED implementation uses a weighted average of backlog, denoted $B_w(t)$, to determine the packet marking/dropping probability. The marking probability is related linearly to $B_w(t)$, by $P(t) = \alpha B_w(t)$, where α is the reciprocal of the maximum queue size q. In Scenario 1B q equals 10.

Fig. 4 confirms that a fair allocation of capacity is achieved with the RB and WFQ, as the magnitude of the flow rate from each class is proportional to its minimum rate *W* when the traffic from different classes is switched on and off.
Figures 5 and 7 show the backlog of the RB and BB (WRED) system, with the thick black line being the average backlog measured over 300 packets. The figures illustrate the poorer queuing performance of WRED and WFQ compared to RB and WFQ congestion control. In the interval 50s to 100s, when all classes are active, note how backlog increases with increasing traffic load. This illustrates the previous analysis, that with BB control where $P(t)=f(b(t))$, backlog is necessitated by the control system. With increased traffic load, the feedback signal $P(t)$ must also increase to control the sources, and since $P(t)$ is coupled with backlog, the backlog must also increase. Compare this with RB congestion control in Fig. 5, where the backlog varies about 0 regardless of the traffic.







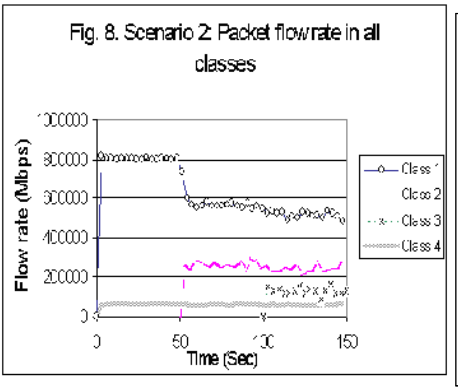


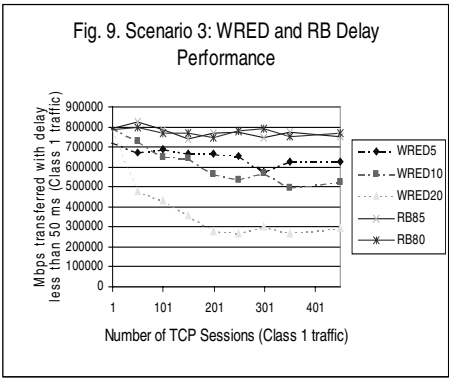
Scenario 2: TCP and UDP Traffic

This traffic scenario consists of both UDP and TCP sources. Classes 1 and 4 are UDP constant bit rate sources transmitting at 0.8 Mbps and 0.05 Mbps respectively. UDP sources ignore congestion notification. Classes 2 and 3 are comprised of TCP sources. For the complete parameters refer to Table 2.

Fig. 8 shows that RB control allocates bandwidth fairly, despite the presence of an unfriendly, non-congestion-controlled UDP sources. Notice that at 50sec, when Class 2 traffic is switched on, the UDP traffic in Class 1 is throttled down to its fair share by an increased packet dropping rate. At this point Class 1 becomes a bottlenecked class.

In this way, the TCP sources can attain their fair share despite the aggressive UDP source.





Scenario 3: Real-Time Traffic

In this scenario, it is demonstrated how a RB Diffserv architecture outperforms BB control for real-time traffic. Two classes are used to simulate the interaction of data traffic and real-time traffic. Class 2 contains TCP/FTP data traffic, and is insensitive to delay. Class 1 is the real-time traffic, with a hard maximum queuing delay requirement of 50 ms. The traffic in Class 1, the real-time traffic, consists of saturated TCP transfers, with the number of sessions increasing linearly from 1 to 450. A number of trails were simulated, using WRED with queue size value q set to 5 (WRED5),10 (WRED10) and 20 (WRED20) packets, and using RB control with parameter u_1 set to 0.8 (RB80) and 0.85 (RB85). The Diffserv link capacity is 2Mbps, with 1Mbps assigned to Class 1 and 1Mbps assigned to Class 2.

Table 2. Simulation Parameters for Scenario 2

Class	u_i *Utilisation*	W_i	Sources	Start (sec)	Stop (sec)
1	0.93	8001	1 UDP 20ms RTT 0.8 Mbps	0	150
2	0.93	4001	16 TCP 20ms RTT	50	150
3	0.93	2001	16 TCP 20ms RTT	100	150
4	0.93	1001	1 UDP 20ms RTT 0.05 Mbps	0	150

Table 3. Simulation Parameters for Scenario 3

Class	u_i *Utilisation*	W_i	Sources	Start (sec)	Stop (sec)
1	0.8, 0.85	2001	50-450 TCP 40ms RTT	0	450
2	0.95	2001	8 TCP 40ms RTT	0	450

TCP is used to approximate a real-time adaptive multi-rate source [11] [12] [13]. Audio and video protocols are typically based on UDP, RTP and RTCP. Recent real-time multimedia protocols respond to loss by adjusting their rate, and are thus in principle similar to TCP [11] [13]. Although their transient behaviour, and amount of

response to loss is different than TCP, any real-time protocol that seeks to take advantage of available capacity on a best effort network, must in principle be congestion controlled. Unless the real-time source increases its rate when there is available capacity, and decreases it when capacity decreases, the quality of transmission is suboptimal. Many existing CODECS are designed for varying channel conditions, such as a best effort network. For instance, the G.723.1 Audio speech codec adjusts its output rate, and adapts to the available bandwidth. Similarly, MPEG-4 includes extensive support for multi-layered, multi-rate video. The RTP communicates the amount of packets lost, which allows the sender to adapt its rate to the channel. At a bottleneck link, adaptive multimedia sources are like saturated sources, such as an FTP transfer, as the source always has more video or audio information that it could possibly send to improve quality.

In the simulation we measure the amount of packets, in Mbps, which are delivered with less than 50ms queuing delay in the Diffserv queue. Packets served late, >50ms, no longer contain useful information to a real-time application and do not contribute to the Mbps. Since real-time sources do not retransmit packets, the TCP packet retransmissions are considered as new packets in the simulation. The results, in Fig. 9, show how for a variety of settings, and traffic loads, RB control effectively delivers more useful data.

As discussed previously, the problem with BB schemes such as WRED, is that the backlog must be positive for source rate to be controlled. In this trial, the maximum queue size for WRED was reduced from 20 to 10 and then to 5. Reducing the maximum queue size gave diminishing returns since the utilisation was significantly lowered. On the other hand, increasing the queue size resulted in a higher average backlog, which delayed more traffic beyond the 50ms requirement. Also, as evident in Fig. 9, unlike RB control, the optimal setting of parameters for WRED varied widely with the traffic load. RB control was able to deliver more data in the delay specification, since it was able to control the arrival rate to some specified fraction below the service capacity, leaving spare capacity for the bursts in the traffic.

3.3 UDP: Throw Away – No Delay

In result in this section we focused on the possible disruptive effect of UDP traffic on TCP traffic, or the interaction between TCP traffic in different classes. An important issue is the performance of non-congestion controlled UDP traffic. UDP is typically used for real-time services with an upper bound delay requirement. If such traffic receives enough capacity, both BB and RB schemes function identically. However, when the amount of non-congestion controlled UDP traffic exceeds the capacity, BB schemes, such as WRED will increase backlog and delay, whereas RB control will prevent excessive delay by increasing the dropping rate. This means, that is instead of being excessively delayed, packets are discarded. Therefore in a congestion situation, the portion of packets which are transmitted, still meet the delay requirements. The portion which are discarded would likely not have been able to be served within the delay requirement. With WRED, in a congestion situation, the delay performance of all packets suffers.

4 Conclusion

We have presented a technique for applying rate based active queue management to a class based scheduling algorithm. The method presented is scalable, and low in computational complexity. It forms a solid architecture for DiffServ implementation in routers and switches and has been shown to outperform the current WRED with WFQ architecture. Furthermore, this work will enable the wide body of research into rate based congestion control schemes to be applied to improving the performance of DiffServ.

References

1. Cisco Systems Document, “Class-Based Weighted Fair Queueing”
2. Cisco Systems Document, “Low Latency Queueing”, http://www.cisco.com/warp/public/732/Tech/qos/techdoc/diffserv.shtml
3. S. Floyd and V. Jacobson, “Random early detection gateways for congestion avoidance” IEEE/ACM Transactions on Networking, 1(4):397--413, August 1993.
4. Wu-chang Feng, Dilip D Kandlur, Debanjan Saham Kang G.Shin, “Blue: A new class of active queue management”. Department of EECS University of Michigan
5. S. H. Low and D. E. Lapsley, “Optimization Flow Control, I: Basic Algorithm and Convergence”, *IEEE/ACM Transactions on Networking*, vol 7 part 6 pp861-875, Dec. 1999.
6. Internet Engineering Task Force IETF, “Recommendations on Queue Management and Congestion Avoidance in the Internet”, RFC 2309.
7. F. P. Kelly, A.K. Maulloo and D.K.H, “Rate control in communication networks: shadow prices, proportional fairness and stability”, Tan (Statistical Laboratory, University of Cambridge), *Journal of the Operational Research Society*, vol. 49, pp 237-252. 1998
8. B. Wydrowski and M. Zukerman, “GREEN: An Active Queue Management Algorithm”, 2001, (submitted for publication, available: http://www.ee.mu.oz.au/pgrad/bpw).
9. The Network Simulator - ns-2 homepage: http://www.isi.edu/nsnam/ns/
10. F. Paganini, J. C. Doyle and S. H. Low, “Scalable Laws for Stable Network Congestion Control”, submitted to CDC01. March 2, 2001.
11. J. Padhye, J. Kurose, D. Towsley, and R. Koodli, "A model based TCP-friendly rate control protocol," in Proc. International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV), Basking Ridge, NJ, June 1999.
12. I. Busse, B. Deffner, and H. Schulzrinne, "Dynamic QoS control of multimedia applications based on RTP," Computer Communications, Jan. 1996.
13. R. Rejaie, D. Estrin, and M. Handley, "Quality Adaptation for Congestion Controlled Video Playback over the Internet," Proc. of ACM SIGCOMM '99, Cambridge, Sept. 1999.
14. Anupama Sundaresan, Gowri Dhandapani, “Diffspec - A Differentiated Services tool”, The University of Kansas Lawrence, KS 66045-2228, December 19, 1999. http://qos.ittc.ukans.edu/DiffSpec/diffspec.html.