RZ 3374 (# 93420) 10/01/01
Computer Science 12 pages

Research Report

Bandwidth Allocation for Non-Responsive Flows with Active Queue Management

Ed Bowen, Clark Jeffries

IBM Microelectronics
Research Triangle Park, NC 27709
USA
{edbowen,clajef}@us.ibm.com

Lukas Kencl, Andreas Kind, Roman Pletka

IBM Research
Zurich Research Laboratory
8803 Rüschlikon
Switzerland
{lke,ank,rap}@zurich.ibm.com

Bandwidth Allocation for Non-Responsive Flows with Active Queue Management

Ed Bowen, Clark Jeffries

IBM Microelectronics, Research Triangle Park, NC 27709, USA

Lukas Kencl, Andreas Kind, Roman Pletka

IBM Research, Zurich Research Laboratory, 8803 Rüschlikon, Switzerland

Abstract

This paper addresses the problem of configuring active queue management systems (e.g. WRED and RIO) for service level specifications in Internetworks. In particular, we focus on Assured Forwarding (AF) for non-responsive flows in Differentiated Services networks. The difficulty is to determine the correct queue level thresholds that will result in correct drop rates for various AF precedence levels under any combination of offered loads.
A new active queue management scheme based on a control algorithm is proposed that senses not only queue levels but also rates of queue levels changes and per flow bit rates to converge automatically to an optimal set of transmit fractions. The scheme has been implemented and tested on a network processor. Results show that the new active queue management scheme protects assured aggregated flow rates during periods of congestion. For non-responsive traffic the buffer occupancy level remains low during 250% offered load.
Keywords: Differentiated Services, Active Queue Management, Quality of Service, AIMD

1 Introduction

The Differentiated Services architecture (DS) [1] is a scalable approach to service level specifications in IP networks based on traffic conditioning at the network edge and differentiated forwarding of aggregated traffic flows at the network core. With DS Assured Forwarding (AF) [2], three levels of drop precedences are specified. Depending on the actual sending rate and the minimum assured rate, packets are marked as belonging to one of the three AF levels when entering a DS domain. If the actual sending rate is below the minimum assured rate packets are marked as being *green*. If the actual sending rate exceeds the minimum assured rate packets are marked as either *yellow* or *red* [3, 4].

The AF drop precedence levels (or colors) are implemented using an active queue management algorithm at routers in the DS domain. This algorithm has to detect and respond to long-term congestion by discarding or marking packets. The algorithm should, however, not react to short-term bursts, and each precedence level has to be addressed differently. During long-term congestion, green traffic should receive the lowest discard rate, whereas red traffic should get the highest discard rate.

This work proposes a new approach to perform active queue management in DS domains. The presented algorithm overcomes the configuration problem with Random Early Detection (RED), which is—in variations—used as active queue management algorithm in most AF implementations. The difficulty with RED is to determine the correct drop precedence parameters for any combination of offered loads [5]–[7]. RED parameters are specified as thresholds for the average queue occupancy (which can also be regarded as delay thresholds) rather than in packet rates as would be convenient for configuration. If discard rates during long-term congestion cannot be correctly configured, service level specifications based on the AF service would potentially be violated. The active queue management algorithm presented here is configured with minimum and maximum flow rates. The drop rates are managed using a control loop with monitored individual offered loads acting as feedback. The unit of feedback is thus bits/s rather than bits.

Expedited Forwarding (EF) [8] is a low-jitter, assured bandwidth service in the DS framework. It must be ensured that the EF aggregate is served at a certain configured rate independent of the intensity of any other traffic attempting to transit a node. Thus, EF service behavior can be achieved by accurate policing at the ingress of the DS domain in conjunction with appropriate scheduling at transit nodes. Therefore, EF traffic can bypass the proposed adaptive control algorithm while AF and BE traffic should be flow-controlled by the new scheme.

The algorithm has been implemented in the IBM PowerNP network processor. The paper presents results from testing this implementation with non-responsive flows. The reason for focusing on non-responsive traffic is to learn how active queue management can be used to control the increasing amount of non-*TCP-compatible* (i.e. not responsive to congestion notification and in steady state more aggressive than TCP) traffic with Assured Forwarding. Non-TCP-compatible traffic is typically caused by UDP-based multimedia streaming and multicast applications.

The remainder of the paper is structured as follows: Section 2 introduces Random Early Detection including the variations typically used for implementing AF forwarding (e.g. WRED, RIO). The configuration problem these active queue management schemes entail is highlighted. The new approach to active queue management that addresses the configuration problem is presented in Section 3. Section 4 discusses results from testing the new active queue man-

agement algorithm in a network processor. Future work is identified in Section 5. Section 6 points to related work, and conclusions are given in Section 7.

2 Background

The stability of the Internet depends on congestion avoidance mechanisms at the edges of the network as well as at routers within the network [9]. For responsive flows, such as TCP flows, the end-to-end congestion control mechanism is typically combined with an active queue management system called Random Early Detection (RED) [10]. Simulations have shown that RED overcomes some of the congestion problems that occur with simple drop-tail queuing:

- Packet drops and delays are reduced because congestion is signaled to senders prior to periods of queue overflow.
- During congestion TCP flows are more likely to go into fast recovery because packets are not dropped consecutively.
- TCP flows are less likely to backoff and retransmit at about the same time (TCP global synchronization effect).

However, the effectiveness of RED in real networks has not been studied thoroughly. In fact, an analytical study of RED performance revealed that, in contrast to the simulation results listed above, the number of consecutive packet drops is higher with RED than with drop-tail queuing [11]. The study further shows that RED while reducing average delay, increases jitter. Another empirical evaluation concluded that active queue management based on RED offers no clear advantage over tail-drop with HTTP traffic [12].

Depending on the queue occupancy, the RED algorithm drops packets randomly with a given probability at routers. The probability is determined by the average queue length (see Figure 1). If the average queue length remains below a minimum threshold ($0 < Q_{\text{avg}} < q_{\text{min\_th}}$), no packets are dropped. If it is between the minimum and maximum thresholds ($q_{\text{min\_th}} < Q_{\text{avg}} < q_{\text{min\_th}}$), the drop probability increases linearly with Q_{avg} up to a maximum probability $p_{\max}$ at $q_{\text{avg}} = q_{\text{max\_th}}$. If the average queue length exceeds the maximum queue threshold ($q_{\text{max\_th}} < Q_{\text{avg}}$), all packets are dropped.

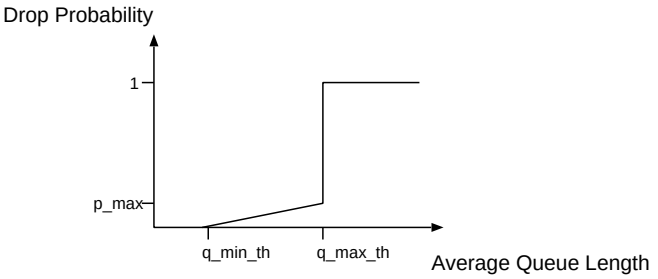

Figure 1: RED

Different variations of RED have been used to implement AF [13]–[16]. The principal objective is to define a drop probability function for each AF precedence level so that during congestion packets with a higher drop precedence value are discriminated against packets with a lower drop precedence value.

An active queue management scheme with a single average queue length but multiple sets of queue thresholds and maximum probabilities is referred to as Weighted RED (WRED). The queue threshold can be set to be fully overlapped, partially overlapped of staggered (see Figure 2).

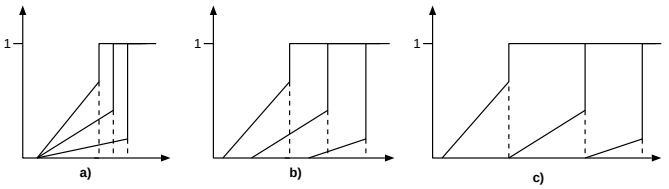

Figure 2: WRED variants: a) fully overlapped, b) partially overlapped, c) staggered

RED with separate average queues for each color and corresponding sets of minimum/maximum queue thresholds and maximum probabilities is called RIO (RED with in/out queues). With RIO the contribution of packets to a particular color in the current total queue level can be addressed with an individual set of parameters. In contrast, WRED operates only on the total queue size. In that respect, RIO is better suited to protect lower drop precedence flows against higher drop precedence flows [16].

Common to both active queue management schemes, WRED and RIO, is that the feedback signal is defined with a (color-blind/-aware) average of the queue length. This feedback signal is useful to sense congestion status, but should not be directly used to determine actual drop rates, because it is difficult for network administrators to find the correct parameter settings for any combination of offered loads in real networks. In order to show this difficulty with WRED, consider a simple case of two flows, one with an assured rate of 500 Mb/s and the other with no assured rate, both flowing into a single 1 Gb/s target port. Bandwidth allocation should proceed such that each aggregated flow gets its minimum and then half of the excess bandwidth. If both aggregated flows offer 1 Gb/s, then the first should get 750 Mb/s, the second 250 Mb/s. The first should transmit a ratio of 0.75 of its offered traffic, the second 0.25.

A scheduler would allocate these transmit ratios if supplied at all times with a surplus of frames of both aggregated flows. The problem is that with a finite buffer, flow control upstream of the scheduler must, on average, transmit and discard frames with the same ratios because a scheduler itself does not discard excess frames. Thus flow control must already solve the bandwidth allocation problem, at least approximately, to hand the correct combination of frames to the scheduler. The main strength of the scheduler would then not be the aggregated flow-wise bandwidth allocation but rather precise timing of frames within an aggregated flow.

The following example shows that finding WRED thresholds can be overdetermined for some combinations of offered loads. Suppose two flows converge into a single 1 Gb/s target port. Suppose Flow 1 is specified with a minimum assured rate of 500 Mb/s, and Flow 2 has no minimum no assured rate. Consider two congested Cases A, B with ideal bandwidth allocations I_1, I_2, offered loads O_1, O_2, and drop probabilities P_1, P_2 as given in Table 1.

The problem is now to tune the six parameters $q_{\mathrm{min\_th},1}$, $q_{\mathrm{max\_th},1}$, $p_{\mathrm{max},1}$, $q_{\mathrm{min\_th},2}$, $q_{\mathrm{max\_th},2}$, $p_{\mathrm{max},2}$ so that for Cases A and B some queue levels $Q_{\mathrm{avg,A}}$, $Q_{\mathrm{avg,B}}$ exist with the correct P_1, P_2 pairs of drop probabilities. Case A requires to find $Q_{\mathrm{avg,A}}$ such that

$$Q_{\mathrm{avg,A}} \leq q_{\mathrm{min\_th},1} \qquad \text{and}$$

Table 1: Example for overdetermined WRED configuration

Case	O_1 [Gb/s]	O_2 [Gb/s]	I_1 [Gb/s]	I_2 [Gb/s]	P_1	P_2
A	0.75	1	0.75	0.25	0	0.75
B	1	0.25	0.75	0.25	0.25	0

$$0.75 = p_{\max,2} \frac{Q_{\text{avg,A}} - q_{\min\_\text{th},2}}{q_{\max\_\text{th},2} - q_{\min\_\text{th},2}}$$

This implies:

$$Q_{\text{avg,A}} = 0.75 \frac{q_{\max\_\text{th},2} - q_{\min\_\text{th},2}}{p_{\max,2}} + q_{\min\_\text{th},2} \leq q_{\min\_\text{th},1}$$

In particular,

$$q_{\min\_\text{th},2} < q_{\min\_\text{th},1}$$

assuming $q_{\min\_\text{th},2} < q_{\max\_\text{th},2}$. Case B requires to find $Q_{\text{avg,B}}$ such that

$$Q_{\text{avg,B}} \leq q_{\min\_\text{th},2} \qquad \text{and}$$

$$0.25 = p_{\max,1} \frac{Q_{\text{avg,B}} - q_{\min\_\text{th},1}}{q_{\max\_\text{th},1} - q_{\min\_\text{th},1}}$$

This implies:

$$Q_{\text{avg,B}} = 0.25 \frac{q_{\max\_\text{th},1} - q_{\min\_\text{th},1}}{p_{\max,1}} + q_{\min\_\text{th},1} \leq q_{\min\_\text{th},2}$$

In particular,

$$q_{\min\_\text{th},1} < q_{\min\_\text{th},2}$$

assuming $q_{\min\_\text{th},1} < q_{\max\_\text{th},1}$, which is in contradiction to the implications in Case A.

As an alternative, individual flow queue thresholds might be proposed. The number of bits currently waiting to be processed for a given flow is compared with with a tuned threshold. If a flow queue is below the threshold, then all packets in the flow are sent to the scheduler. If a flow queue is above the threshold, then a WRED function on the shared buffer is invoked. However, such a design would, during congestion, amount to toggling between "send everything" and "send nothing". This is because during congestion the shared buffer occupancy would be high, hence the transmit fraction when WRED is invoked would be small or zero. The result would be to bypass the WRED effect on flow control, and revert to a kind of tail drop policy.

A flexible and adaptive way to support relative behaviors in DS AF has been proposed in [17]. The scheme that consists of a scheduler and an active queue management algorithm,

enables dynamic drop rate adaptation of different traffic classes, efficient and early congestion avoidance, and easy setting up of thresholds. To achieve these goals the scheme can drop already enqueued packets according to a virtual queue length that is being maintained. A virtual scheduler requires access to packets already enqueued. This feature is unfortunately only rarely given.

Another alternative is to use a scheduler and a separate RED function with the input being each flow queue rather than shared buffer occupancy of all flow queues (RIO). With this approach, the allocation example above can be solved. However, with more parameters than WRED, the configuration with RIO is even more difficult. Furthermore, RIO shows high routine buffer occupancy during congestion.

Finally, the alternative pursued in this work is to use an adaptive control algorithm that senses not just queue levels but rates of queue levels changes and per flow bit rates to converge to an optimal set of transmit fractions automatically. A description of this new active queue management system is given in the next section.

3 New Active Queue Management

The fundamental premise of the new active management scheme is that the use of more information than just shared buffer occupancy, together with control analysis should lead to improved network performance. The objectives for the new approach are:

- Bandwidth allocation should be easy to administer using only natural parameters such as assured minimum bandwidth and upper maximum limit of bandwidth.
- An aggregated flow at or below its assured rate should suffer no drops (and, in some cases, no delays) whatsoever, regardless of the offered rates of other aggregated flows such as Best Effort (BE) or AF aggregated flows.
- Any aggregated flow offering traffic above its assured rate should obtain its assured rate plus a share of any excess bandwidth.
- Buffer occupancy should not exceed a minimal level in routine operation because high buffer occupancy means high latency, unpredictable jitter and low tolerance of traffic surges. High oversubscription in itself should not cause high buffer occupancy; rather, extreme burstiness of traffic might temporarily lead to high buffer occupancy.

Let $f_{\text{i,min}}$ and $f_{\text{i,max}}$ be respectively the assured minimum rate and the upper maximum limit of the rate for a flow i, let $O_i(t)$ and $f_i(t)$ be respectively the measured current offered and current serviced rate of flow i at time t. Then, the *new active queue management scheme* is defined by an incremental transmit probability function $T_i(t)$ for each flow i:

$T_i(0) = 1$

$$T_i(t+dt) = \begin{cases} min(1, T_i(t) + w) & \text{if} \quad f_i(t) \leq f_{\text{i,min}} \\ T_i(t)(1-w) & \text{else if} \quad f_i(t) > f_{\text{i,max}} \\ min(1, T_i(t) + CB_{\text{avg}}(t)) & \text{else if} \quad B(t) = 1 \\ T_i(t)(1 - DO_i(t)) & \text{otherwise} \end{cases}$$

The transmit probability is computed periodically using the preceding transmit probability value, the current offered rate of the flow O_i, the current serviced rate of the flow $f_i = T_i O_i$, and the excess bandwidth signal $B(t)$:

$$B(t) = \begin{cases} 1 & \text{if} \quad Q(t) < q_{\text{min\_th}} \\ 0 & \text{else if} \quad Q(t) > q_{\text{max\_th}} \\ 1 & \text{else if} \quad dQ(t)/dt < d_{\text{min}} < 0 \\ 0 & \text{otherwise} \end{cases}$$

The excess bandwidth signal is a binary function. Excess bandwidth is indicated (return value 1) if the queue is either below a minimum threshold ($q_{\text{min\_th}}$) or depleted at a rate higher than $|d_{\text{min}}|$ while being below a maximum threshold. Otherwise, there is no excess bandwidth (return value 0). The exponential weighted moving average of $B(t)$ is given by $B_{\text{avg}}(t)$.

The transmit probability is drastically reduced if the actual flow rate is above the maximum $f_{\text{i,max}}$. Likewise, the transmit probability is drastically increased when the actual flow rate is below the minimum rate $f_{\text{i,min}}$. If the actual flow rate is between minimum and maximum, the excess bandwidth signal determines whether the transmit probability will be increased or reduced. The increase for the cases $f_i(t) \leq f_{\text{i,min}}$ and $f_{\text{i,min}} \leq f_i(t) < f_{\text{i,max}}$ is additive and the decrease for the cases $f_i(t) > f_{\text{i,max}}$ and $f_{\text{i,min}} \leq f_i(t) < f_{\text{i,max}}$ is multiplicative.

The algorithm thus practically consists of two parts—the more aggressive one, serving to bring the actual flow rate within the maximum and minimum rate bounds ($f_{\text{i,min}}, f_{\text{i,max}}$), and a fine-tuned control loop operating within these bounds. The aggressive part is carried out by additive increase or multiplicative decrease of constant rate, determined by a constant w.

The control loop within the flow maximum and minimum rate bounds is based on feedback of the excess bandwidth signal. The control algorithm is again additive increase and multiplicative decrease, chosen for its good stability and convergence properties [18]. The coefficients C and D determine the rate of increase and decrease. The exact values of the C and D constants depend largely on the environment and can be fine-tuned by a more detailed analysis of the control loop.

Further parameters are employed to improve the rate of convergence of the algorithm. The increase step is proportionally aligned with the exponential weighted moving average of the excess bandwidth signal $B_{avg}(t)$ in order to differentiate among cases where, in the near past, excess bandwidth either primarily *was* available, or primarily *was not*. In the negative case, the algorithm is more likely to be close to the ideal T_i value and thus the rate of increase becomes more fine-grained, whereas, in the positive case, the likelihood is higher that there is room for a more robust increase. Furthermore, the rate of decrease is proportionally aligned with the offered load of the individual flows O_i, in order to prevent flows with higher offered loads from monopolizing the available excess bandwidth.

4 Testing

The active queue management system described in Section 3 has been implemented on the IBM PowerNP 4GS3 [19] network processor. The implementation makes use of the hardware-supported flow control mechanism that is invoked when a packet is enqueued. For each packet, the transmit probability is determined from a table that is loaded into the transmit probability

memory. The key to index into the table is composed from the DSCP (DiffServ code point). Flow accounting information provided by the normal DS forwarding process in the PowerNP is used to determine individual offered loads. The PowerNP reference platform used for testing is equipped with 40 100 Mb/s and two 1G/s Ethernet ports. The maximum queue length is 256×64 KB.[1]

The function to update the transmit probability memory is triggered in fixed time intervals. An application programming interface (API) is used to communicate flow specifications to the BAT implementation in picocode. This interface is used by a standard signaling protocol (i.e. RSVP) to communicate bandwidth reservations to the network processor.

The algorithm has been tested with a variety of congestion scenarios. The results presented here are obtained when sending three flows from an IXIA traffic generator to a single 100 Mb/s output port. Each flow enters the network processor at a different input port. The flows are specified as given in Table 2.

Table 2: Flow specifications

Phase	Flow	$f_{\min}$ [Mb/s]	$f_{\max}$ [Mb/s]	O [Mb/s]	I [Mb/s]
A	1	20	20	80	20
A	2	0	30	100	30
A	3	70	100	0	0
B	1	20	20	80	20
B	2	0	30	100	10
B	3	70	100	70	70

Flow 1 has equal minimum and maximum assured rates. Flow 2 has no minimum assured rate and is limited to 30 Mb/s.[2] Flow 3 has a minimum assured rate of 70 Mb/s but no upper limit. The target port is not oversubscribed (i.e. the sum of minimum rates is less than 100 Mb/s), and during phase A not even congested as the offered load of Flow 3 is zero. During phase B, Flow 3 sends with 70 Mb/s so that Flow 2 can no longer benefit from the 20 Mb/s excess bandwidth that existed before. The ideal rate for Flow 2 would be 10 Mb/s. With this rate for Flow 2, Flows 1 and 3 would receive their assured minimum of 20 Mb/s and 70 Mb/s, respectively.

Figures 3 and 4 show offered and actual rates for Flows 1 and 2. At time $t = 20\ s$, Flow 2 starts sending 100 Mb/s. About 50 s later Flow 1 starts sending with 80 Mb/s. During $100\ s < t < 200\ s$ and $250\ s < t < 350\ s$ Flow 3 sends an additional 70 Mb/s. The periods when only two flows are sending are referred to as Phase A and the periods when three flows are sending are referred to as Phase B. No congestion occurs during Phase A and congestion does occur during Phase B. All flows are non-responsive, constant bit rate flows with a packet size of 1024 bytes.

The actual flow rates in Figures 3 and 4 indicate that the algorithm protects the minimum bandwidth assured rates of 20 Mb/s and 70 Mb/s of Flows 2 and 3, respectively. When discriminating Flow 2, the rate oscillations during Phase B are higher than for the other two flows. This is because the transmit probability function operates in the non-drastic cases $f_{2,\min} < f_2 \leq f_{2,\max}$.

[1] Testing was limited to packet processing at the egress side of the router.

[2] A specification with a minimum assured rate of 0 Mb/s can be regarded as BE specification.

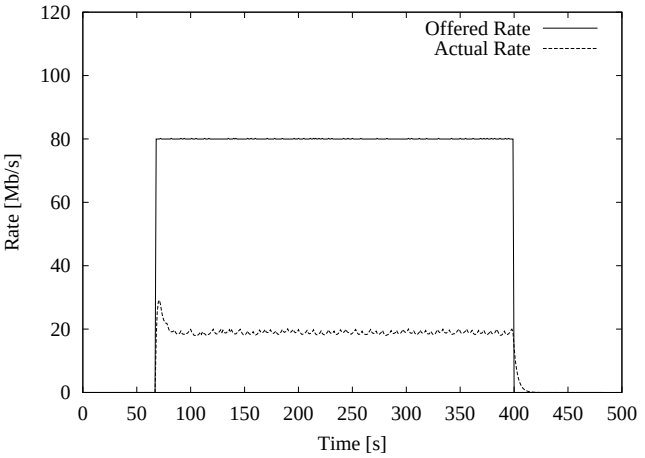

Figure 3: Offered *vs.* actual rate of Flow 1

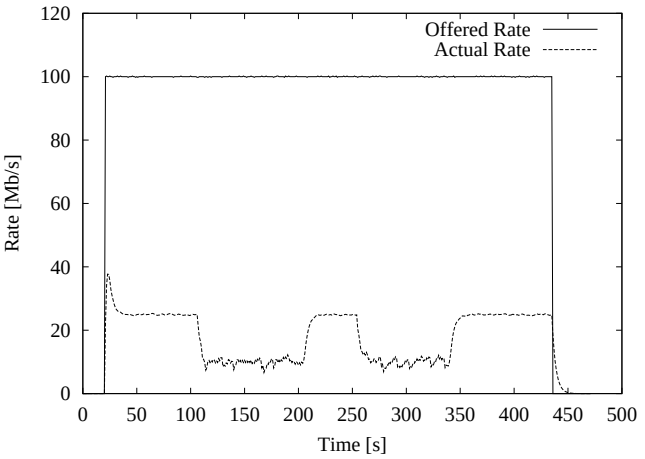

Figure 4: Offered *vs.* actual rate of Flow 2

Congestion during Phase B clearly has an effect on queue occupancy as depicted in Figure 5. The average queue level during this phase of 250% offered load is about 1% or about 160 packets. The corresponding excess bandwidth signal $B_{\text{avg}}(t)$ as defined in Section 3 is approx. 0.93 during periods of congestion (see Figure 6). This excess bandwidth signal directs the control algorithm to the ideal transmission rates.

The parameters used with the testing were set to $w = 1/8$, $C = 1/128$, $D = 1/2$, $q_{\text{min\_th}} =$ 1500 Bytes, $q_{\text{max\_th}} = 1/4$ of maximum queue length, and $d_{\text{min}} = -1/1024$.

In summary, the test results demonstrate that the algorithm fulfills the objectives set in Section 3. For management purposes, the administrators only need to set the f_{min} and f_{max} values of each flow. No flow is pushed below its assured rate and non-saturated flows share the available excess bandwidth (within the limit of the flow maximum rate bound). Finally, buffer occupancy remains below 3.5% and around 1% in average.

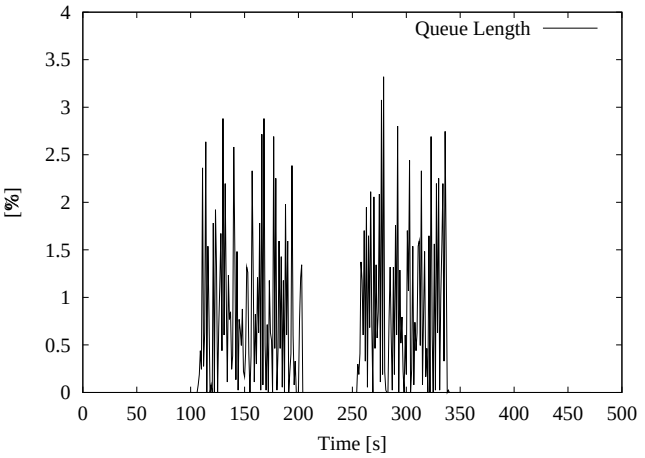

Figure 5: Queue occupancy

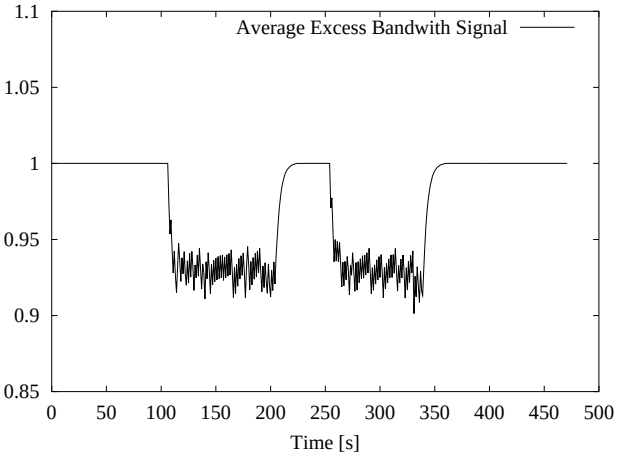

Figure 6: Excess bandwidth signal $B_{\text{avg}}(t)$

5 Future Work

The work presented here focuses on investigating a new active queue management algorithm for non-responsive traffic. An important question for a stable operation of the Internet is whether the proposed adaptive queue management technique works well with responsive flows such as TCP. In design of the technique, we have attempted to preserve features of RED which make it a desirable queue management scheme in the best-effort Internet. We discard packets randomly and maintain routinely low buffer occupancy to avoid the undesirable effects of tail-drops and TCP synchronization. Still, additional work is needed to ensure effective utilization of bandwidth with responsive flows. We intend to investigate this issue in the near future.

In the objectives for the new active queue management scheme we stated that any aggregated flow offering traffic above its assured rate should obtain its assured rate plus a share of any excess bandwidth. An important question is, according to which fairness definition, the excess bandwidth is shared (e.g. Max-Min, proportional). This issue will also be investigated in the near future.

6 Related Work

There is related work on automatic tuning of the Random Early Detection algorithm:

Floyd *et al.* [20] extended an idea by Feng *et al.* [5] to automatically tune RED parameters in order to achieve independence from the level of congestion. In the original proposal, it was shown that there is no single set of RED parameters that would work well under different congestion scenarios. An algorithm was presented that automatically adjusts $p_{\max}$ to keep the average queue length between $q_{\min}$ and $q_{\max}$. In the extended approach to automatically tune RED parameters, also $q_{\max\_th}$ and the weight for computing the exponential moving average of the current queue length are automatically tuned. Furthermore, $p_{\max}$ is adapted using additive increase and multiplicative decrease (AIMD) rather than multiplicative increase/decrease.

Another active queue management algorithm is BLUE [21]. In contrast to RED that uses the average queue occupancy BLUE uses the packet loss rate and link utilization history of queues for queue management. Only a single probability to mark (or drop) packets is maintained. BLUE increments the marking probability if the queue is continually dropping packets due to buffer overflows and decreases the probability during empty queue periods or when the link is idle.

Stochastic fair BLUE (SFB) is an extension that addresses scalability and fairness amongst flows in large aggregates using the BLUE algorithm. Combined with FIFO queuing and levels of independent hash functions non-responsive flows can be efficiently rate-limited to a fair share of the link bandwidth. The drawback is that this advantages are coupled with additional expenses in implementation complexity.

A number of papers deal with either analyzing RED by using control theory [22, 7, 23] or designing a new active queue management systems based on control theory [24]. These approaches as well as the adaptive RED described above are not really designed for implementing AF drop precedences, but are alternatives to the original Random Early Detection algorithm.

7 Conclusions

We have proposed the use of an adaptive queue management technique to actively adjust router buffers in networks implementing service level specifications. To test the value of this approach, we have implemented the Assured Forwarding per-hop behavior of Differentiated Services on a network processor and taken measurements under congestion conditions. These measurements show that such an approach can converge quickly for a variety of offered traffic rates, while consistently assuring service level specifications are met. Excess bandwidth is effectively utilized by the traffic streams. These measurements also show routinely low buffer occupancy for a variety of traffic patterns, indicating the technique will offer good throughput and latency under bursty traffic conditions.

We believe this adaptive queue management technique may offer a promising approach to queue management in routers. Such an approach can potentially reduce the cost and complexity of implementing service level specifications in Internetworks. This approach does not require a network administrator to tune parameters for each router queue depending on historic traffic patterns, but instead actively reacts to changes in traffic rates.

References

[1] S. Blake, D. Blake, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An Architecture for Differentiated Services," IETF, RFC2475, Dec. 1998.

[2] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski, "Assured Forwarding PHB Group," IETF, RFC2597, June 1999.

[3] J. Heinanen and R. Guerin, "A Single Rate Three Color Marker," IETF, RFC2697, Sept. 1999.

[4] J. Heinanen and R. Guerin, "A Two Rate Three Color Marker," IETF, RFC2698, Sept. 1999.

[5] W. Feng, D. Kandlur, D. Saha, and K. Shin, "A Self-Configuring RED Gateway," in *Proceedings of IEEE INFOCOM*, Mar. 1999, pp. 1320–1328.

[6] M. May, J. Bolot, C. Diot, and B. Lyles, "Reasons Not to Deploy RED," in *Proceedings of the Workshop on Quality of Service (IWQoS)*, June 1999, pp. 260–264.

[7] V. Misra, W. Gong, and D. F. Towsley, "Fluid-Based Analysis of a Network of AQM Routers Supporting TCP Flows with an Application to RED," in *Proceedings of ACM SIGCOM*, 2000, pp. 151–160.

[8] V. Jacobson, K. Nichols, and K. Poduri, "An Expedited Forwarding PHB," IETF, RFC2598, June 1999.

[9] B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, and L. Zhang, "Recommendations on Queue Management and Congestion Avoidance in the Internet," IETF, RFC2309, Apr. 1998.

[10] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," *ACM Transactions on Networking*, pp. 397–413, Aug. 1993.

[11] M. May, Th. Bonald, and J. Bolot, "Analytic Evaluation of RED Performance," in *Proceedings of IEEE INFOCOM*, Mar. 2000, pp. 1415–1424.

[12] M. Christiansen, K. Jeffay, D. Ott, and F. Smith, "Tuning RED for Web Traffic," in *Proceedings of ACM SIGCOM*, 2000, pp. 139–150.

[13] D. Clark and W. Fang, "Expicit Allocation of Best Effort Packet Delivery Service," *ACM Trans. on Networking, Vol. 6, No. 4*, pp. 362–373, Aug. 1998.

[14] U. Bodin, O. Schelen, and S. Pink, "Load-Tolerant Differentiation with Active Queue Management," *ACM Computer Communication Review*, vol. 30, no. 3, pp. 362–373, July 2000.

[15] D. Lin and R. Morris, "Dynamics of Random Early Detection," in *Proceedings of ACM SIGCOM*, Oct. 1997, pp. 127–137.

[16] R. Makkar, I. Lamadaris, J. H. Salim, N. Seddigh, B. Nandy, and J. Babiarz, "Empirical Study of Buffer Management Scheme for DiffServ Assured Forwarding PHB," in *International Conference on Computer Communications and Networks (ICCCN)*, 2000.

[17] R. Pletka, P. Droz, and B. Stiller, "A Buffer Management Scheme for Bandwidth and Delay Differentiation Using a Virtual Scheduler," in *Proceedings of International Conference on Networking (ICN)*, July 2000, pp. 218–234.

[18] D. Chiu and R. Jain, "Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks," *Journal of Computer Networks and ISDN*, vol. 17, no. 1, pp. 1–14, June 1989.

[19] "IBM PowerNP NP4GS3," http://www.chips.ibm.com/products/wired/ communications/network_processors.html.

[20] S. Floyd, R. Gummadi, and S. Shenker, "Adaptive RED: An Algorithm for Increasing the Robustness of RED," Technical Report, to appear, 2001.

[21] W. Feng, D. Kandlur, D. Saha, and K. Shin, "BLUE: A New Class of Active Queue Management Algorithms," Tech. Rep. CSE-TR-387-99, University of Michigan, 15, 1999.

[22] V. Firoiu and M. Borden, "A Study of Active Queue Management for Congestion Control," in *Proceedings of IEEE INFOCOM*, 2000, pp. 1435–1444.

[23] C. Hollot, V. Misra, D. Towsley, and W. Gong, "A Control Theoretic Analysis of RED," in *Proceedings of IEEE INFOCOM*, Apr. 2001.

[24] C. Hollot, V. Misra, D. Towsley, and W. Gong, "On Designing Improved Controllers for AQM Routers Supporting TCP Flows," in *Proceedings of IEEE INFOCOM*, Apr. 2001.