Active Queue Control Scheme for Achieving Approximately Fair Bandwidth Allocation

Takashi Miyamura[†*] Takashi Kurimoto[†] Kenji Nakagawa[††]
Prasad Dhananjaya[††] Michihiro Aoki[†] Naoaki Yamanaka[†]

[†]NTT Network Service Systems Laboratories, Musashino-shi, Tokyo, 180-8585 Japan
[††] Nagaoka University of Technology, Nagaoka-shi, Niigata, 940-2188 Japan

Abstract— **We propose a buffer management mechanism, called V-WFQ (Virtual Weighted Fair Queueing), for achieving approximately fair bandwidth allocation with a small hardware amount in high-speed networks. The basic process for allocating bandwidth fairly uses selective packet dropping to compare the measured input rate of the flow with an estimated fair bandwidth share. Though V-WFQ is a hardware-efficient FIFO-based algorithm, it can achieve almost ideal fairness in bandwidth allocation. Simulation results show that V-WFQ achieves a good balance between fairness and link utilization under various simulation conditions.**

I. Introduction

In today's Internet, network resources are shared among many flows under best-effort service conditions. If there are malicious users in a network, they can get more bandwidth than well-behaved users can. It is thus important in the Internet to protect well-behaved flows from ill-behaved ones. One solution for isolating ill-behaved flows is to allocate a fair bandwidth to each flow.

Two types of algorithms can be used to achieve fair bandwidth allocation: scheduling based algorithms and preferential dropping based algorithms.

Scheduling based algorithms (e.g., Weighted Fair Queueing (WFQ)[3] and its variants [4], [5], [6]) are known to be ideal mechanisms for providing fair bandwidth allocation and QoS guarantees. However, these approaches must maintain separate queues for each flow and maintain the per-flow state, so they do not use hardware-efficient mechanisms. In particular, WFQ has a computational complexity of $O(\log(n))$, where n is the number of flows currently queued at the router. WFQ is hard to implement in high-speed backbone routers whose trunks have large numbers of flows.

Core-Stateless Fair Queueing (CSFQ), proposed by Stoica et al. [8], is well known as one of the preferential dropping based algorithms. The CSFQ mechanism can achieve almost ideal fairness without using the per-flow state in the core routers. To avoid maintaining the per-flow state at each router, they use a distributed algorithm in which only edge routers maintain per-flow state, while core (non-edge) routers do not. An edge router estimates the arrival rate of each flow and this information is carried via a label in each packet's header and a core router utilizes this information to decide whether to discard or queue arriving packets. Thus, the core routers of CSFQ are simple and easy to implement compared with WFQ. However, the architecture of the edge routers is still complicated. Though CSFQ can be extended easily to support flows having different weights, their algorithm cannot accommodate situations where the relative weights of flows differ from router to router.

Recently, Cao et al. [9] have proposed Rainbow Fair Queueing (RFQ), which can also achieve approximately fair sharing as well as CSFQ. Their approach is to divide each flow into a set of layers, based on rate. The packets in a flow are marked at an edge router with a layer label rather than the explicit rate of their flows. While the fair share calculation in CSFQ requires exponential averaging, the core routers in RFQ only need to perform threshold-based dropping, so these routers are simple and amenable to hardware implementation. However, the edge routers in their approach remain complicated. It is important to note that both CSFQ and RFQ require extensions of the IP packet's header in order to be applied to today's Internet.

In this paper, we propose an active queue control scheme, called V-WFQ (Virtual Weighted Fair Queueing), for achieving almost fair bandwidth allocation and minimum bandwidth guarantees in high-speed networks. The basic process for allocating bandwidth fairly uses preferential packet dropping to compare the measured input rate of the flow with an estimated fair bandwidth share. Though V-WFQ is a hardware-efficient FIFO-based algorithm, it can achieve almost ideal fair bandwidth allocation. Unlike CSFQ and RFQ, V-WFQ does not require any header extensions of IP packets and does not need to distinguish routers in the network into core and edge routers, so it can be easily applied to today's Internet backbone networks.

II. Proposed Algorithm

A. Overview of V-WFQ

The buffer architecture of a V-WFQ router is shown in Fig. 1. The proposed mechanism: (i) estimates the input rates of the flows at each router in the network, (2) calculates the allocated bandwidth of each flow at each router, and (3) uses FIFO buffering with preferential dropping.

* email: miyamura.takashi@lab.ntt.co.jp, tel: +81-422-59-3527, fax :+81-422-59-4549
N. Yamanaka is now with NTT Network Innovation Laboratories, 3-9-11 Midori-cho, Musashino-shi, Tokyo, 180-8585 Japan

To avoid the need for per-flow buffer management and scheduling, we use a FIFO buffer with preferential packet discarding on arrival. Our proposed mechanism enables efficient implementation of high-speed core routers as well as improved in fairness.

A V-WFQ router measures the input rate of flows. When each packet arrives, the V-WFQ router calculates the fair bandwidth for the flow based on the degree of output link congestion. By comparing the estimated input rate with the calculated allocated bandwidth, the router decides whether or not to discard the packet. If it is not discarded, the packet is enqueued in the buffer of the output link.

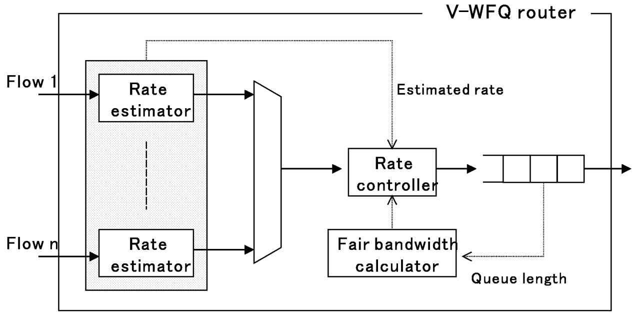

Fig. 1. Buffer architecture of V-WFQ router.

B. Algorithm of V-WFQ

1) Flow Rate Estimator: To estimate the flow arrival rate, we use the simple Jumping Window method. This requires only two states for each flow, without any complicated computation to estimate the arrival rate. Let R_i and T_w be the estimated rate of flow i and the window size for the Jumping Window method, respectively. Let Ct_i be the cumulative packet length of flow i. At the beginning of the window, Ct_i is set to 0. If a packet of flow i arrives, Ct_i is renewed as follows:

$$Ct_{i+1} = Ct_i + l_i,$$

where l_i is the packet length. At the end of the window, the estimated rate R_i is calculated as

$$R_i = \frac{Ct_i}{T_w}.$$

Ct_i is then reset to 0.

2) Packet Dropping Decision: The V-WFQ router estimates the degree of output link congestion from the variation in queue length. The following terms are used in calculating the allocated rate:

- N: number of sources,
- $weight_i$: weight of flow i,
- R_i: estimated arrival rate of flow i,
- AR: fair sharing,
- AR_i: bandwidth allocated to flow i,
- AR_{max}: maximum allocated bandwidth,

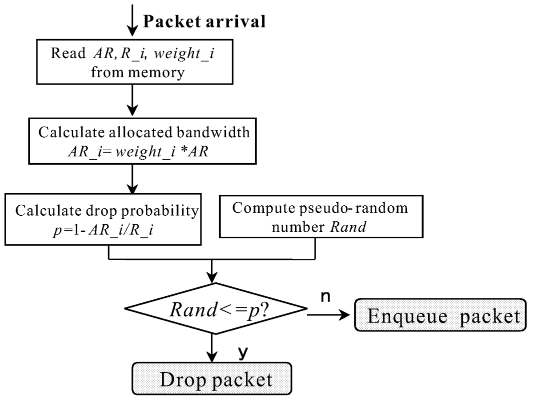
Packet arrival
Read AR, R_i, weight_i from memory
Calculate allocated bandwidth AR_i= weight_i *AR
Calculate drop probability p=1- AR_i/R_i
Compute pseudo-random number Rand
Rand<=p?
n
Enqueue packet
y
Drop packet

Fig. 2. The V-WFQ algorithm.

```
Every time interval t_update:
If (q_average> q_max){
        AR = MGR;
}
Else if(q_average<q_min){
        AR = AR+0.1AR;
}
Else{
        AR = AR/2;
}
 AR=min{AR, AR_max};
/* Updating q_average*/
 q_average= α*q_now+(1 − α )*q_average;


On packet arrival (flowID=i):
/* Calculating the bandwidth allocated AR_i */
 AR_i= weight_i * AR ;
/* Calculating dropping probability p_drop*/
If(R_i ≦ AR_i){
        p_drop= 0;
}
Else if{
        p_drop= (R_i−AR_i)/R_i;
}

If(p_drop≦ Rand){
        drop packet;
}
Enqueue packet;
```

Fig. 3. Algorithm of packet dropping decision.

- t_{update}: time interval of updating AR,
- C: capacity of output link.

The pseudocode of the V-WFQ router algorithm is shown in Figure 3. The router updates the fair sharing AR every time interval t_{update} according to the current congestion status of the output link. When the average queue length $q_{average}$ reaches the threshold q_{max}, it is assumed that the output link is congested and AR should be decreased. If $q_{average}$ is below the threshold q_{min}, it is assumed that the output link is underutilized and AR should be increased to raise the link utilization.

When a packet arrives, the router calculates the allocated bandwidth AR_i of the flow.

$$AR_i = weight_i \cdot AR$$

Then, the V-WFQ router decides whether or not to discard the arriving packet. We use the random dropping approach. The probability of the arriving packet being dropped p_{drop} is given by

$$p_{drop} = \begin{cases} \frac{R_i - AR_i}{R_i} & if \;\; R_i \geq AR_i \\ 0 & if \;\; R_i < AR_i \end{cases}$$

If one ill-behaved flow keeps on sending packets at a higher rate than its fair share, the V-WFQ router discards packets of this flow probabilistically and provides only the bandwidth corresponding to the calculated fair share for the ill-behaved flow. This mechanism can ensure fair bandwidth allocations for all flows sharing the link resource. Furthermore, by dropping packets long before the buffer becomes full, it can avoid the congestion collapse and provide low latency.

3) Hardware Implementation: Here we compare V-WFQ with WFQ in terms of hardware implementation cost. In VLSI, this depends directly on the required chip area, which in turn is dominated by memory [11].

Figure 4 compares the amount of memory required by V-WFQ and WFQ for various numbers of flows. The amount of memory required by V-WFQ is much less than that required by WFQ. For example, V-WFQ requires about only 11 Kbytes at 4000 flows, while WFQ requires about 1560 Kbytes. Thus V-WFQ is a more hardware-efficient algorithm than WFQ.

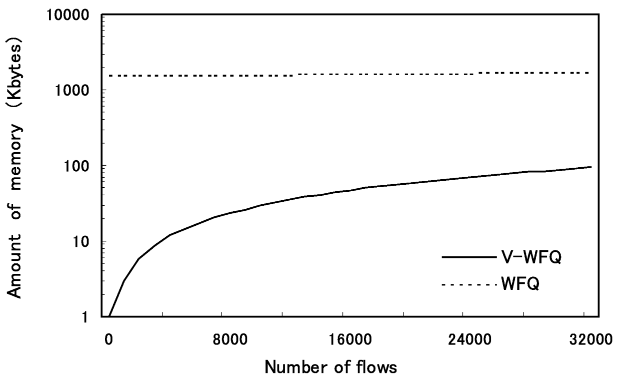

Fig. 4. Comparison of V-WFQ with WFQ in terms of required memory.

III. Performance Evaluation

A. Simulation Configuration

To investigate the performance of V-WFQ when there is single congested link in the network, we considered the network configuration shown in Figure 5. The congested link lay between two routers. The link, which had a capacity of 10 Mbps, was shared by m TCP flows and n UDP flows. An end host was connected to the router using a 10 Mbps link and one end host transmitted packets to the other end host persistently. We compared the performance of V-WFQ with Random Early Discard (RED)[12], Flow Random Early Discard (FRED)[6], Deficit Round Robin (DRR)[13], and DropTail.

RED monitors the average queue length and starts dropping arriving packets probabilistically when the average queue length exceeds certain threshold. To start to drop packets before the buffer becomes full, RED gives an early congestion indication to each flow and avoids congestion collapse. However, RED cannot provide fairness among competing flows. FRED improves the fairness of the bandwidth allocation in RED by maintaining the per-flow state. FRED drops packets from flows that have had many packets dropped in the past or flows that have queues larger than the average queue length.

DRR is a well-known implementation of WFQ that has computation complexity of $O(1)$. The buffer management scheme of DRR requires a sophisticated per-flow queueing mechanism, while the other algorithms used in our simulations only require simple FIFO buffer mechanisms. DRR can achieve a higher degree of fairness than the other algorithms, so we used DRR as the benchmark for fair bandwidth allocations.

All simulations were performed in ns-2[14]. The simulation code for DRR, RED, and FRED algorithms is available with the ns-2 package. In all simulations, unless otherwise stated, we used the following assumptions and parameters.

- All TCP sources were persistent TCPs and started at the same time. For a TCP flow, maximum segment size (MSS) was 1000 bytes and the version of TCP was Reno. The TCP hosts transmitted infinite-size files by FTP to the corresponding end hosts.
- The UDP hosts sent packets at a constant bit rate (CBR) of k Kbps, where k is a variable, and started at the same time. The packet size of the UDP flows was set to 500 bytes.
- The output buffer size was 128 KB.
- All flows had the same weight values.

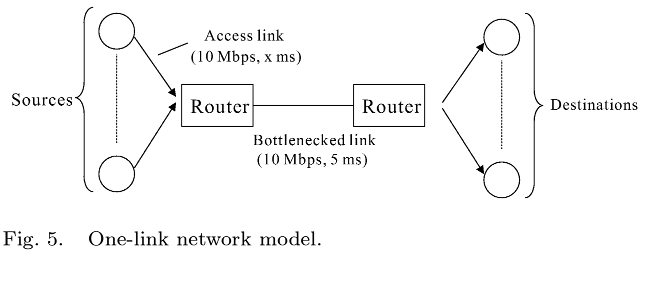

Fig. 5. One-link network model.

B. Performance Evaluation Index

We evaluated the performance of our algorithms in terms of fairness and efficiency. Let N be the number of TCP sources, C be the link capacity of the bottlenecked link, x_i be the measured throughput of the i-th TCP source, and z_i be the ideal throughput of the i-th

TCP source. We define efficiency as

$$efficiency = \frac{\sum_{i=1}^{N} x_i}{C}.$$

We measured the fairness of algorithms in terms of the *fairness index* (FI)[15], which is defined

$$fairness\ index = \frac{\left(\sum_{i=1}^{N} \frac{z_i}{x_i}\right)^2}{N \times \sum_{i=1} \left(\frac{z_i}{x_i}\right)^2}.$$

C. Simulation Results

1) Fair Allocation for UDP flows: In the first experiment, we evaluated the performance of V-WFQ when all hosts transmitted infinite data using UDP. Each of 16 UDP sources sent packets at $i \times \frac{10}{16}$ Kbps, where i $(1, \ldots, 16)$ is the flow ID. During 30 seconds of simulation, each UDP source had infinite data to transmit so the backbone link was always severely congested. Under max-min fairness [15], each flow should have achieved an average throughput of 626 Kbps. Figure 6 shows the average throughput achieved by 16 UDP sources sharing the bottlenecked link which was configured using V-WFQ, DRR, FRED, RED or DropTail and Table I shows the fairness and efficiency achieved by each algorithm used in this simulation. These results show that RED and DropTail completely failed to allocate fair bandwidth to each flow during congestion while DRR achieved almost ideal fairness of bandwidth sharing among competing flows. V-WFQ showed the same performance as DRR and performed much better than FRED.

TABLE I
Comparison in fairness index and efficiency.

	Efficiency	**Fairness**
V-WFQ	0.999	0.999
DRR	0.999	0.999
FRED	0.999	0.859
RED	0.999	0.766
DropTail	0.999	0.768

2) Single Misbehaving Flow: In the second experiment, we examined the protection of well-behaved flows against single misbehaving one. In this simulation, we assumed that 15 TCP flows and 1 UDP flow shared a 10-Mbps bottlenecked link. The UDP source sent packets at the constant bit rate of 10 Mbps during the 30 seconds of simulation. Figure 7 shows the normalized throughput, which is defined as the throughput/fair-share ratio. Note that the normalized throughput of each flow should equal 1.0 in the optimal situation. Table II shows the fairness and efficiency achieved by each algorithm used in this simulation. These results show that RED and DropTail failed to provide fair sharing to the well-behaved TCP flows while DRR and V-WFQ protected TCP flows against the over-transmitting UDP flow and achieved almost ideal fairness.

TABLE II
Fairness index and efficiency for a configuration with a single misbehaving UDP flow and 15 TCP flows.

	Efficiency	**Fairness**
V-WFQ	0.995	0.926
DRR	1.00	0.973
FRED	1.00	0.735
RED	1.00	0.0778
DropTail	1.00	0.0778

3) Flows with Different Weights: Next, we investigated the performance of V-WFQ for flows with different weights. We compared the performance of V-WFQ with DRR in terms of throughput achieved for each flow and queue dynamics. Here, the number of source-destination pairs was 5. The weight $weight_i$ is given by

$$weight_1 : weight_2 : \cdots : weight_5 = 1 : 2 : \cdots : 5.$$

In this simulation, the sources were all TCP sources. During the 30-s simulation, each TCP source had infinite data to transmit so the backbone link was always severely congested. Each flow should have been allocated bandwidth in proportion to its weight.

Figure 8 shows the average throughput achieved by 5 TCP sources over a 30-s interval. RED and FIFO failed to ensure fairness in throughputs, while V-WFQ and DRR achieved almost ideal fairness.

Figure 9 compares the queue dynamics for V-WFQ and DRR. The queue length of DRR is the sum of the separate queues. The queue in V-WFQ fluctuated greatly, while that in DRR tended to stabilize at higher buffer occupancy. V-WFQ estimated the fair allocation of each flow from the queue length, hence the rate control of a V-WFQ router did not work as long as the queue length did not become longer. The average queue length in V-WFQ was much shorter than that in DRR. Because V-WFQ, like RED, starts to drop packets before the buffer becomes full, it gives an early congestion indication to each flow and hence can provide much lower latency than DRR.

IV. Concluding Remarks

We have proposed a FIFO-based mechanism called V-WFQ for achieving fairness in throughput and investigated its performance in fairness and link utilization through extensive simulations, in which we compared the performance of V-WFQ with DRR and other algorithms. DRR requires a sophisticated per-flow queueing mechanism, while V-WFQ only requires simple FIFO buffer mechanisms. DRR achieved a higher degree of fairness than the other algorithms, so we used DRR as the benchmark for fair bandwidth allocations. The simulation results shows that V-WFQ achieved almost ideal fairness in various simulation conditions. In particular, we should note that it showed almost the same performance as DRR and achieved a much higher degree of fairness than FRED, which is a preferential dropping algorithm like V-WFQ.

It is also possible to use the V-WFQ architecture to provide minimum bandwidth guarantees for flows. As one

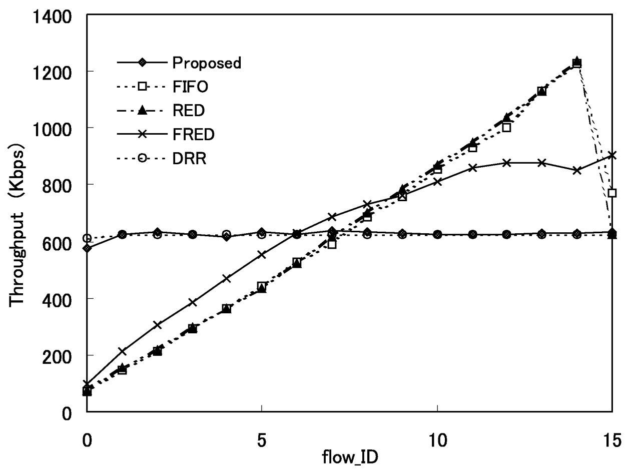

Fig. 6. Throughput for 16 UDP flows sharing a 10-Mbps bottlenecked link. Each UDP flow i sends packets at i times its fair allocation (626 Kbps).

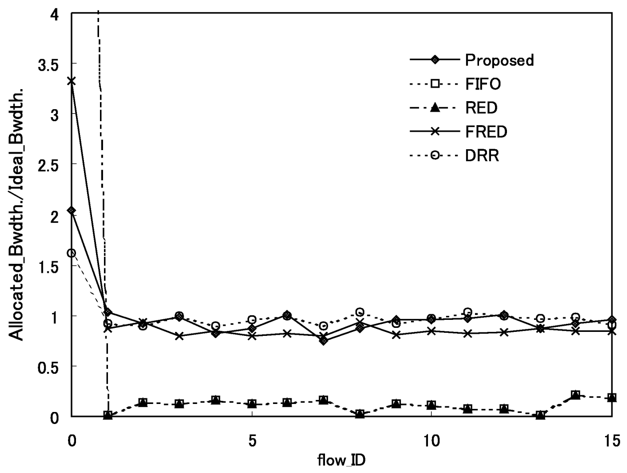

Fig. 7. Throughput for 1 UDP flow and 15 TCP flows sharing a 10-Mbps bottlenecked link. The UDP flow i sends packets at 10 Mbps.

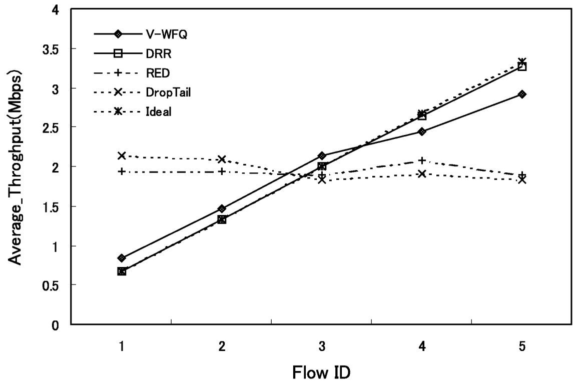

Fig. 8. Average throughput achieved by each of 5 TCP flows with different weights.

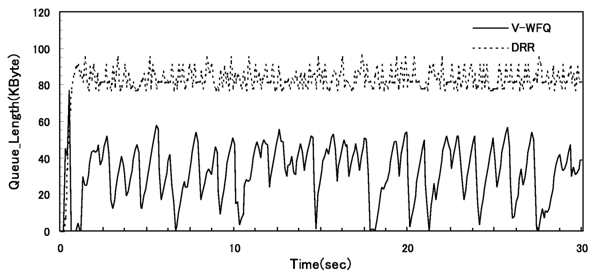

Fig. 9. Queue dynamics in V-WFQ and DRR.

example of applications, V-WFQ can be used for an IP VPN backbone network to provide minimum bandwidth guarantees for each VPN.

References

[1] T. Miyamura, T. Kurimoto and K. Nakagawa *et.al.*, "A New Queueing Mechanism for Achieving Fair Bandwidth Allocation in High-speed Networks", *Proc. of APCC 2001*, **E84-B**, 473-476, 2001.

[2] N. Yamanaka , T. Kurimoto, T. Miyamura, Michihiko Aoki,"MSN Type-X: Next Generation Internet Backbone Switch / Router Architecture", *Proc. of IEEE ICC 2002*, in press.

[3] A. Parekh, "Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks,"*Massachusetts Institute of Technology*, Feb. 1992.

[4] S. J. Golestani,"A Self-Clocked Fair Queueing Scheme for Broadband Applications", *Proc. of IEEE INFOCOMM*, **2**, 636-645 1994.

[5] S. Suri and G. Varghese, "Leap Forward Virtual Clock: A New Fair Queueing Scheme with Guaranteed Delays and Throughput Fairness", *Proc. of IEEE INFOCOM*, **3**, 557-565 1997.

[6] M. Schreedhar and George Varghese, "Efficient Fair Queueing using Deficit Round Robin," *Proc. of SIGCOMM'95*, Sept. 1995.

[7] M. Katevenis, S. Stefanos and C. Courcoubetis, "Weighted Round-Robin Cell Multiplexing in a General-Purpose ATM Switch Chip", *IEEE J. Select. Areas Commun.*, **9**, 1265-1279 1991.

[8] I. Stoica, S. Shenker and H. Zhang, "*Core*-Stateless Fair Queueing: Achieving Approximately Fair Bandwidth Allocations in High-speed Networks", *Proc. of ACM SIGCOMM'98*, Sept. 1998.

[9] Z. Cao, Z. Wang and E. Zegura, "Rainbow Fair Queueing: Fair Bandwidth Sharing Without Per-Flow State", *Proc. of IEEE IMFOCOMM 2000*, 2000.

[10] B. Braden *et al.*, "Recommendations on Queue and Congestion Avoidance in the Internet," *IETF* **RFC2309**, April 1998.

[11] M. Grossglauser and S. Keshav, "On CBR Service", *Proc. of IEEE INFOCOM*, 129-137 1996.

[12] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," *IEEE/ACM Transactions on Networking*, **1(4)**, 397-413 1993.

[13] D. Lin and R. Morris, "Dynamics of Random Early Detection," *Proc. of SIGCOMM'97*, Oct. 1997.

[14] UCB/LBNL/VINT, "Network Simulator - NS (Version 2)," `http://www-mash.cs.berkley.edu/ns/`.

[15] R. Jain *et al.*, "Throughput Fairness Index: An Explanation," *ATM Forum* **99-0045**, 1999.

[16] B. Vandalore, R. Jain, R. Goyal and S. Fahmy, "Design and Analysis of Queue Control Functions for Explicit Rate Switch Schemes", *Proc. of IEEE ICCCN*, 780-786 1998.