In IEEE HotInterconnects 2002

Feedback Output Queuing: A Novel Architecture for Efficient Switching Systems

Victor Firoiu Xiaohui Zhang Emre Gündüzhan
{vfiroiu,xiaohui,egunduzh}@nortelnetworks.com
Advanced Technology
Nortel Networks
600 Technology Park
Billerica, MA 01821 USA

Abstract

We consider the problem of providing QoS guarantees in a high-speed packet switch. As basic requirements, the switch should be scalable to high speeds per port, a large number of ports and a large number of traffic flows with independent QoS guarantees. Existing scalable solutions are based on Virtual Output Queueing, which is computationally complex when required to provide QoS guarantees for a large number of flows.

We present a novel architecture for packet switching that provides support for such QoS guarantees. A cost-effective fabric with small external speedup is combined with a feedback mechanism that enables the fabric to be virtually lossless, thus avoiding packet drops indiscriminate of QoS flows. Through analysis and simulation, we show that this architecture provides accurate QoS support, has low computational complexity and is scalable to very high port speeds.

1 Introduction

High speed communication between businesses has been a large share of telecommunications market in recent years. This communication needs to be of high quality, secure and reliable. Traditionally, these services were provided using ATM and Frame Relay technologies, but at a premium cost. Recent advances in traffic engineering and the advent of Voice over IP technologies provide an opportunity to carry all enterprise traffic (voice, streaming and non-real-time data) at a lower cost. Virtual Private Networks (VPNs) [5] and Virtual Private LAN Services (VPLS) [14] are two examples of such network services. A main requirement for such services is to provide Quality of Service (QoS) guarantees. Interactive media such as VoIP needs low delay and low loss, other traffic needs minimum bandwidth guarantee.

In this paper we consider the problem of providing such guarantees in a high-speed, cost-effective switch at the interface (edge) between enterprise and service provider networks. At a minimum, the switch is required to provide three types of service: Premium, Assured and Best Effort [1],[7]. Premium service provides low loss and small delay for a flow sending within a pre-determined rate limit (anything above the limit is discarded). Assured service guarantees delivery for traffic within a limit, but allows and forwards extra traffic within a higher limit if transmit opportunities are available.

A provider edge switch is required to differentiate between traffic from different customers (here called flows) and provide separate guarantees to each flow. A requirement is to support a large number of such flow guarantees (100s,1000s) per port. Traffic from one customer (flow) can enter through one or multiple ingress ports and exit through one or multiple ports. We consider the problem of providing 1-to-1 and N-to-1 services (or “Pipe” and “Funnel scope” as defined in [6]), as 1-to-N and N-to-N can be provided as combinations of services of the first two kinds. In the case of Assured N-to-1 service, it is also desirable to provide a fair distribution of service among the N components of the flow.

Current state-of-the-art switch architectures are based on Virtual Output Queuing (VOQ) [2], [11], requiring a fabric speedup $s \geq 2$ and a matching algorithm to find which packets will be sent into the fabric at each fabric cycle. Some of these algorithms can also support QoS guarantees. A major problem is that the matching algorithms have high complexity $O(M^2N^2)$ (where N is the number of ports and M is the number of flows with independent QoS guarantees per port), are run at each fabric cycle, and all VOQs at all input lines in the system need to participate in a centralized algorithm. Recent proposals [9] decrease the time interval between two runs of the matching algorithm, but with a tradeoff in increased burstiness and additional scheduling algorithms for mitigating unbounded delays. Moreover, the service presented in [9] is of type Premium 1-to-1. To our knowledge, no architecture can provide Assured N-to-1 service.

To provide a low-complexity switch architecture that fulfills the above requirements, we observe that the main cause for high complexity in current architecture is the necessity of addressing congestion at an output line. Short term congestion can be absorbed by buffers, whereas long term congestion results in packet loss. We also observe that many measurement studies (for example [10]) have shown that traffic in the Internet is dominated (about 90%) by the TCP protocol. A salient feature of TCP is that packet transmission is controlled by a congestion avoidance algorithm [8], [13]. As an effect, the average sending rate of a TCP flow is a decreasing function of drop probability and of round trip time (see [12] for a quantitative evaluation of this function). In practice, TCP flows have a stable (long-term) operation at $0..0.1$ drop probability, and very rarely operate above 0.2 [12]. Heavy long-term congestion that results in drop probability above 0.2 can be produced by non-TCP (and more general, non-congestion-controlled) traffic such as multimedia traffic over UDP.

Our proposed architecture named "Feedback Output Queuing" (FOQ) exploits this observation by efficiently supporting fast fabrics with relatively slow output memory interfaces and hence a small effective speedup. For example, a speedup of 1.25 at the fabric-to-line interface is sufficient to maintain an output drop probability up to 0.2 for a traffic fully utilizing this interface. For higher level of long-term congestion (for example drop probability above 0.2), the FOQ architecture uses a feedback mechanism for reducing traffic volume before it enters the switch fabric. Short-term congestion is mitigated by switch buffering. This FOQ mechanism provides support for the Assured service, 1-to-1 and N-to-1 scope.

Premium traffic, being policed at its guaranteed rate at the ingress, and given that rate guarantees are ensured to be within switch capacity by some admission control procedure, cannot create congestion in the absence of other type of traffic. Thus, Premium service can be provided through a simple priority scheduling in OUT ports and fabric, bypassing the FOQ mechanism.

In the following we show through analysis and simulation studies that the proposed FOQ architecture can alleviate congestion at the output lines of an OQ switch with slow output memory interface, and thus provide deterministic QoS guarantees.

2 Feedback Output Queuing architecture

We consider a switch as in Figure 1 with a fabric having internal speedup of N, an internal buffer capability and one or a very small number of queues per port. [1]

[1]This fabric has a cost-effective implementation with shared memory technology. The case of zero/small memory fabric with no/small internal speedup is a separate problem, and we report our study elsewhere.

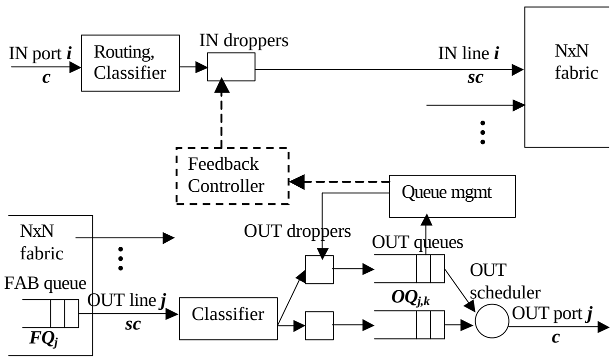

Figure 1. Detailed FOQ switch architecture

Packets enter through a set of N input ports of speed c. As a packet is received at port i, a destination port j is determined by a routing module, its QoS flow k is determined by a classifier and an IN dropper determines if the packet is discarded. If not discarded, the packet is transmitted to the fabric through a line of speed sc. We assume a fabric with internal speed of Nsc, i.e., at each fabric cycle one packet from each IN line can be moved to an OUT line while sustaining speeds of sc from all IN lines. Multiple (up to N) packets can be received at an OUT line in one cycle, and in that case the packets are placed in a fabric queue FQ_j corresponding to the destination line j.

Packets are forwarded by the OUT line j at speed sc, separated into OUT queues $\{OQ_{j,k}\}_k$ based on their QoS flow, and scheduled for transmission to OUT port j of speed c. The OUT scheduling implements various QoS guarantees such as priority, minimum rate guarantee, maximum rate limit, maximum delay guarantee. This OUT scheduling results in a certain service rate (in general variable in time) for each OUT queue.

If traffic to $OQ_{j,k}$ has a rate higher than the current service rate of flow k, packets accumulate in this queue and some of them may be dropped by a queue management mechanism such as Drop tail or RED (see [3] for details). If the traffic to all queues at OUT line j amounts to an aggregate rate above sc, then packets accummulate at the fabric queue FQ_j. If this situation persists, FQ_j fills and packets get dropped in the fabric. In this case, QoS guarantees for some flow k may be violated since fabric drops do not discriminate between different flows.

We define the *relative congestion* at a queue: $C = 1 - r_O/r_I$ where r_I and r_O are traffic rates input to and output from the queue respectively. It is easy to see that, as long as the traffic coming out of OUT line j is such that the relative congestion $C_{j,k}$ at each queue $\{OQ_{j,k}\}_k$ is below a threshold $d_{max} < 1 - 1/s$, and the OUT port j is utilized at its full capacity c, then the traffic throughput at the interface of fabric to OUT line j is below sc, and thus there is no congestion at that interface and no fabric drop.

In the FOQ architecture, a feedback mechanism controls the relative congestion at each OUT queue to be below a certain threshold in the following way. The relative congestion at each OUT queue is measured over an interval T

$$RelCong(T) = 1 - OutPkts(T)/InPkts(T)$$

A control algorithm computes a drop indication based on the last sample of relative congestion and an internal state, and transmits it to all IN modules. There, packets of the indicated flow are randomly dropped with a probability that is a function of the drop indication. By keeping the traffic below a congestion threshold, the fabric congestion and drop are avoided. Thus, packets are dropped only from those flows that create congestion, and the QoS guarantees are provided to all flows as configured.

An implementation issue is the data rate of feedback transmission. Considering M flows at each of the N OUT ports and that the drop information is coded in F bits, the total feedback data rate is MNF/T. For example, for $M = 1000$, $N = 32$, $F = 8$, $T = 1ms$, the feedback data rate for the entire switch is $256Mb/s$. It is possible to reduce this rate by reducing the precision of the feedback data, and thus its encoding. In an extreme case, the feedback has three values: increase, decrease or keep same drop level. All IN modules use this indication in conjunction with a pre-defined table of drop levels. We call this the "Gear-Box algorithm" (GB), model it in Section 3 and show its performance in Section 4. This data rate can further be reduced substantially if increase/decrease indications are only sent when needed.

Using similar arguments, we find the computational complexity of FOQ to be $O(NM)$ for the entire switch and run at every FOQ cycle (typically $1ms$). Note that this is much lower complexity than VOQ with matching, $O(M^2N^2)$, which is run at every fabric cell cycle (typically $100ns$).

3. A control theoretical model for the Gear-Box algorithm

In this section we develop an analytical model for the FOQ architecture by a control theoretical approach. In our analysis, we start with a classical discrete-time Proportional-Integral (PI) controller [4] to adjust the drop rate of each flow. We then show that an efficient algorithm can be obtained by quantizing the control decisions of the PI controller, which we call the Gear Box algorithm. Here we present the analysis for only a single flow, and omit some details due to space limitations.

The basic control structure for a particular flow k at an OUT port j is shown in Figure 2. All variables we use in this section are for the aggregate traffic in flow k originating from all IN ports and destined to OUT port j, unless we note otherwise (i.e., we omit the subscript (j,k) for notational convenience). $\lambda(t)$ is the arrival rate (bits/s) of

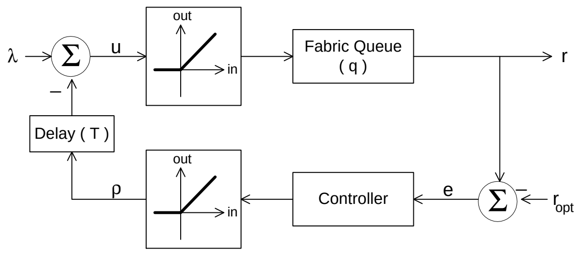

Figure 2. FOQ architecture.

flow (j,k) (i.e., traffic from all inputs destined for the OUT queue $OQ_{j,k}$). A portion of the arriving traffic is dropped at the IN droppers (we denote by $\rho(t)$ (bits/s) the aggregate rate of such dropped traffic), and the surviving portion goes into the fabric queue FQ_j at a rate $u(t) = \lambda(t) - \rho(t)$. This traffic shares the fabric queue with other traffic destined to OUT line j, and then it is delivered to OUT dropper (j,k) at a rate $r(t)$. In the following model we assume that the fabric queue does not overflow due to the FOQ mechanism.

The aggregate drop rate $\rho(t)$, is adjusted by a PI controller. The purpose of the controller is to keep the fabric output rate for packets destined to $OQ_{j,k}$ at a desired level, $r_{opt}(t) = \alpha s r_{O(j,k)}(t)$, where α, a constant smaller than but close to 1, is a safety margin and $r_{O(j,k)}(t)$ is the service rate for flow (j,k). The two nonlinearities in the figure simply state that the drop rate can not be negative or greater than the arrival rate $\lambda(t)$. In our analysis we assume that the controller is operating in the linear region, and ignore the nonlinearities.

The delay T between the output of the controller and the arrival rate models a zero-order hold at the controller output. The controller operates on time-average of the error signal taken over an interval T, rather than the signal itself, and modifies its output only at intervals of T. In the rest of this section we denote the time-averages by a discrete notation, e.g., $r[n] = \frac{1}{T}\int_{nT}^{(n+1)T} r(t)dt$ for $n = 0, 1, ...$ Assuming the amount of traffic in the fabric queue destined to $OQ_{j,k}$ does not change significantly during T at steady state, the closed-loop system is governed by the following equations at $t = nT$ epochs:

$$r[n] \approx \lambda[n] - \rho[n-1], \qquad (1)$$

$$\rho[n] = Ke[n] + K_I \sum_{m=0}^{n} e[m]. \tag{2}$$

where $e[n] = r[n] - r_{opt}[n]$.

We can now investigate the step response of the system for the case of a single flow. In this case, $r_O = c$ and $r_{opt} = \alpha sc$ are constant. The magnitude of the arrival rate can in general be larger than the maximum fabric output rate, i.e., $\lambda[n] = \lambda_0 > sc$. In this case the fabric output is constant, $r[n] = sc$, for an initial period $0 \leq n \leq N_0 - 1$. During this period the controller output increases linearly, and the fabric queue size increases until the drop rate reaches $\rho = \lambda_0 - sc$ and then decreases back to zero. The fabric queue size can be computed from (1) and (2),

$$\begin{aligned} q[n] &= T[(n+1)(\lambda_0 - sc) - nK(sc - r_{opt}) \\ &\quad - \frac{n(n+1)}{2} K_I (sc - r_{opt})] \end{aligned} \tag{3}$$

and N_0 can be derived from $q[N_0] = 0$. The behavior of the system for $n \geq N_0$ can be found by a z-transform analysis. The system characteristic equation is given by

$$z^2 + (K + K_I - 1)z - K = 0, \tag{4}$$

therefore the system is stable for

$$0 < K_I < 2(1 - K). \tag{5}$$

Using a straightforward partial fraction method, and combining this result with the previous case gives

$$\rho[n] = \begin{cases} [K + (n+1)K_I](sc - r_{opt}), & n < N_0 \\ (1 - A_1 z_1^{n-N_0} + A_2 z_2^{n-N_0})(\lambda_0 - r_{opt}), & n \geq N_0 \end{cases} \tag{6}$$

where

$$A_j = \frac{z_j^2 - \frac{K_I(N_0+1)(sc - r_{opt})}{\lambda_0 - r_{opt}} z_j}{z_1 - z_2} \tag{7}$$

for $j = 1, 2$, and z_1 and z_2 are the roots of (4).

This drop rate can be divided fairly among the N IN droppers by introducing a packet drop probability

$$p[n] = \frac{\rho[n]}{\hat{\lambda}[n+1]} = \frac{(1 - p[n-1])\rho[n]}{r[n]} \tag{8}$$

where we used the fabric output rate divided by the admit probability (i.e. $1 - p[n-1]$) as an estimate of the next average arrival rate. If we define

$$\delta[n] = \frac{(K + K_I)e[n] - Ke[n-1]}{r[n]} \tag{9}$$

then the drop probability is recursively given by

$$p[n] = (1 - \delta[n])p[n-1] + \delta[n]. \tag{10}$$

(9) and (10) are (approximately) equivalent to the PI controller without the need for an actual integrator. The ingress drops require sending $p[n]$ to all IN ports as the feedback. Since this may require too much bandwidth in a practical switch implementation, we further simplify the feedback mechanism by setting $K = 0$ and using only finite number of values for $p[n]$. Quantizing $\delta[n]$ in (9) to three levels and expressing the result in terms of the relative congestion $C[n] = 1 - r_O[n]/r[n]$ yields

$$\delta_q[n] = \begin{cases} \beta & C[n] > d_{\max} \\ \frac{\beta}{\beta - 1} & C[n] < d_{\min} \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

We call this quantized mechanism the *Gear Box (GB)* controller, since there are only three possible actions: increase the drop probability, decrease the drop probability, and no change. The two constants, $d_{\max}$ and $d_{\min}$, determine when the increase and decrease should take place, respectively. With the GB controller it is sufficient to have a 2-bit feedback signal every T seconds. Furthermore, by using (11) in (10), it can be seen that the different levels of the admit probabilities are the different powers of $(1 - \beta)$. Therefore the calculation at the IN droppers can be implemented by storing

$$P_k = 1 - (1 - \beta)^k \tag{12}$$

as a table in the memory and just updating a pointer to this table based on the feedback signal. We also set

$$\beta = \sqrt{\frac{1 - d_{max}}{1 - d_{min}}}.$$

With this selection the relative congestion after a step increase or decrease are equal, thus providing hysteresis to the GB control for stability against small perturbations.

4 Simulation experiments

We simulate a 16x10Gb/s-port switch with a $5MB$ shared memory fabric having external speedup $s = 1.28$, $2MB$ drop-tail OUT queues per flow, and no ingress queues. The FOQ-GB mechanism has a sampling rate $T = 1ms$ and feedback thresholds $d_{max} = 0.17$, $d_{min} = 0.02$. We run each simulation for $200ms$.

The offered load is composed of three flows sending at constant rates starting at $t = 0$: flow 0: $0.952Gb/s$, flow 1 and 2: $9.52Gb/s$ each, all ingressing on separate ports and exiting the same port. Given that the total offered load is $20Gb/s$, the OUT port has a potential 200% overload. The required guarantee for flow 0 is Premium service ($0.952Gb/s$ rate guarantee), and minimum rate guarantees of $7.75Gb/s$ and $1.3Gb/s$ are required for flows 1 and 2 respectively. Flow 0 is assigned to Fabric queue 0 at high

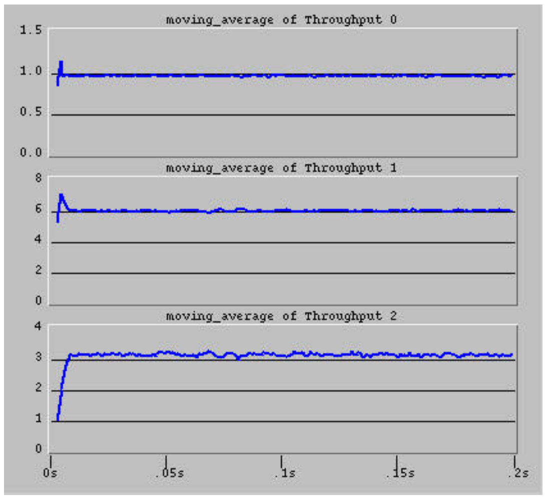

Figure 3. Throughput without FOQ

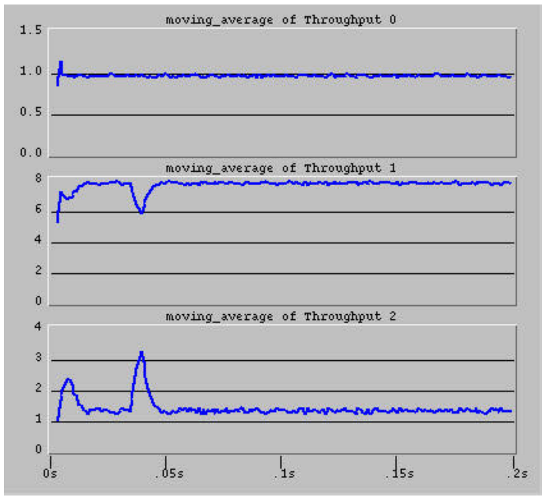

Figure 4. Throughput with FOQ

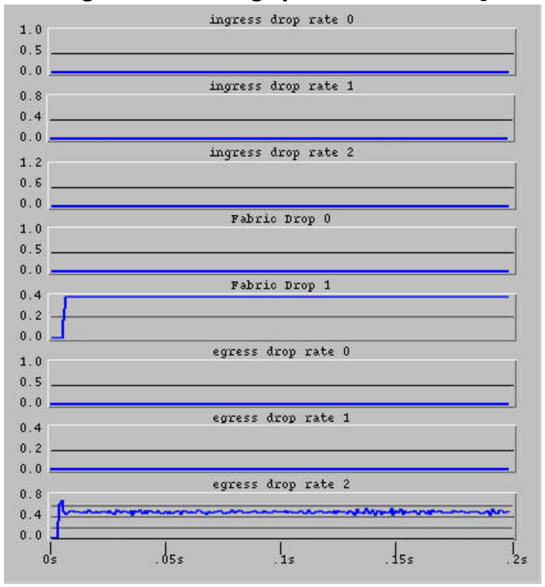

Figure 5. Drop rate without FOQ

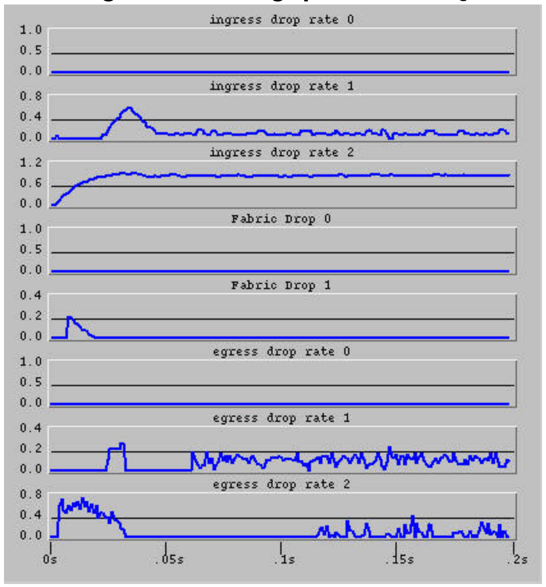

Figure 6. Drop rate with FOQ

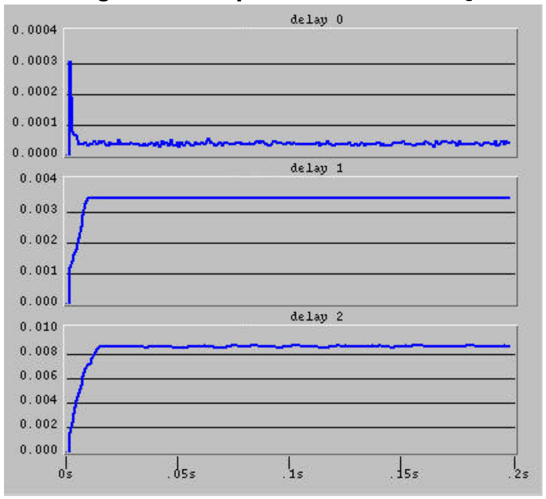

Figure 7. Delay without FOQ

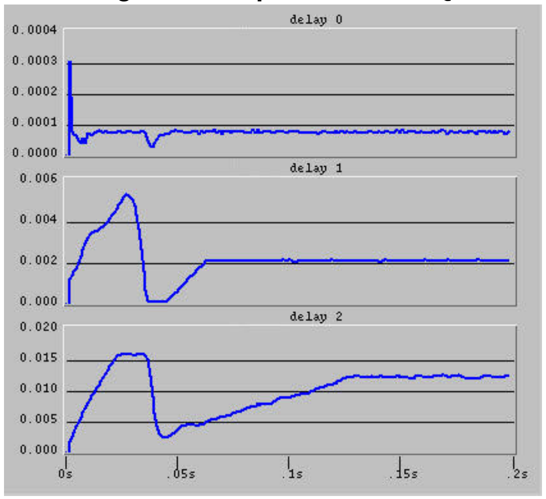

Figure 8. Delay with FOQ

priority, and flows 2 and 3 to Fabric queue 1 at lower priority. At the OUT scheduler, each flow is assigned a separate queue. Queue 0 is scheduled at high priority, whereas queues 2 and 3 are scheduled at lower priority in a Weigted Fair Queueing discipline between them with $6:1$ weights, corresponding to the required rate guarantees.

In Figures 3 and 4 we plot the evolution in time of the service rate for the three flows, without and with FOQ respectively. In Figures 5 and 6 we show the dynamics of drop rate for the same scenarios. Flow 0 is serviced at its arrival rate in both cases, due to its high priority assignment in the fabric and OUT scheduler. But the rate received by flow 1 in the non-FOQ case, $5.93Gb/s$ (Figure 3), is below its requirement. This is due to the drop in the fabric queue 1 (Figure 5) without discrimination between flows 1 and 2. When using FOQ (Figure 4), flow 1 receives $7.62Gb/s$ and flow 2 $1.37Gb/s$, thus both achieving their minimum rate guarantees. This is explained by the FOQ action reflected in Figure 6 where we see an increase of input drop for flows 1 and 2 as a reaction to output congestion. As a consequence, the fabric drop is zero almost all the time in the FOQ case, in contrast with the high drop rate in the base case. The spike in fabric drop is due to the transient state where ingress drop is increasing but not yet sufficient for eliminating fabric congestion. With FOQ, fabric drop occurs only at bursts with high rate and long duration. It can be mitigated by larger fabric memory or higher frequency of feedback. Also note that flow 0 is not affected even during the FOQ transient due to its assignment to the high priority fabric queue.

In Figures 7 and 8 we show the dynamics of packet transit delay through the whole switch. While flow 0 receives minimum delay in both cases due to its high priority assignment, flows 1 and 2 experience delays that are proportional to their respective service rates (their OUT queues are close to full in the steady state due to the drop-tail queue management).

5 Conclusion

In this paper we presented the Feedback Output Queuing architecture for packet switching that provides support for QoS guarantees. As a key feature, feedback control is applied to a fabric with small external speedup. This control enables the fabric to be virtually loss-less, thus avoiding packet drops indiscriminate of QoS class.

We apply discrete feedback control theory to derive a stable configuration. Through analysis and simulations we show that a quantized version of a PI controller named “Gear-Box control” is stable, responds quickly to traffic bursts and provides accurate QoS guaratees. We also show that FOQ’s computational complexity is much lower than VOQ with matching.

We believe that this work has sparked many venues for future research. There is a range of control algorithms to be investigated besides those presented here. The interaction between the TCP congestion control algorithm and FOQ (and RED queue management) is an interesing control problem. The FOQ architecture can be extended with a set of input queues in order to provide zero loss for a wider range of bursty traffic, given a limited fabric memory size.

6 Acknowledgements

The authors would like to thank Eric Haversat, Tom Holtey and Franco Travostino of Nortel Networks for many useful discussions.

References

[1] B. Davie et al. An Expedited Forwarding PHB (Per-Hop Behavior). Internet RFC3246, March 2002.

[2] T. A. et al. High-speed switch scheduling for local-area networks. *ACM Trans. Comp. Sys.*, 11(4), Nov 1993.

[3] S. Floyd and V. Jacobson. Random Early Detection gateways for Congestion Avoidance. *IEEE/ACM Transactions on Networking*, 1(4), August 1997.

[4] G. Franklin, D. Powell, and A. Emami-Naeini. *Feedback Contol of Dynamic Systems*. Addison Wesley, 1995.

[5] B. Gleeson et al. A Framework for IP Based Virtual Private Networks. IETF RFC 2764, Feb 2000.

[6] D. Goderis et al. Service Level Specification Semantics and Parameters. IETF Draft, Feb 2002.

[7] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski. Assured Forwarding PHB Group. Internet RFC2597, June 1999.

[8] V. Jacobson and M. J. Karels. Congestion Avoidance and Control. In *SIGCOMM’88*, 1988.

[9] K. Kar, T. Lakshman, D. Stiliadis, and L. Tassiulas. Reduced Complexity Input Buffered Switches. In *Hot Interconnects*, 2000.

[10] S. McCreary and K. Claffy. Trends in Wide Area IP Traffic Patterns. CAIDA, May 2000.

[11] N. McKeown and T. Anderson. A quantitative comparison of iterative scheduling algortihms for input-queued switches. *Comp. Netw. and ISDN Sys.*, 30, 1998.

[12] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP Reno Performance: A Simple Model and Its Empirical Validation. *IEEE/ACM Transactions on Networking*, 8(2), April 2000.

[13] W. Stevens. TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms. RFC2001, Jan 1997.

[14] A. Waldemar et al. Requirements for Virtual Private LAN Services (VPLS). IETF draft work in progress.