The Pipeback Switch: High Performance Packet Switching with Guaranteed Delivery and Linear Buffer Complexity

Saqib Raza and Zartash Afzal Uzmi
Computer Science Department
Lahore University of Management Sciences, Pakistan
{saqibr,zartash}@lums.edu.pk

***Abstract*— A high performance packet switching architecture called the Pipeback switch is proposed. This architecture ensures lossless packet delivery while maintaining linear buffer complexity. The Pipeback switch improves upon the popular Knockout switch proposed by Y. Yeh et al. Both switches use an $N \times N$ space division fabric with output queuing and both designs are motivated by the observation that the probability of more than L packets arriving in a given timeslot being destined for one particular output port sharply decreases as L is increased. This probability is comparable to the packet loss probability due to transmission errors for $L \ll N$. While the arrival of more than L packets destined for a single output in a single timeslot in the Knockout switch results in dropped packets due to *buffer blocking*, the Pipeback switch avoids such loss by maintaining a separate shared buffer architecture common to all the output ports. This common architecture consists of a novel *Pipeback concentration network* and a *buffer pool*. The buffer pool accommodates all the knocked out packets that the Knockout switch would have dropped as a result of buffer blocking, and pipes them back to a separate input line. We further show that the use of buffer pool leads to a reduction in the number of separate output buffers required at each output port.**

I. Introduction

It is well known that input queuing and its associated head-of-line (HOL) blocking limits the throughput to approximately 58% in an $N \times N$ non-blocking space division switch. Output queuing, on the other hand, achieves 100% throughput because the output queues only saturate as the utilization approaches unity [1]–[4]. Furthermore, mean queue lengths are greater for input queuing compared to when output queuing is employed. Thus, output queuing yields far better results as compared to input queuing in terms of maximum throughput and average packet delay. Implementation of output queuing is, however, comparatively difficult because of the following: during a single timeslot, many input ports might receive packets that are all addressed to a single output port. To ensure that no packet is lost in the switch fabric before it arrives at the output queue, the switch must be capable of transferring multiple packets in a single timeslot. In the worst case, packet transfer must be performed at N times the speed of the input ports. Physically this constraint translates into a need for a special mechanism to increase the bus speed of the switch [5]–[8] or requires more hardware components to allow all input ports to simultaneously access all output ports [9], [10]. Many switches used in practice are based on output buffering. The IFS Knockout, GPS, Self-routing GPS, Distributed KS, Tandem I, Shuffleout I, Christmas Tree, SCOQ and MULTIPAR are typical output queuing switches.

The Pipeback switch improves upon the Knockout switch which is a fully connected architecture that provides the implementation simplicity of input queuing while maintaining the throughput performance of output queuing. The primary motivation behind an $N \times N$ Knockout switch is that the probability of all N packets arriving at a given timeslot being destined for one particular output port is so low that the switch can be constructed to cater to L such packets, where L is a small fraction of N. Arrival of more than L packets destined for a single output in a single timeslot results in dropped packets due to *buffer blocking*. For an appropriate value of L, the number of dropped or knocked out packets as a result of buffer blocking is negligible. In this paper, we propose a scheme to reduce the number of output buffers at each output port required in the Knockout switch whilst also ensuring that no packet is dropped due to buffer blocking. The Pipeback switch avoids packet loss by using a shared buffer architecture common to all the output ports; the shared buffer is called the *buffer pool*. The buffer pool accommodates all the knocked out packets and pipes them back to a separate Pipeback input line. The benefit of this new architecture lies in the possibility of reducing L due to the added assurance that the otherwise knocked out packets are accommodated in the buffer pool. The memory at each output port can then receive the L packets that reach it. While these packets are being processed, any additional packets for

the same output will be piped back through the buffer pool and come back during subsequent timeslots. We show that the increase in the average packet delay when compared to the Knockout switch is negligible for an adequate value of L.

A characteristic of the Pipeback switch is that FIFO may be violated for a given input: a packet $\wp$ that is piped back may be delivered after a packet that arrived subsequent to the original arrival of $\wp$. Packet mis-sequencing is not strictly disallowed in a router [11], nor is it entirely uncommon for Internet traffic. Parallelism in various network components and links causes packets to be mis-sequenced under normal operation [12]. Such mis-sequencing may adversely affect the performance of TCP [12], [13]. This, however, can be mitigated by making TCP more robust to packet mis-sequencing [13]–[15]. Furthermore, the Pipeback switch mis-sequences only those packets that are piped back. Since the probability of packets being piped back is low, the mis-sequencing of packets in the Pipeback switch is limited. We also note that the piped back packets delivered out of sequence, would have otherwise been dropped in an equivalent Knockout switch. Packet loss caused by the Knockout switch has worse implications on the performance of TCP than mis-sequencing resulting from the Pipeback switch. Nonetheless current implementations of TCP work best when the packets are not mis-sequenced [16]. Packet re-sequencing problem can be addressed in the Pipeback switch by adding a re-sequencing buffer at each output port. However, this entails increased hardware complexity. Efficient packet re-sequencing is a problem of interest for the Internet community and is currently being investigated.

This paper is organized as follows: Section II gives a brief description of the Knockout switch. Section III details the architecture of the Pipeback switch and also provides a description of the Pipeback concentration network. Performance analysis of the Pipeback switch and comparison with the Knockout switch is presented in section IV. We present our conclusions in section V.

II. The Knockout Switch

The interconnection fabric of an $N \times N$ Knockout switch has two basic characteristics: each input has a separate broadcast bus, and each output has access to packets arriving at all inputs. Thus, an output module directly interfaces with each of the N broadcast buses. Each output module has four major components: N packet filters, an $N \times L$ concentrator where $L \ll N$, a shifter to ensure that the output buffers are filled cyclically, and L output buffers. An output module and with these components is illustrated in Fig. 1. Fixed-length packets arrive at the inputs in discrete timeslots and are broadcast to each output module. The packet

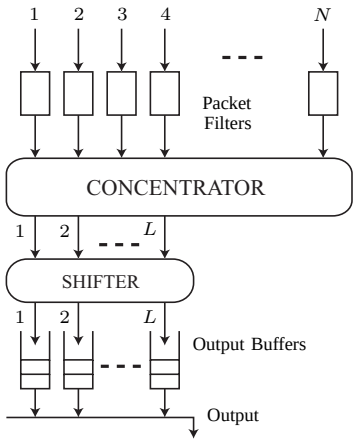

Fig. 1. The Knockout Switch Output Module

filters examine each packet and set an activity bit if the packet is addressed to that output. The packets then enter an $N \times L$ concentrator that forwards up to L active packets to the output buffers through the shifter. If more than L packets arrive for one output in a single timeslot, the additional packets are dropped (knocked out). By properly choosing L, the probability of packet loss can be controlled to be comparable to the packet loss probability due to transmission errors [10].

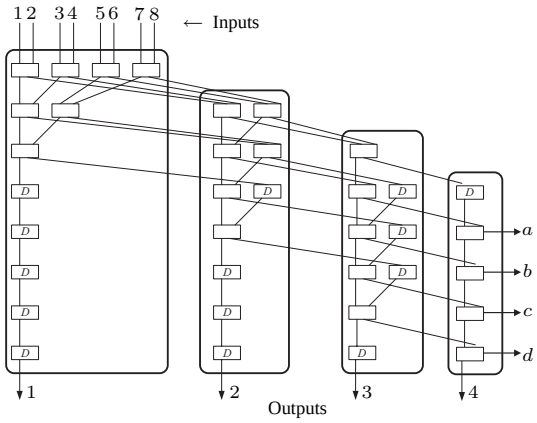

Fig. 2. The Knockout Switch Concentration Architecture

This self-induced source of packet loss is motivated by the desire to reduce the number of output buffers. The Knockout switch provides complete sharing of the L output buffers and a first-in-first-out discipline for the transmission of packets arriving in those buffers.

Fig. 2 shows an 8×4 Knockout concentrator composed of 2×2 switch elements and delay elements marked by D. At each switch element, packets compete and the winner comes out of the left output. The algorithm to set the switch elements is that an active packet never

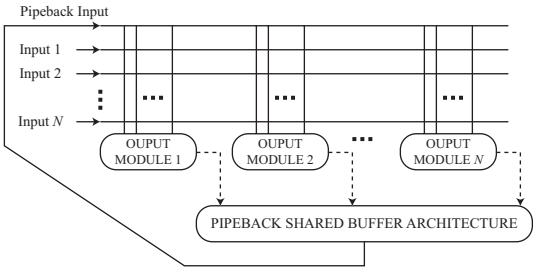

Fig. 3. Interconnection Fabric of the Pipeback Switch

loses to an inactive packet. When two active packets or two inactive packets compete among themselves, the winner is chosen randomly. Thus, if there is only one active packet, it comes out of the output labelled 1. If there is another active packet, it comes out of the output labelled 2, and so on. When the number of active packets exceeds 4 (L in general), four packets are transferred to the shifter and the rest are knocked out appearing at the *knockout lines*. An important observation used in the design of the Pipeback concentration network, detailed in the next section, is that active packets appear at the knockout lines in an ordered fashion, from a to d. This means that the output labelled c has an active packet only if outputs labelled a and b both have active packets.

III. The Pipeback Switch

A. Pipeback Switch Architecture

The switch fabric used in the Pipeback architecture is similar to that used in the Knockout switch. The Pipeback switch consists of $N + 1$ inputs and outputs. The inputs include the N regular inputs as are present in the case of the Knockout switch and an additional Pipeback input as shown in Fig. 3. As in the Knockout switch, each fixed-length packet arriving at one of the input ports is placed on the broadcast bus for that port. The packet filters mark the active packets, which then enter an $(N + 1) \times L$ Knockout concentrator that feeds the L output buffers through the shifter. Thus, up to L active packets are accommodated in the output buffers. Any active packets in excess of L, which would have been dropped by the Knockout switch, are directed to another concentration network called the *Pipeback concentration network*. It is important to note that there is a single Pipeback concentration network common to all the output modules as shown in Fig. 4. This network takes $N - L + 1$ inputs from each of the N output modules, therefore having a total of $N(N-L+1)$ inputs, and achieves an $N(N-L+1)$ to $N-L+1$ concentration. Since the maximum number of active packets arriving at the Pipeback concentration network in a single timeslot is $N - L + 1$ (when all $N + 1$ packets are addressed

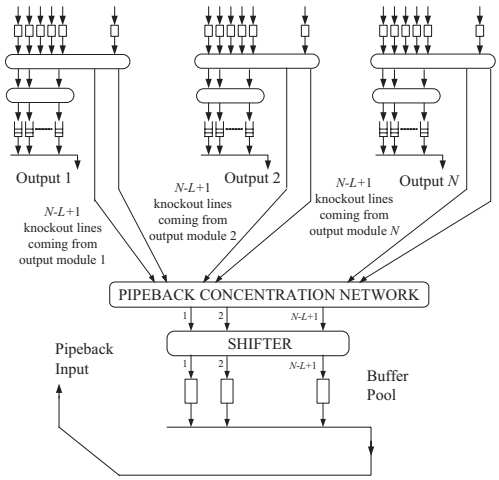

Fig. 4. The Pipeback Switch

to the same output port), no active packet is lost in the Pipeback concentration network.

The Pipeback concentration network interfaces through a shifter to another shared memory referred to as the *buffer pool*. The buffer pool is implemented as a set of $N - L + 1$ physical queues and is synchronized so that it may pipe back its contents through the *Pipeback input*, one packet at a time, during subsequent timeslots. Thus, the common buffer pool provides a mechanism that precludes the possibility of packets being knocked out while retaining the linear complexity of the Knockout switch. We will further show that the use of buffer pool leads to a reduction in L, the number of output buffers at each output port, without any significant increase in the average packet delay.

B. Pipeback Concentration Network Structure

The Pipeback concentration network has $N - L + 1$ inputs from each of the N output modules for a total of $N(N - L + 1)$ inputs. These inputs are concentrated onto $N - L + 1$ outputs. From [10], we note that each of the L sections of the Knockout concentrator contains approximately N switch elements. The number of switch elements needed to construct an Knockout concentrator is, therefore, NL. Constructing the Pipeback concentration network exactly like the Knockout concentrator results in a switch element complexity of $O\left(N^3\right)$.

The switch element complexity of the Pipeback concentration network can be reduced. To show this, we note that among the $N(N - L + 1)$ inputs to the Pipeback concentration network, a maximum of $N - L + 1$ inputs may be active during a single timeslot, assuming there is no multicast traffic. Further note that the knocked out packets from the Knockout concentrator at each output module are ordered such that if m active packets are knocked out, they come out at the first m of the $N-L+1$

knockout lines of that output module, as shown in Fig. 2 and explained in section II. We label the $N-L+1$ knockout lines from 1 to t for each output module, where $t = N-L+1$. We then divide the Nt inputs coming from the knockout lines of all the N output modules into t mutually exclusive and collectively exhaustive sets, each with cardinality N. The knockout lines having the same label and belonging to different output modules constitute one set, as indicated in Fig. 5. The sets are also labelled and the label of a set is the same as the labels of the knockout lines included in it.

If there are y active packets in set i, where i is an integer between 1 and t, then at least $y(L+i)$ active packets arrived at the switch during that timeslot. Since the maximum number of packets that can arrive in one timeslot is $N+1$, we get the constraint $y(L+i) \leq N+1$. Therefore, x_i, the maximum number of knockout lines in set i with an active packet in a single timeslot, can be obtained from $x_i(L+i) = N+1$, which yields $x_i = \left\lfloor \frac{N+1}{L+i} \right\rfloor$. It is evident that x_i is maximum for $i = 1$ and minimum for $i = t$.

Fig. 5 depicts the implementation of the Pipeback concentration network. Each of the t sets of knockout lines is input to a corresponding $N \times x_i$ Knockout concentrator. The outputs of these knockout concentrators collectively feed another Knockout concentrator with $\sum_{i=1}^{t} x_i$ inputs and t outputs. These outputs interface with the buffer pool through a shifter network. The buffer pool can receive a maximum of t packets in a single timeslot, which is also the maximum number of packets that may appear on all the knockout lines. Therefore, the Pipeback concentration network ensures that all knocked out packets are accommodated in the buffer pool. The number of switch elements required for the Pipeback concentration network is given by:

$$N \sum_{i=1}^{t} \left\lfloor \frac{N+1}{L+i} \right\rfloor + t \sum_{i=1}^{t} \left\lfloor \frac{N+1}{L+i} \right\rfloor \tag{1}$$

For $L \ll N$, we have $t \approx N$, then the switch element complexity of the Pipeback concentration network is given by:

$$O\left(N \sum_{i=1}^{N} \left\lfloor \frac{N}{i} \right\rfloor \right) = O\left(N^2 \log N \right) \tag{2}$$

IV. Performance Analysis

A. Assumptions

In this section, we present the performance analysis of the Pipeback switch and use it to compare the Pipeback and the Knockout switching architectures. Towards this end, we assume the following for both switches:

i) Fixed-length packets arrive at the switches in synchronized timeslots.

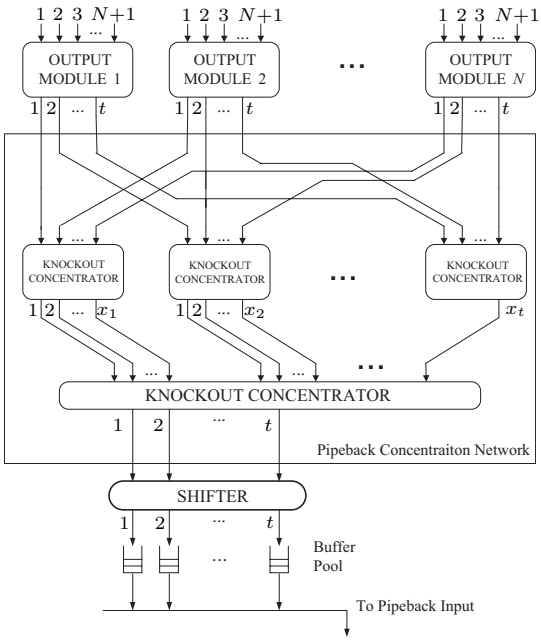

Fig. 5. The Pipeback Concentration Network

ii) Arrivals at each input are independent and identically distributed over time.

iii) Arrival processes at inputs are independent of each other and have the same arrival rate ρ.

iv) For each packet, destinations are uniformly distributed over all outputs.

For the Pipeback switch, these assumptions apply to the N external inputs. The arrival process at the Pipeback input, however, is internally controlled by the switch architectural parameters.

B. The Knockout Switch

The probability $\beta_{KO}(k)$ for k packets arriving in a timeslot destined for a given output is given by the binomial distribution:

$$\beta_{KO}(k) = \binom{N}{k} \left(\frac{\rho}{N}\right)^k \left(1 - \frac{\rho}{N}\right)^{N-k} \tag{3}$$

From above expression, the probability of packet loss in the Knockout concentrator is [10]:

$$\pi_c = \frac{1}{\rho} \sum_{k=L+1}^{N} (k-L)\, \beta_{KO}(k) \tag{4}$$

As $N \to \infty$, this probability is given by [10]:

$$\pi_c = \left(1 - \frac{L}{\rho}\right) \left[1 - \sum_{k=0}^{L} \frac{\rho^k e^{-\rho}}{k!}\right] + \frac{\rho^L e^{-\rho}}{L!} \tag{5}$$

For large N, the value of L required to maintain a given loss rate is relatively small and is independent of N. For example, $L = 8$ is sufficient to maintain the concentrator

packet loss rate at one packet per million, for large N and full input load. Moreover, L only needs to grow logarithmically for further reduction in the loss rate. For example, $L = 11$ reduces the loss rate to one packet in a billion.

The L output buffers at each output behave as a $Geom(L)/D/1/B_0$ queue with offered load $\lambda_{load} = \rho(1 - \pi_0)$ [17]. As $B_0 \rightarrow \infty$, the queue becomes $Geom(L)/D/1$ and the average waiting time for a packet is given by [17]:

$$\tau_b = \frac{L-1}{L} \cdot \frac{\lambda_{load}}{2(1-\lambda_{load})} \tag{6}$$

C. The Pipeback Switch

As we noted above, the packet loss probability can be characterized in terms of the probability of packets arriving in a single timeslot destined for the same output. While the latter follows a binomial distribution for the Knockout switch and it is straightforward to write its expression, writing a similar expression for the Pipeback switch requires considering three distinct scenarios:

i) $k-1$ packets arrive at any of the N external inputs and 1 packet arrives at the Pipeback input destined for the output we are considering, or
ii) k packets arrive at N external inputs and no packet arrives at the Pipeback input, or
iii) k packets arrive at any of the N external inputs and 1 packet that arrives at the Pipeback input is destined for one of the $N-1$ outputs other than the one we are considering

All of these three scenarios result in k packets arriving in a single timeslot destined for the same output – the one that we are considering. Let α_i be the probability of a packet arriving at the Pipeback input in timeslot i. Then, the probability $\beta_{PB}(k,i)$ of k packets arriving in timeslot i destined for a particular output is given by:

$$\beta_{PB}(k,i) = \binom{N}{k}\left(\frac{\rho}{N}\right)^k\left(1-\frac{\rho}{N}\right)^{N-k}\left(1-\frac{\alpha_{i-1}}{N}\right) + \binom{N}{k-1}\left(\frac{\rho}{N}\right)^{k-1}\left(1-\frac{\rho}{N}\right)^{N-k+1}\left(\frac{\alpha_{i-1}}{N}\right) \tag{7}$$

For bounded queue size, α_i is the same as the average number of arrivals into the Pipeback queue from the N concentrators in timeslot i and is given by:

$$\alpha_i = N \sum_{k=L+1}^{N} (k-L)\,\beta_{PB}(k,i) \tag{8}$$

Since equations (7) and (8) are collectively recursive, we simplify the analysis by first assuming that the Pipeback input has a packet in every timeslot, i.e., we take $\alpha_0 = 1$ in equation (7) to evaluate $\beta_B(k,1)$ for the worst case performance of the Pipeback switch. And then, we perform two recursions of these equations to get:

$$P_k = \beta_{PB}(k,2); \qquad \lambda_{PB} = \alpha_2 \tag{9}$$

For further analysis, we use P_k and λ_{PB} to bound the performance of the Pipeback switch. Note that P_k is the probability of k packets arriving in a single timeslot destined for the same output and λ_{PB} is the arrival rate at the Pipeback input queue. Since the Pipeback input queue sends one packet per timeslot, the theoretical maximum value for λ_{PB} is 1. The arrival rate λ_{OP} at each of the output queues can also be given in terms of P_k as follows:

$$\lambda_{OP} = \sum_{k=0}^{L} kP_k + L \sum_{k=L+1}^{N+1} P_k \tag{10}$$

Since the arriving packets have a fixed length and the transmission time for each packet is deterministic (i.e., one timeslot), the queuing processes at each output queue and at the Pipeback input queue can be modeled as $M/D/1$ queuing disciplines. Thus, the average waiting times in these queues can be evaluated as:

$$\text{For Pipeback input queue: } \tau_{PB} = \frac{\lambda_{PB}}{2(1-\lambda_{PB})} \tag{11}$$

$$\text{For the output queue: } \tau_{OP} = \frac{\lambda_{OP}}{2(1-\lambda_{OP})} \tag{12}$$

Finally, the average waiting time for a packet through the Pipeback switch is:

$$\tau_W = \tau_{OP} + \left(\frac{\lambda_{PB}}{\lambda_{PB} + N\lambda_{OP}}\right)\tau_{PB} \tag{13}$$

where the term in parentheses indicates the fraction of packets that are piped back. Fig. 6 depicts the mean waiting times of packets in a Knockout switch with $L = 8$ and a Pipeback switch with $L = 3$ for various arrival rates. The mean waiting times in the two cases are almost identical. The Pipeback switch, however, uses fewer buffers ($L = 3$ as compared to $L = 8$) and does

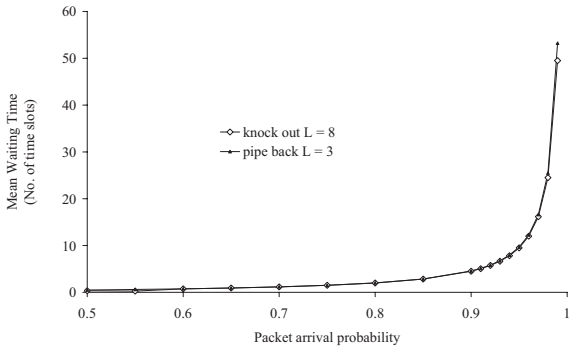

Fig. 6. Mean waiting times for $N = 32$ and infinite-length queues

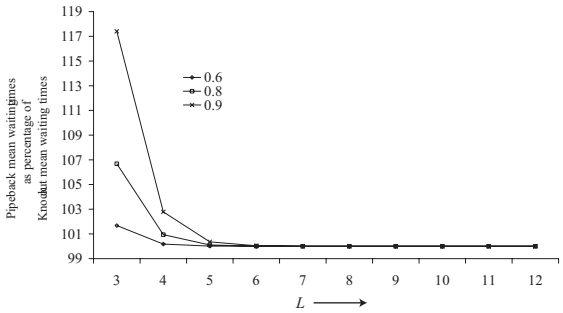

Fig. 7. Mean waiting times for the Knockout and Pipeback switches

not suffer from buffer blocking, thus avoiding packet loss.

The mean waiting times for the Pipeback switch as a percentage of the mean waiting times for the Knockout switch, for three different arrival rates, are shown in Fig. 7. It can be seen that the difference between the two mean waiting times is marginal and quickly diminishes for increasing values of L. Additionally, the Pipeback implementation provides the assurance that a packet knocked out of an output module will subsequently be piped back. Note that a knocked out packet always leaves at least L packets in the output buffers. With the Pipeback mechanism, it can go to the Pipeback input to arrive before the L packets in the output buffers are processed. Thus, there is no use putting buffers to separately queue the packets knocked out of different output modules.

Our results show that for $L \geq 4$, and packet arrival probability less than or equal to 0.9, the increase in the mean waiting times of packets from Knockout switch to the Pipeback switch is insignificant. Therefore, the Pipeback switch reduces the number of required output buffers while ensuring lossless packet delivery

V. Conclusions

A fully connected high performance packet switching architecture called the Pipeback switch is proposed. The Pipeback switch, like the Knockout switch, achieves the performance advantages of output queuing with the implementation simplicity of input queuing. At each output module, the Knockout switch uses $L \ll N$ output buffers and drops packets in excess of L in a single timeslot. It is observed that while such dropping is negligible for a single concentrator, this is not the case when many Knockout switches are connected in series. We, therefore, proposed the Pipeback switch that pipes back all the knocked out packets to a separate Pipeback input. Thus, the Pipeback switch avoids buffer blocking ensuring lossless packet delivery. We showed that the Pipeback architecture allows a reduction in the number of output buffers required at each output module. We further demonstrated that this reduction does not significantly increase the mean waiting time for the packets. This is possible because a packet is only piped back when there are at least L packets in the output buffers of the same output module. Thus, with high probability, the piped back packet will come back before other L packets in the output buffers are processed. We also provided a design for the Pipeback concentration network to reduce its switch element complexity. As such the Pipeback switch offers high performance packet switching with linear buffer complexity and lossless packet delivery.

References

[1] N. McKeown, A. Mekkitikul, V. Anantharam and J. Walrand, "Achieving 100% throughput in an Input-Queued Switch," IEEE Trans. Communications, vol. 47, No. 8, pp. 1260-1267, August 1999

[2] A. Mekkitikul and N. McKeown, "A Practical Scheduling Algorithm to Achieve 100% Throughput in Input-Queued Switches," Proc. of IEEE INFOCOM '98, pages 792-799

[3] N. McKeown, V. Anantharam, and J. Walrand, "Achieving 100% Throughput in an Input-Queued Switch," Proc. of IEEE INFOCOM '96, vol. 1, pp. 296-302

[4] J. S.-C. Chen and T. E. Stern, "Throughput Analysis, Optimal Buffer Allocation, and Traffic Imbalance Study of a Generic Nonblocking Packet Switch," IEEE Journal on Selected Areas in Communications, vol. 9, no. 3, pp. 439-449, April 1991

[5] M. De Prycker and M. De Somer, "Performance of a Service Independent Switching Network with Distributed Control," IEEE Journal on Selected Areas in Communications, vol. 5, no. 8, pp. 1293-1301, October 1987

[6] K. Hajikano, K. Murakami, E. Iwabuchi, O. Isono, and T. Kobayashi, "Asynchronous Transfer Mode Switching Architecture for Broadband ISDN – Multistage Self-Routing Switching (MSSR)," Proc. of ICC '88, vol. 2, pp. 911-915

[7] J. S. Meditch and X.Jiang, "An Integrated Services Fast Packet/Fast Circuit Switch," Proc. of ICC '89, vol. 1, pp. 104-110

[8] H. Suzuki, H. Nagano, T. Suzuki, T. Takeuchi, and S. Iwasaki, "Output-Buffer Switch Architecture for Asynchronous Transfer Mode," Proc. of ICC '89, vol. 1, pp. 99-103

[9] H. Ahmadi, W. E. Denzel, C. A. Murphy, and E. Port, "A High-Performance Switch Fabric for Integrated Circuit and Packet Switching," Proc. of INFOCOM '88, pp. 9-18

[10] Y. Yeh, M. G. Hluchyj, and A. S. Acampora, "The Knockout Switch: A Simple, Modular Architecture for High-Performance Packet Switching," IEEE Journal on selected Areas in communications, vol. SAC-5, no. 8, pp. 1274-1283, October 1987

[11] F. Baker, "Requirements for IP Version 4 Routers," RFC - 1812, Network Working Group, June 1995

[12] J.C.R. Bennett, C. Partridge and N. Shectman, "Packet reordering is not pathological network behavior," IEEE/ACM Transactions on Networking, Vol. 7, No. 6, pp. 789-798, December 1999

[13] E. Blanton and M. Allman, "On making TCP more robust to packet reordering," ACM Computer Communication Review, 32(1), January 2002

[14] M. Allman, H. Balakrishnan, and Sally Floyd, "Enhancing TCP's Loss Recovery Using Limited Transmit," RFC 3042, January 2001

[15] R. Ludwig and R. Katz, "The Eifel Algorithm: Making TCP Robust Against Spurious Retransmissions," Computer Communication Review, 30(1), January 2000

[16] I. Keslassy and N. McKeown, "Maintaining packet order in two-stage switches," Proc. of IEEE INFOCOM, 2002

[17] A. Pattavina, Switching Theory, John Wiley & Sons, 1998