Optical Packet Buffers with Active Queue Management

Assaf Shacham and Keren Bergman

Columbia University, Department of Electrical Engineering, New York, NY 10027
bergman@ee.columbia.edu,
http://lightwave.ee.columbia.edu

Abstract. Active queue management (AQM) is an important function in today's core routers that will be required in the future optical internet core. A recently reported novel architecture for optical packet buffers is extended by implementing necessary AQM functions. The suggested AQM scheme is validated and explore via simulations.

1 Introduction

One of the key challenges to the implementation of all-optical routers is the difficulty of realizing optical packet buffering. Contemporary internet routers use very large packet buffers, which store millions of packets, to efficiently utilize expensive long haul links. These large buffers are clearly impractical for implementation using photonic technology. Recent studies have indicated that by sacrificing some of the link utilization, buffer sizes can be reduced dramatically to the capacity of approximately 20 packets [3].

Photonic packet buffers of this capacity have the potential of realization. Numerous optical buffer architectures have been suggested (see, for example, Refs. [1, 2, 5], among others). However, many of the suggested architectures suffer from fundamental drawbacks that prohibit scaled implementations or make them unusable in optical routers. For example, some are designed to store a single packet and others require a complex and nonscalable control schemes. In previous work [8, 9] we have presented a new buffer architecture that is modular, scalable, extensible, and transparent and therefore provides significantly improved performance.

In this paper we extend this work by considering an active queue management (AQM) scheme which can be straightforwardly implemented on the optical buffer architecture. AQM is a technique used for congestion control in packet-switched routers. In a typical AQM technique, known as random early detect (RED) [4], packets are **deliberately dropped** even when the router's buffers are not completely full, to provide an early congestion notification (ECN) signal to the TCP terminals and the network endpoints.

In the scheme suggested here, AQM is employed to solve another problem: packet-loss may occur when a low-capacity buffer is operated under high load. While this may be a price that a designer is willing to pay in an optical router

with small buffers [3], the buffer is operated in a nearly full state most of the time, thus incurring an unnecessarily high queueing latency on all packets. AQM can be used to reduce the queueing latency while increasing the packet loss rate (PLR) by a small amount.

This paper is organized as follows: In Section 2 the buffer architecture is reviewed and explained. The suggested AQM scheme and its mapping on the existing architecture are presented in Section 3. A simulation based exploration of the AQM parameters then follows in Section 4 and a concluding discussion is provided in Section 5.

2 Architecture Overview

The optical packet buffer is comprised of identical buffer building-block modules that are cascaded to form a complete buffer (see Fig. 1). Each building-block module has two input ports and two output ports and is capable of storing a single packet on a fiber delay line (FDL). A pair of ports (*Up-in* and *Up-out*) connects the module to the next module in the cascade, and the *Down-in* and *Down-out* ports are connected to the previous module. In the root module the *Down-in* and *Down-out* ports are used as the system input and output ports, respectively. Each module is also connected to the next module in the cascade by an electronic cable, for the transmission of *Read* signals.

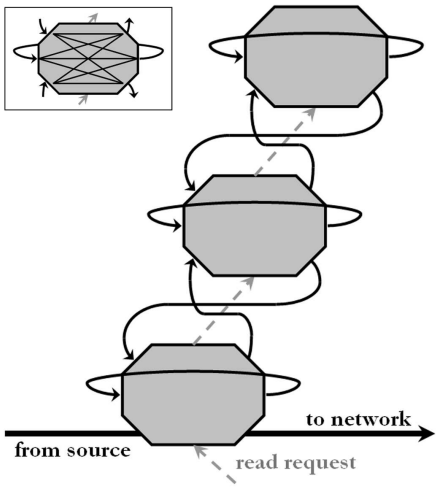

Fig. 1. The buffer's cascade structure (large image). A 3×3 non-blocking switch, of which a reduced version is used to implement the modules (inset) [9]

Writing packets to the buffer is performed implicitly: optical packets arrive into the *Down-in* port of the root module, aligned with system timeslots. The

packets are either stored locally if the internal FDL buffer is empty, or routed to the *Up-out* port to be stored in the next module in the chain. Each packet is forwarded in this manner up the cascade, and is stored in the first empty module it encounters, which is necessarily the last position in the queue.

The read process is completely independent from the write process: when a *Read* signal is received, the locally stored packet is transmitted from the *Down-out* port and a *Read* signal is sent to the next module in the cascade to retrieve the next packet. With each *Read* signal all the packets in the chain move a step closer to the output port, while maintaining the packet sequence.

This distributed modular structure has several advantages: (1) no central management is required – all modules follow an identical simple set of rules; (2) the buffer capacity can be increased simply by connecting additional modules at the end of the cascade; (3) packet dropping, in the case of overflow, is cleanly executed by routing packets to the *Up-out* port of the last module, which is not connected.

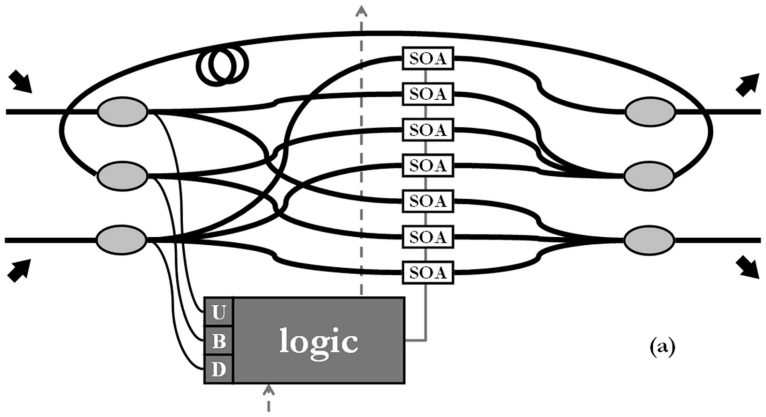

Fig. 2. Schematic diagram of the building-block module with ovals for couplers; U, B, and D for their respective low-speed receivers; dashed lines for read request signals [9]

2.1 Building-block module structure

Each building-block module in the buffer is implemented as an SOA-based 3×3 non-blocking optical switch (Fig. 2). At each of the three inputs (*Up-in*, *Down-in*, and *Buffer-in*), the packet's power is split by a coupler. A portion of the packet power is directed to a low-speed *p-i-n*-TIA optical receiver acting as an envelope detector. The envelopes of the packets from all input ports along with the input electronic *Read* signal are used in an electronic decision circuit to set the state of the SOA-based switch. The decision rule used by the electronic circuitry is represented by the truth table in Table 1. The table should be read

as follows: the four left columns R,U,B,D are the *Read* signal, and the three inputs representing the existence of packet on the input ports (*Up-in*, *Buffer-in*, *Down-in*), respectively. The next nine columns represent the switching states of the SOAs (e.g. U2B means *Up-in* to *Buffer-out*, etc.) Finally, the last column (ER) represents when a *Read* signal is transmitted to the next module in the chain.

Table 1. Truth table for building-block module ($\emptyset$ represents a don't-care value)[8]

R	**U**	**B**	**D**	**U2U**	**U2B**	**U2D**	**B2U**	**B2B**	**B2D**	**D2U**	**D2B**	**D2D**	**ER**
$\emptyset$	0	0	0										
0	0	0	1								1		
0	0	1	0					1					
0	0	1	1					1		1			
0	1	0	0		1								
0	1	0	1		1					1			
$\emptyset$	1	1	$\emptyset$	*illegal states*									
1	0	0	1									1	
1	0	1	0						1				1
1	0	1	1						1	1			1
1	1	0	0			1							1
1	1	0	1			1				1			1

To assist in understanding the decision rule, the following examples are given: line {0011} represents a case where a new packet is received from the *Down-in* port (D=1), while another packet is locally stored in the buffer (B=1). In that case the locally stored packet is routed back to *Buffer-out* (B2B=1) while the new packet is routed to the next module through the *Up-out* port (D2U=1). Line {1010} represents the case where a *Read* signal is received (R=1) and a packet is locally stored (B=1). In this case, the local packet is sent to the *Down-out* port (B2D=1) and a *Read* signal is sent to the next module (ER=1) to retrieve the next packet in the queue. Careful inspection of Table 1 reveals that some states can never occur if all the modules follow the rules. More importantly, two paths (*Up-in* to *Up-out* and *Buffer-in* to *Up-out*) are never used so the module can be implemented using only seven SOAs, as appearing in Fig. 2.

3 Introducing AQM to the Buffer

As explained in the introduction, the suggested buffer architecture, as any other FDL-based optical buffer architecture, is practical for the implementation of small buffers, capable of storing tens of packets, at most. In an optical router, typically, a buffer would be designed to store as many as 20 packets [3]. In these systems, when routers face periods of high load and congestion, it is reasonable

to assume that buffers will fill up very quickly and packet dropping due to buffer overflow will be frequent. While higher layer mechanisms, such as TCP, are used to recover lost packets and regulate the transmission rates of network terminals in these cases, these mechanisms operate with fairly slow time-constants. Thus, when overflow-generated packet-loss is used as the sole mean of congestion notification, buffers remain full and overflowing for long periods of time, and the penalty in terms of packet loss rate (PLR) and end-to-end latency can be severe. Further, in the case of buffer overflow many packets are dropped at the same time and thus many sources are requested to throttle their transmission at the same time. This effect leads to the well-known and undesirable effects of terminal-synchronization, TCP oscillations, and failure to reach equilibrium [6].

AQM can be used, in optical routers, to improve the overall network performance by dropping packets even before the buffers reach overflow. By dropping these packets, an early congestion notification (ECN) signal is sent to the end terminals, thus causing rate throttling before the buffers reach full capacity. Further, when probability-based methods are employed to decide when to drop packets, ECN messages are sent to different TCP terminals in different times, oscillations are prevented and the network is more likely to reach equilibrium [6, 4].

The distributed structure of the buffer facilitates a simple AQM implementation. The digital decision circuit in every building block module is modified by the addition of a circuit emulating a Bernoulli random number generator (BRNG), or a biased coin-flip. In every slot, the result of the BRNG (a true/false value) is checked. If the BRNG result is **true** while a packet is about to be buffered in the module, the packet is discarded. This policy is implemented by performing a logical *AND* operations of the individual control signals of three SOA gates (U2B, B2B, and D2B) with the negation of the BRNG value:

$$U2B_{NEW} = U2B \wedge \neg BRNG$$

$$B2B_{NEW} = B2B \wedge \neg BRNG$$

$$D2B_{NEW} = D2B \wedge \neg BRNG$$

When a packet is dropped, a read signal is emitted to the next module up the cascade, so that the module is replenished and the packets in the queue advance by one spot:

$$ER_{NEW} = (U2B \vee B2B \vee D2B) \wedge BRNG$$

The BRNG in each module is programmed with a different Bernoulli parameter which is the probability that a packet buffered in that module will be dropped. The parameter in the kth module (The 0th module is the root module) is $p_d(k)$, or the **packet-dropping probability** function. Hence, in a given slot time a packet which is buffered in the kth spot in the queue is discarded with a probability $p_d(k)$.

The packet-dropping probability function is a monotonously non-decreasing function in k and $p_d(0) = 0$. Thus, packets are never dropped in the root module

and the probability that a packet is dropped rises in the modules that are further down the queue. Constructing specific $p_d(k)$ functions is the topic of Section 4.

The feasibility and correctness of the suggested distributed AQM scheme is validated using simulations on a specifically developed simulator, built using the OMNeT++ event-driven simulation environment [7]. OMNeT++ provides support for modular structures and message exchange and the simulator is highly parameterized and offers complete configuration flexibility to simulate buffers of varying sizes and varying internal parameters. The simulator is also used to explore the design parameters of the AQM probability function in the next section.

4 Exploring AQM Parameters

The buffer performance, i.e. PLR and latency, is highly dependent on the chosen dropping-probability function. As described in Section 3, the function $p_d(k)$ is a monotonously non-decreasing function in k and $p_d(0) = 0$. To perform an initial study of the optical buffer with AQM, and to explore different functions we define the function $p_{AQM}(k)$ as follows:

$$p_{AQM}(k) = \left\{ \begin{array}{cc} 0 & k \leq x_1 \\ \frac{y_2 \cdot (k - x_1)}{C - x_1} & k > x_1 \end{array} \right\}$$

where C is the total number of modules in the buffer (i.e., the buffer capacity) and x_1 and y_2 are parameters defining a given function. Fig. 3 clarifies the $p_{AQM}(k)$ function and its parameters: x_1 is the *threshold capacity* above which packets start to be actively dropped and y_2 is the maximum drop probability under AQM.

In this section we explore the effect of the two parameters: the threshold capacity (x_1) and the maximum drop probability (y_2), examine their effect on the PLR and latency, and compare them to the case where AQM is not used. In the simulations conducted, the arrival and service processes are both Bernoulli. The goal of the simulations is perform an initial evaluation of the suggested AQM scheme. The simulated case is, therefore, constructed such that the buffer is heavily loaded and the PLR due to overflow, in the absence of AQM, is fairly high (nearly 3%). To achieve these conditions, the arrival parameter is chosen to be $p = 0.50$ and the service parameter is $q = 0.52$. The buffer capacity is chosen to be 10 packets.

Obviously, to fully evaluate the performance of an AQM scheme, the systems should be simulated with traffic comprised of a large number of multiplexed TCP flows, such that the interaction of the AQM mechanism and the TCP sources can be studied. This work is planned but is beyond the scope of this paper. The work here studies the effect of the AQM function on the latency and PLR under simpler traffic models.

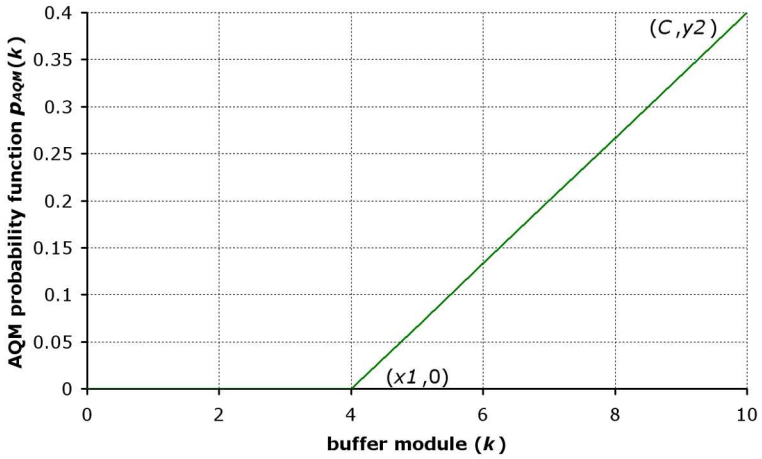

Fig. 3. The AQM dropping-probability function, defined by x_1 and y_2. In this case $x_1 = 4$, $y_2 = 0.4$ and the buffer capacity is $C = 10$.

4.1 Controlling AQM Maximum Dropping Probability

To evaluate the effect of the dropping-probability function, we first simulate different values of y_2, the maximum drop probability. This value can be seen as the *intensity* of the AQM scheme. When y_2 is increased the dropping probability due to AQM is increased and the dropping due to overflow is reduced. Simulation should be used, however, to verify that the AQM is not too strong such that it unnecessarily increases the total PLR. The results of the y_2 study are shown in Fig. 4. In these simulations, the threshold capacity value is $x_1 = 4$.

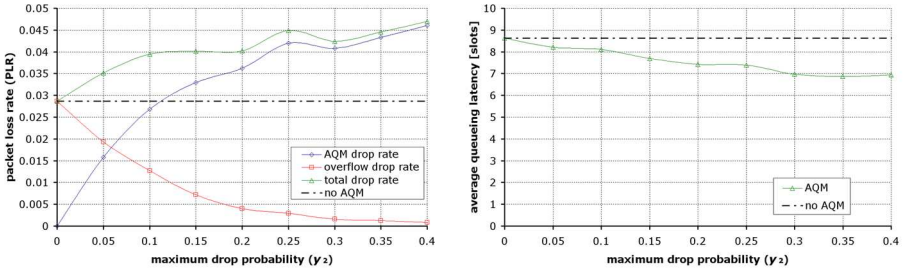

Fig. 4. The PLR and average latency of a 10-packet buffer with varying levels of maximum drop probability, y_2 ($x_1 = 4$).

The results in Fig. 4 show that with any degree of AQM applied to the buffer, the total PLR increases. As expected, as the AQM becomes more dominant, the PLR due to overflow is diminished. The negative effect of the higher PLR is

accompanied by a favorable effect of lower queueing latency, resulting from the lower occupancy in the buffer due to the AQM. Another effect that should be considered is that we can view the AQM packet drops as *better* drops, because of the more gradual effect they have on the TCP sources at the network edges [6, 4].

4.2 Controlling the Threshold Capacity

The threshold capacity can be used as a mean of controlling the spread of the AQM function across the buffer. In this case, for a given maximum dropping-probability level, the AQM threshold and the slope of the AQM function are varied. When x_1 is low, the function becomes smoother: packet dropping starts at a low capacity and the dropping-probability function rises slowly. When, conversely, x_1 is high, the AQM does not have an effect until the buffer occupancy is high, and the dropping-probability function, then, rises sharply. The results of the x_1 value study are shown in Fig. 5. In these simulations, the maximum drop probability value is $y_2 = 0.20$.

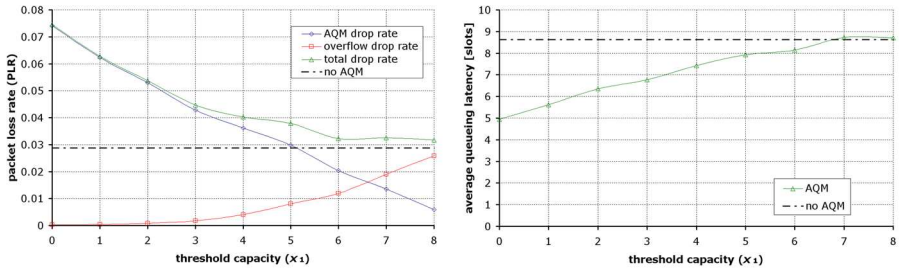

Fig. 5. The PLR and average latency of a 10-packet buffer with varying levels of threshold capacity, x_1 ($y_2 = 0.20$).

The results in Fig. 5 resemble the ones in Fig. 4. As x_1 moves down, the total AQM drop probability increases, suppressing the overflow drop rate, but making a contribution to the total PLR. The latency is reduced by a value corresponding to the rising PLR. When we attempt to compare the two scenarios, trying to evaluate which is a better method to control the effect of the AQM, we see that for a given PLR – the corresponding latency is equal in both scenarios, so there is no conclusively better method.

5 Conclusions and Future Work

In this paper we extend previous work on the design of a scalable optical packet buffer architecture by adding AQM capabilities. Several versions of AQM-RED are used in contemporary electronic internet routers to complement TCP and

overcome some of its inherent problems. It is expected that the implementation of AQM will be required in the optical routers in the future internet core.

The presented buffer architecture, which is scalable, extensible and transparent, also lends itself to a simple implementation of AQM. This implementation was presented in this paper and was validated through extensive simulations. The design of the AQM dropping-probability function determines the performance of the AQM system. In this paper we present a simple function and explore its parameters using simulations. The simulations reveal that while the suggested AQM scheme does reduce latency, it does it at the expense of increasing the PLR.

In future work, the optical packet buffer with the suggested AQM scheme and other schemes will be simulated under a large number of multiplexed TCP streams. This complex modeling of the actual traffic and environment will facilitate the development of AQM schemes which are appropriate for the future optical routers in the internet core.

References

1. C.-S. Chang, Y.-T. Chen, and D.-S. Lee. Constructions of optical FIFO queues. *IEEE Trans. Inform. Theory*, 52(6):2838–2843, June 2006.

2. I. Chlamtac et al. CORD: Contention resolution by delay lines. *IEEE J. Select. Areas Commun.*, 14(5):1014–1029, June 1996.

3. M. Enachescu, Y. Ganjali, A. Goel, N. McKeown, and T. Roughgarden. Part iii: Routers with very small buffers. *SIGCOMM Comput. Commun. Rev.*, 35(3):83–90, 2005.

4. C. Hollot, V. Misra, D. Towsley, and W.-B. Gong. On designing improved controllers for AQM routers supporting TCP flows. In *20th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2001)*, volume 3, pages 1726–1734, Apr. 2001.

5. D. K. Hunter and I. Andonovic. Optical contention resolution and buffering module for ATM networks. *Electronic Letters*, 29(3):280–281, Feb. 1993.

6. C. Jin, D. X. Wei, and S. H. Low. Fast TCP: Motivation, architecture, algorithms, performance. In *20th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2001)*, volume 4, pages 2490–2501, Mar. 2004.

7. OMNeT++ discrete event simulation system. available online at http://www.omnetpp.org/.

8. A. Shacham, B. A. Small, and K. Bergman. A novel optical buffer architecture for optical packet switching routers. In *European Conference on Optical Communications (ECOC '06)*, Sept. 2006.

9. B. A. Small, A. Shacham, and K. Bergman. A modular, scalable, extensible, and transparent optical packet buffer. *J. Lightwave Technol.*, 25(4), Apr. 2007.