Experimental Demonstration of Network Congestion Control with a Programmable Optical Packet Injection Buffer

Howard Wang, Caroline P. Lai, Assaf Shacham, and Keren Bergman
Department of Electrical Engineering, Columbia University, New York, NY 10027; howard@ee.columbia.edu

Abstract: **We experimentally demonstrate active queue management for network congestion control using a programmable optical packet buffer architecture. Optical packets with 10 Gb/s payload are processed transparently and propagate error-free with bit-error rate of $< 10^{-12}$.**

Introduction

Packet buffering represents a significant challenge in the implementation of all-optical routers for use in next generation optical networks. Existing electronic routers employ large packet buffers to store millions of packets, yielding the efficient use of expensive long-haul links. Although all-optical routers have the potential to realize routers with greater capacity and lower power than their electronic counterparts, large optical buffers remain impractical. Recent studies have indicated that buffer sizes can be dramatically reduced to the capacity of 20 packets by sacrificing a part of the link utilization [1], [2]. Optical packet buffers of this capacity are thus feasible, and a novel modular and scalable optical packet buffer architecture has previously been presented [3], [4].

Further, practical packet-switched buffers used in contemporary optical networks should be able to employ advanced congestion control algorithms, such as active queue management (AQM). By incorporating simple modifications into the programmable buffer architecture, AQM can be easily implemented, enhancing the overall router performance. This exemplifies the buffer's flexibility and reconfigurability. Previously, a simulation-based investigation has been performed [5]. Here, we experimentally demonstrate AQM implemented for network congestion control within the programmable buffer architecture. Optical packets containing 10 Gb/s payloads are processed transparently and shown to emerge error-free (bit-error rate $< 10^{-12}$) from the buffer.

Buffer Architecture

The programmable optical packet buffer architecture is composed of a cascade of identical building-block modules (Fig. 1). Each building-block has two input and two output ports, as well as a fiber delay line (FDL) to store a single packet. Two ports (*Up_or_Buffer_In* and *Up_Out*) connect each module to the next one in the cascade, while two ports (*Down_In* and *Down_Out*) connect the module with the previous module. In the root module, the *Down_In* and *Down_Out* ports serve as the buffer's input and output ports, respectively. Electronic *Read* signals are transmitted between modules using electronic cables.
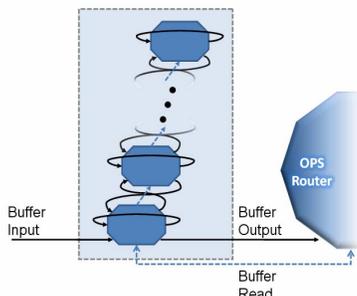

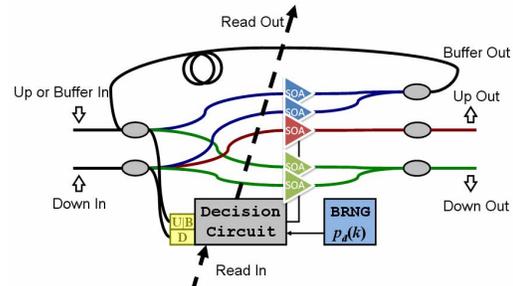**Fig. 1**: Cascaded buffer architecture

Fig. 2: Buffer building block module with five SOAs

Writing packets to the buffer occurs implicitly and in a time-slotted manner; optical packets simply arrive at the root module's *Down_In* port and are injected into the buffer. If the internal FDL is empty, the packet is stored locally; otherwise, the packet is routed to the *Up_Out* port to be stored in the next module. Each packet is similarly forwarded up the cascade and is stored in the first empty module it encounters. In this way, the buffer maintains the correct packet sequence while implementing queue functionality. Simple changes to the buffer's electronic decision circuit also allows for stack functionality [4].

The read process is independent of the write process. If a *Read* signal is received, the locally stored packet propagates from the *Down_Out* port and a *Read* signal is transmitted to the next module to retrieve the next packet. With each *Read* signal, the packets in the chain move one module closer to the root.

The buffer architecture does not require central management, since all the modules follow the same rules; the buffer is also scalable, as increasing buffer capacity simply involves connecting additional modules at the end of the cascade. Further, packet dropping is executed cleanly by routing packets to the *Up_Out* port of the last module.

Programmability

A key feature of the optical packet buffer is its programmable reconfigurability; the buffer's ability to employ active network congestion control schemes, such as AQM, attests to its flexibility and versatility. In a typical AQM scheme known as random early detect (RED), packets are deliberately dropped even before the buffers reach overflow and an early congestion notification (ECN) signal is transmitted to the TCP terminals and network endpoints. AQM also improves the packet loss rate by reducing the buffer's queueing latency.

The programmable buffer architecture effortlessly facilitates the implementation of AQM. The building block module's decision circuitry can hence be modified to incorporate a Bernoulli random number generator (BNRG); each module is programmed with a different Bernoulli parameter, which is the probability that a buffered packet will be dropped. The packet-dropping probability function $p_d(k)$ is a monotonically non-decreasing function that is related to the packet's position in the queue. In the experimental demonstration, the BRNG is used to generate a probabilistic electronic *Drop* signal, indicating that the module should drop its buffered packet.

The basic buffer module is realized with an SOA-based functional subset of a 3x3 optical switch (Fig. 2). While previous

implementations utilized seven SOA switching gates to perform the necessary routing operations [3], [4], we have reduced the number of SOAs in the current implementation without loss of functionality. At the two inputs of each building block module, the packet's power is split and a portion is directed to low-speed p-i-n optical receivers with integrated transimpedance amplifiers (TIAs), which serve to detect the envelope of each high speed packet. Using the packets' envelopes and the *Drop* and *Read* signals, the electronic decision circuit then sets the SOA-based switch; the decision rule used by the electronic circuitry is represented by the truth table in Table 1. The four left columns *Dr, R, UB, D* represent the *Drop* signal for implementing AQM, the *Read* signal, and the two module inputs (*Up_or_Buffer_In* and *Down_In*), respectively. The next five columns indicate the switching states of the SOAs (e.g. UB2D represents *Up_or_Buffer_In* to *Down_Out*), and the last column shows the situations in which a *Read* signal is sent to the following module.

Dr	R	UB	D	UB2B	UB2D	D2U	D2B	D2D	ER
0	0	0	0						
0	0	0	1				1		
0	0	1	0	1					
0	0	1	1	1		1			
0	1	0	0						
0	1	0	1				1		
0	1	1	0		1				1
0	1	1	1		1	1			1
1	0	0	0						
1	0	0	1						1
1	0	1	0						1
1	0	1	1			1			1
1	1	0	0						
1	1	0	1					1	
1	1	1	0		1				1
1	1	1	1		1	1			1

Table 1: Truth table for buffer with AQM

Experimental Demonstration

We demonstrate the operation of the programmable optical packet buffer modeling the functionality of an AQM network congestion control scheme. A two-stage experimental prototype of the proposed managed packet buffer is built. The decision logic is synthesized in a Xilinx complex programmable logic device (CPLD), providing node-to-node granularity and application specific buffer customization. Each building block module consists of the aforementioned CPLD, five SOA switching elements, and two 155 Mb/s p-i-n photodetectors; no optical filters are necessary.

Using SOAs as the switching elements, packet power lost through optical couplers and other sources of insertion loss are compensated. In this way, each SOA hop contributes no net gain or loss, and thus packet longevity is maintained.

In this demonstration, the implemented system supports 115.2 ns timeslots each containing 99.2 ns duration packets with 10 Gb/s modulated data at 1551.64 nm. A 30-bit 312.5 MHz label, modulated separately at 1538.47 nm, facilitates packet identification at the edges of the buffer. Although the experimental packets are comprised of only two wavelengths, one payload and one label, the SOA's broad gain bandwidth can be straightforwardly exploited to allow the buffer to support multiple-wavelength payloads. *Read* pulses are generated to eject the packets from the buffer for analysis.

Fig. 3 depicts the input waveforms and resultant output waveforms for the programmable packet injection buffer. Packets A and B are initially stored in the buffer during the first two timeslots. During the third timeslot, Packet C appears at the buffer's input while a *Drop* signal is generated and delivered to the second buffer module; the buffered packet (Packet B) is subsequently dropped as a result of AQM. A *Read* signal is then

sent downstream; in a larger implementation, the *Read* signal would be received by a third module. In our system, an additional FDL is connected between the *Up_or_Buffer_In* and *Up_Out* ports of the second module to model the functional behavior of a third module. During the fourth timeslot, Packet D enters the buffer; simultaneously, a *Read* signal is generated and the remaining packets are read out one at a time to confirm the expected performance of the buffer. The output shows Packets A, C, and D, which is the result of the architecture's queue functionality with the AQM drop of Packet B.
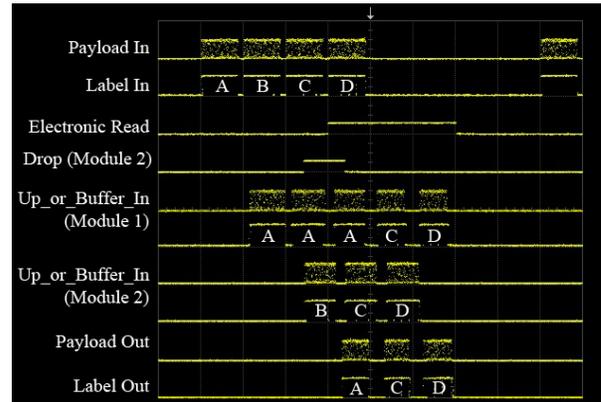

Fig. 3: Waveforms representing functional demonstration of AQM optical buffer operation
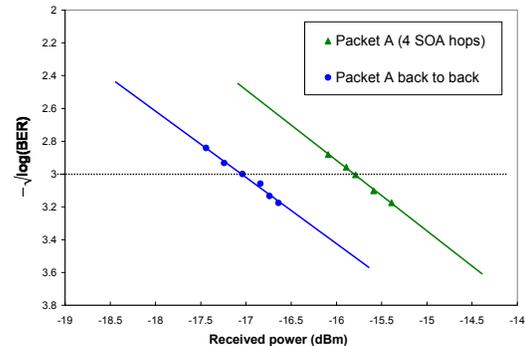

Fig. 4: Power penalty curves for Packet A

All packets received at the output of the buffer are shown to be correctly routed and error-free (bit-error rate $< 10^{-12}$). The longest stored packet, Packet A, is buffered for 921.6 ns. Sensitivity curves indicate that Packet A experiences a power penalty of 1.24 dB (Fig. 4), resulting in an average power penalty of approximately 0.3 dB per SOA hop. This result is consistent with previous findings [4].

Conclusion

The flexibility and feasibility of active network congestion management using a programmable optical packet buffer architecture is presented. Functional verification of a two-stage optical packet buffer modeling AQM is performed and error-free buffering of 10 Gb/s packets is demonstrated for up to 921.6 ns.

References
[1] Appenzeller *et al.* ACM SIGCOMM'04
[2] Enachescu *et al.* IEEE INFOCOM'06
[3] Shacham *et al.* ECOC'06
[4] Small *et al.* JLT, 25 (2007), pp. 978-985
[5] Shacham *et al.* ONDM'07