

Appears in *IEEE T. on Comm.*, v. 47, n. 8, pp. 1792-1795, Dec. 1999.

# **A High Performance Two-Stage Packet Switch Architecture**

**Timothy X Brown<sup>1</sup>**

Dept. of Electrical and Computer Engineering  
University of Colorado, Boulder, CO 80309-0530  
timxb@colorado.edu

## **Abstract**

This paper contributes a distributed packet controller which reduces queuing to a single stage in two-stage packet switches. Software and neural network based controllers are described. Simulations under a range of traffic conditions for a 1024x1024 switch size shows the simplest architecture has the best performance.

Keywords: Packet Switching, Asynchronous transfer mode, Neural Networks

---

1. This work was completed under NSF CAREER Award NCR-9624791.

## 1. Packet Switching

In an  $N \times N$  ATM switch, fixed size packets arrive in periodic *time slots* at one of the  $N$  input ports destined for one of the  $N$  output ports. Because multiple packets can be destined for the same output, packets must queue somewhere in the switch to be sent in a later time slot (so-called *output blocking*). Because of speed and complexity constraints, large ATM switches are built from stages of smaller switch modules as in Figure 1a. Blocked packets can queue at several points in the switching network with different levels of coordination ranging from no coordination between switch stages to various back pressure mechanisms [1]. This paper develops a distributed packet controller that extends a single-stage controller in a prior work [2] and reduces queueing to a single stage. Simulated performance under a range of traffic conditions for a  $1024 \times 1024$  switch size shows the simplest architecture has the best wait time and buffer size performance.

### 1.1 Input vs. Output Switch Modules

We focus on non-blocking switch modules that buffer blocked packets at output or input ports. The output-buffered switch has maximal throughput and the shortest queues [3], but it is also more expensive as the switch must operate at  $N$ -times the input port speed to ensure that every packet can reach its output. Once at the correct output, packets are simply sent, in turn, every time slot. Shared output buffers [4], although they have no effect on the mean performance, reduce the buffer size needed to absorb queue-size variations and are used in this paper.

The input buffered switches in this paper use separate input queues that can send any waiting packet. This avoids head-of-line blocking and enables maximal throughput [2,3]. Unlike an output-buffered switch, a *packet arbitrator* (PA) must decide which packets to send in every time slot. Given a set of input-queued packets, the PA chooses the largest subset so that at most one packet is sent per input queue, and at most one packet arrives per output port.

### 1.2 Software and Neural Network Packet Arbitrators

The packet arbitration problem reduces to the cardinality graph matching problem which has a polynomial-time algorithm solution [5]. This  $O(N^{2.5})$  algorithm readily computes the largest

packet subset in a software simulation although it is unlikely to be fast enough for large high-speed switches to make their selection within a time slot. Also, it does not apply to switches with internal blocking such as banyan networks. For this reason, we consider a neural network PA.

A neural network PA was presented in [2] and the reader is referred to there for details. The  $N \times N$  neuron matrix solution is governed by the differential equations:

$$\frac{du_{ij}}{dt} = I_{ij} - u_{ij} - \sum_{i' \neq j} g(u_{i'j}) - \sum_{j' \neq i} g(u_{ij'}). \quad (1)$$

where for neuron  $ij$ ,  $u_{ij}$  is its state variable,  $I_{ij} < 1.0$  is a constant external input, and  $g(x) \in [0, 1]$  is any of the usual neuron sigmoid functions (e.g.  $g(x) = 1/(1+e^{-4\gamma x})$ ). If  $\gamma = g'(0) > 2$ , (1) approaches a stable equilibrium where—if  $I_{ij} = 0.5$  when a packet at input  $i$  is waiting for output  $j$ ,  $I_{ij} = -1$  otherwise—the set  $\{ij \mid u_{ij} > 0\}$  always corresponds to a valid packet subset [2]. In simulation, the  $I_{ij}$  are set to match the queue state and (1) is numerically integrated using a fifth-order adaptive step size algorithm [6]. To speed the simulation, neurons with  $I_{ij} = -1$  are locked at  $g(u_{ij}) = 0$ . To avoid numerical instabilities the  $u_{ij}$  at  $t = 0$  is a uniform  $[-0.001, +0.001]$ , and  $I_{ij}$  has a uniform  $[-0.001, +0.001]$  added. Setting  $\gamma = N/2$  gives the best performance.

Prior work indicates that the switch fabric for a  $32 \times 32$  622Mbps non-blocking switch or a  $40 \times 40$  neuron neural network can fit on a single VLSI chip [7][8]. Circuits similar to (1) have been built with less than  $1 \mu\text{sec}$  decision time [9]. For signalling, every time slot, a queue sends an  $N$ -bit message to the PA with the queue state, and the PA sends back a  $\log_2 N$  bit message with the selected packet sub-queue [2]. This indicates the input buffered switch has a compact and fast implementation compared to the more complex output buffered switch [4].

## 2. Large Switch Architectures

A common way to build a large  $N^2 \times N^2$  switch is to use two stages of smaller  $N \times N$  switches as in Figure 1a. While straight forward, this may not be optimal. Packets buffer in two stages which, in general, doubles the delay. We consider three versions of the architecture in Figure 1a.

### 2.1 2-Stages of Output Buffered Shared Memory Switches

This is the direct two-stage implementation using the highest performance  $N \times N$  switch. No PA is

necessary. This is the baseline for comparison with other switches and is denoted **OO**.

## **2.2 1 Stage of Output and 1 Stage of Input Buffered Switches with Packet Arbitrator**

Output buffered shared memory switches comprise the first stage. From a second stage switch's perspective, the output buffers of the first stage appear as input buffers. A single stage of buffering is created with stage one buffers logically acting as input buffers to simpler second stage switches (Figure 1b). PAs located in each of the second stage switches controls one output queue in each of the first stage switches. Each queue in the first stage switch must now maintain  $N$  sub-queues so the PA can choose packets destined for any second stage output. If the queues are maintained as linked lists in a single buffer as in [4], then the overhead for the  $N$  linked lists is small. Thus, compared to the OO switch, minor modifications are required in the first stage while the hardware in the second stage is significantly reduced. This switch is denoted **OIA**.

## **2.3 2 Stages of Output Buffered Switches with Packet Arbitrator**

In the OO switch (§ 2.1), the second-stage delay depends on the order that packets are sent from the first-stage (e.g. Figure 2). In the OIA switch (§ 2.2), the PA eliminates second-stage queueing but contention may prevent a non-empty first-stage queue from sending a packet. This section's switch sends the head-of-queue packet from such queues to be buffered in the second stage. As in § 2.1, every non-empty first-stage queue sends one packet while the PA reduces second stage queueing. This switch shows the advantage of packet arbitration and is denoted **OOA**.

## **3. Experiments with a 1024 Input Switch**

The following experiments are designed to compare the switch designs. Absolute switch performance would require a range of switch sizes under more diverse and realistic traffic types. Instead, the designs are compared under different loads for a 1024 switch size (i.e two stages of 32 32x32 switches). This size is an order of magnitude larger than existing switches today, and the 32x32 neurons in each neural PA is well beyond the "toy problem" size. To lower bound the performance, we look at the first stage buffering and wait times of the OO switch. This represents the irreducible queueing and contention at the first to second stage links.

The packet traffic is generated using a Bernoulli process and a burst process with different loads  $\lambda^1$ . The switches are simulated for 11,000 time slots starting from empty queues. For a given load and traffic type the sequence of arrivals is the same for every switch. The first 5,000 time slots of data are discarded so that only steady-state behavior is observed. With 32 switches in each stage, the remaining 6,000 time-slots contain a total of 192,000 observations.

Average packet delay is used as a measure of average performance. To remove details of the switch implementation, the inherent delay (i.e. the delay of a single packet through an empty switch) is subtracted out so that only the queuing delay is measured. In every time slot, the size of every shared buffer is recorded after all arrivals, but before any departures. As a measure of required buffering (an outlier metric), the buffer size exceeded in only  $10^{-3}$  of the time slots is computed directly from the buffer size trace data. Smaller blocking probabilities, while more realistic, require extrapolation techniques that do not illuminate the comparative results further.

Since packet behavior is correlated over time scales ranging up to several hundred time slots, error bars are computed indirectly. For each metric, 6 sub-estimates are computed across successive 1000 time slot intervals which are then assumed independent. Average delay, being a linear combination of the sub-estimates, uses the standard deviation of the sub-measurement average. Buffer size, not being linear, uses the standard deviation of the sub-estimates. The buffer-size sub-estimates are from a smaller sample than the final estimate so this upper bounds the error.

Figure 3 shows results for the three architectures using the software PA. The OO and OOA switches include the total delay/buffering of both stages. Across metrics, the difference between the OO and the lower bound is a factor of 2. For queuing delay, both arbitrated architectures approach the lower bound at high loads. For buffering the simpler OIA architecture approaches the lower bound for low loads while for high loads both arbitrator architectures approach a buffer

1. For the Bernoulli process, in every time slot a packet is generated at an input with probability  $\lambda$ . The destination of the packet is a uniform random destination over all the outputs. For the Burst process, packets arrive in geometrically distributed bursts of at least one packet with mean burst size  $m$ . This is modeled by having a burst start in a time slot with probability  $\lambda/m$ . All packets in a burst arrive in consecutive time slots and have the same destination chosen uniformly. Bursts start immediately or as soon as all prior bursts finish whichever is first. In this paper  $m = 10$ .

size midway between the OO and lower bound. The OIA's superior performance is due to the single stage of buffering. Although the OIA has longer first stage queues, these are the only queues and all queued packets are therefore under the PA's control.

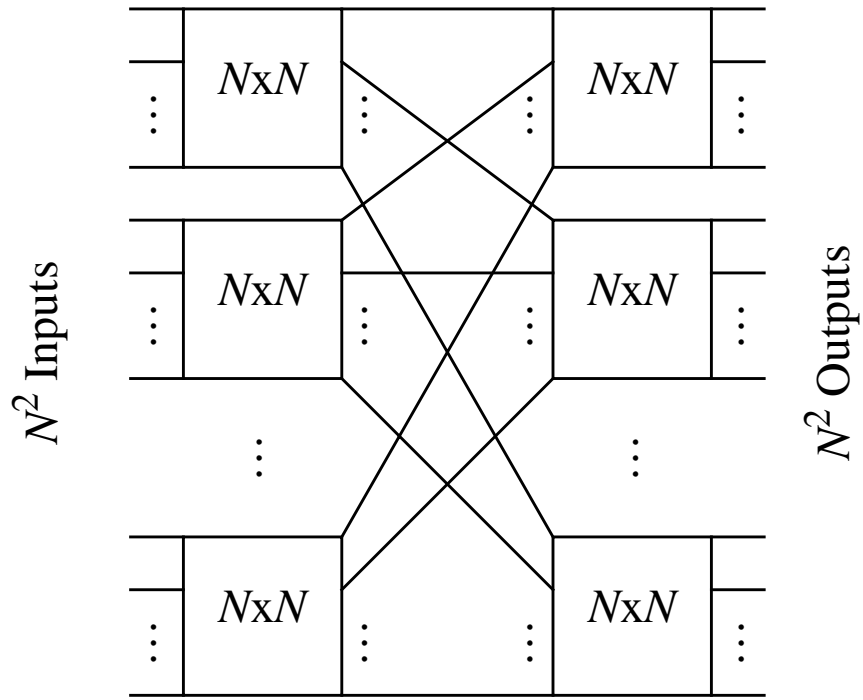
Results using a neural network PA are shown for Bernoulli arrivals in Figure 4. The neural network is guaranteed to provide valid but not necessarily optimal solutions. When applied to the OOA switch there is little difference. The non-optimal solutions of the neural PA are compensated for by always being able to send packets from any non-empty queue so that the number of packets sent is the same with both controllers. In the OIA switch, the non-optimal neural solutions translate directly into longer queues and delay at high loads.

#### 4. Conclusion

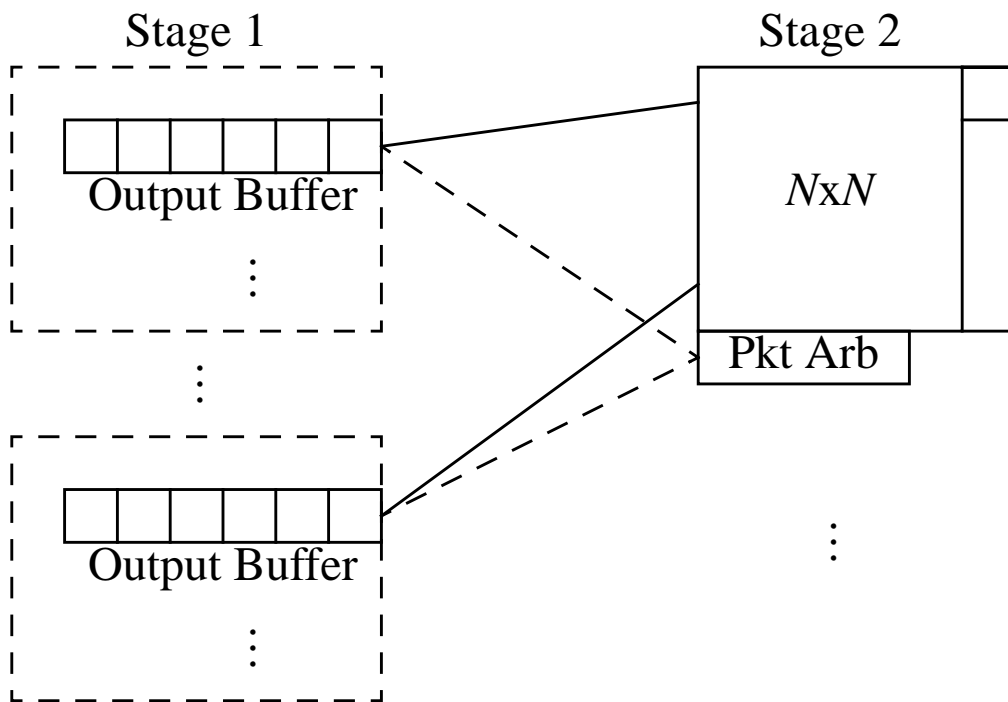
A distributed PA was developed and applied to a large high-speed packet switch. The arbitrated OIA and OOA switches of § 2.2 and 2.3 bring queueing delays to near optimal and required buffer sizes to within a factor of 1.4 of optimal at high loads. This result applies equally across different arrival processes. Surprisingly the best performance is with the less complex OIA switch. Comparing software and neural network PA, the neural network is similar when applied to the OOA switch while performance degrades at high loads when applied to the OIA switch. Even so, the fast neural network decisions could enable high-speed applications. Future work includes: controlling max delay/fairness and simpler internal blocking switches.

#### References

- [1] Pattavina, A., *Switching Theory: Arch. & Perf. in Broadband ATM Net.*, J. Wiley & Sons, West Sussex, 1998.
- [2] Brown, T.X, Lui, K.H., "Neural Network Design of a Banyan Network Controller," *IEEE JSAC*, v. 8, n. 8, pp. 1428–1438, Oct., 1990.
- [3] Karol, M.J., Hluchyj, M.G., Morgan, S.P., "Input vs. Output Queueing on a Space Division Packet Switch," *IEEE Trans. on Comm.*, v. 35, pp. 1347–1356, Dec. 1987.
- [4] Kozaki, T., et al., "32x32 Shared Buffer Type ATM Switch VLSI for B-ISDN," *IEEE JSAC*, v. 8, n. 9, Oct. 1991
- [5] Lawler, E.L., *Combinatorial Optimization: Networks and Matroids*, Holt Rinehart & Winston, NY, 1976, p. 195
- [6] Press, W.H., et al., *Numerical Recipes in C*, 2nd ed. Cambridge U. Press. 1992, pp. 710–722.
- [7] Marcus, W.S., "A CMOS Batcher and Banyan Chip Set for B-ISDN Packet Switching," *IEEE J. of Solid-State Circuits*, v. 25, n. 6, Dec. 1990, pp. 1426–1432.
- [8] Eberhardt, S.P., et al., "Competitive Neural Architecture for Hardware Solution to the Assignment Problem," *Neural Networks*, v. 4, pp. 431–442, 1991.
- [9] Choi, J., Sheu, B.J., "A High-Precision VLSI Winner-Take-all Circuit for Self Organizing Neural Networks," *IEEE J. of Solid-State Circuits*, v. 28, n. 5, pp. 576–584, May 1993.



(a)



(b)

Figure 1: Building a  $N^2 \times N^2$  switch out of  $2N$   $N \times N$  switches (a). Output Buffers of Stage 1 Switches are Input Buffers to a Stage 2 Switch (b).

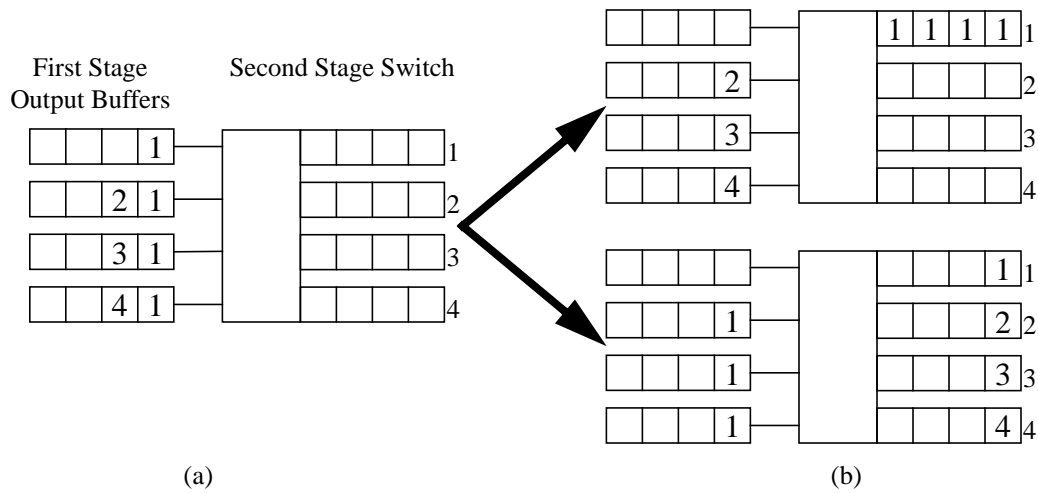
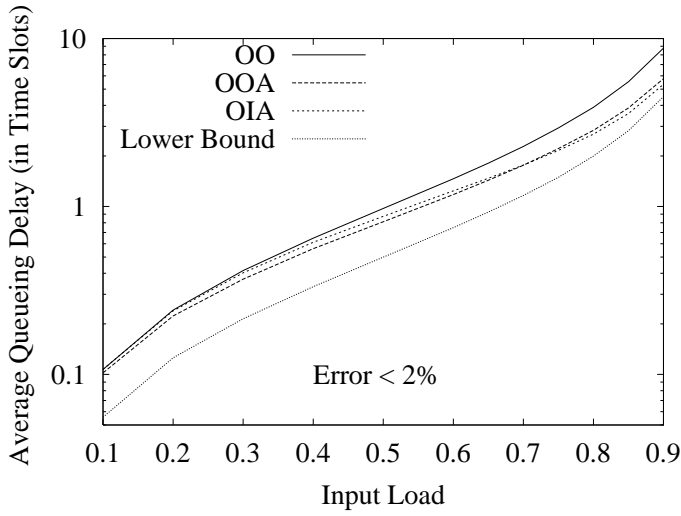
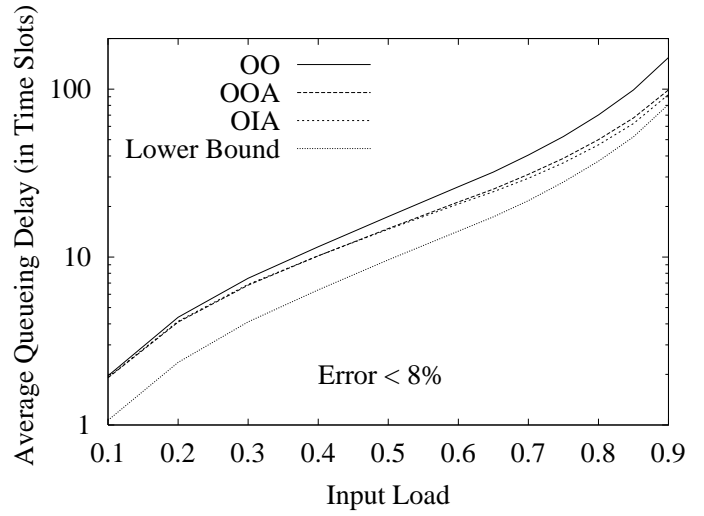


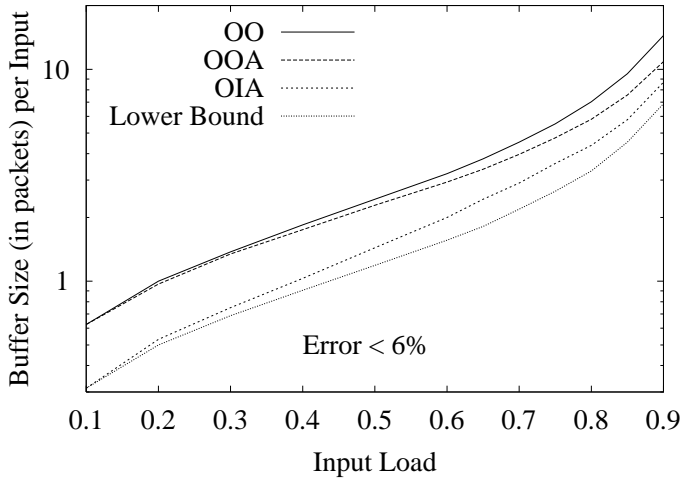
Figure 2: For a 16x16 switch composed of two stages of 4x4 output buffered switches, (a) shows the four first-stage buffers serving one of the second stage switches. The first stage buffers both send 4 packets in (b), but the top set leads to congestion in the second layer, while the bottom set does not.



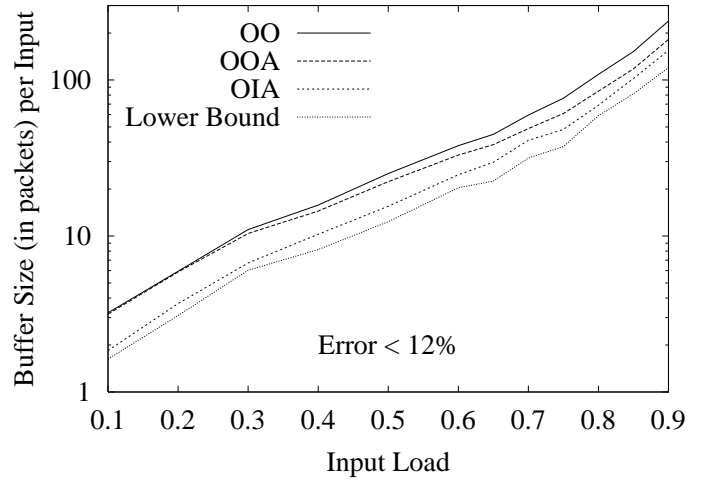
(a) Queueing Delay: Bernoulli Arrival Data



(b) Queueing Delay: Bursty Arrival Data

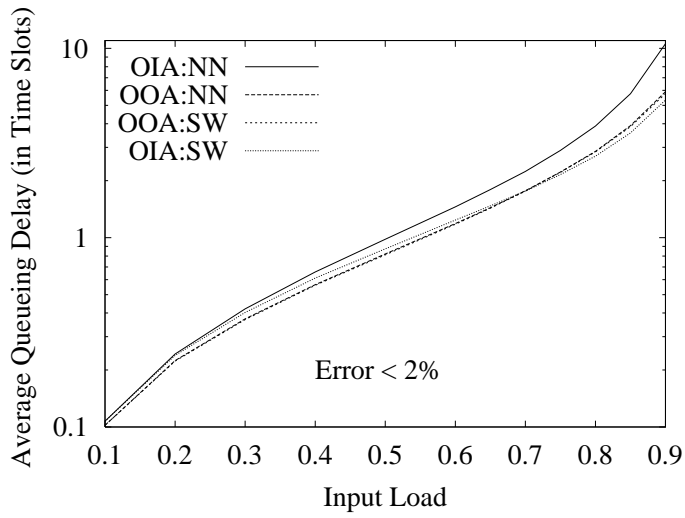


(c) Buffer size for  $10^{-3}$  loss: Bernoulli Arrival Data

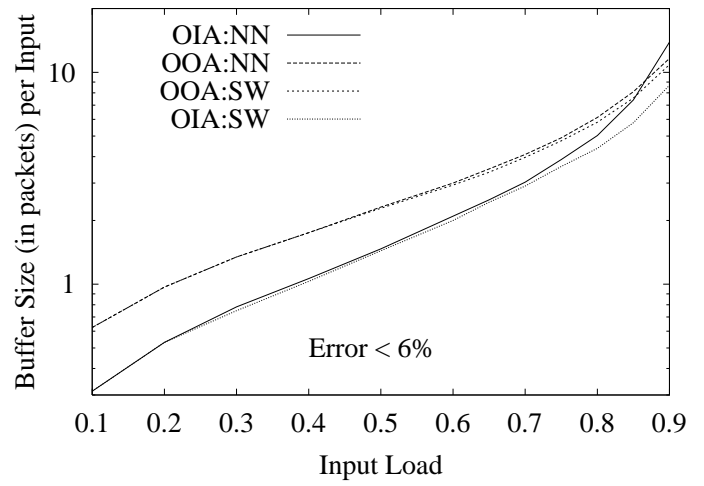


(d) Buffer size for  $10^{-3}$  loss: Bursty Arrival Data

Figure 3: Performance of different switch configurations with software packet arbitrator.



(a) Queueing Delay: Bernoulli Arrival Data



(b) Buffer size for  $10^{-3}$  loss: Bernoulli Arrival Data

Figure 4: Comparing Software (SW) and Neural Network (NN) Packet Arbitrators: Queueing Delay and Buffer Size with Bernoulli Arrivals.