The Inherent Queuing Delay of Parallel Packet Switches*

Hagit Attiya† and David Hay
Department of Computer Science
Technion

ABSTRACT

The *parallel packet switch (PPS)* is extensively used as the core of contemporary commercial switches. This paper investigates the inherent queuing delay and delay jitter introduced by the PPS's demultiplexing algorithm, relative to an optimal work-conserving switch.

We show that the inherent queuing delay and delay jitter of a symmetric and fault-tolerant $N \times N$ PPS, where every demultiplexing algorithm dispatches cells to all the middle-stage switches is $\Omega(N)$, if there are no buffers in the PPS input-ports. If the demultiplexing algorithms dispatch cells only to part of the middle-stage switches, the queuing delay and delay jitter are $\Omega(N/S)$, where S is the PPS speedup. These lower bounds hold unless the demultiplexing algorithm has full and immediate knowledge of the switch status. When the PPS has buffers in its input-ports, an $\Omega(N/S)$ lower bound holds if the demultiplexing algorithm uses only local information, or the input buffers are small relative to the time an input-port needs to learn the switch global information.

Categories and Subject Descriptors

B.4.3 [**Hardware**]: INPUT/OUTPUT AND DATA COMMUNICATIONS—*Interconnections*; C.2.6 [**Computer Systems Organization**]: COMPUTER-COMMUNICATION NETWORKS—*Internetworking*

General Terms

Algorithms, Performance, Theory

Keywords

Inverse multiplexing, Leaky-bucket traffic, Packet switching, Clos networks, Queuing delay, Delay jitter, Load balancing

1. INTRODUCTION

An $N \times N$ *packet switch* routes packets arriving on N input-ports at rate R to N output-ports working at rate R.

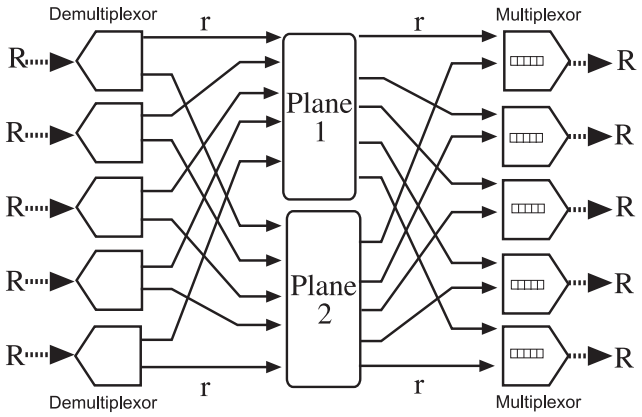

Figure 1: A 5×5 PPS with 2 planes in its center stage, without buffers in the input-ports

Packets are stored and transmitted in the switch as fixed-size *cells*; fragmentation and reassembly are done outside of the switch. Cells arrive to the switch as a collection of *flows* from one input port to the same output-port; the switch should preserve the order of cells within a flow and not drop cells. Since there may be conflicts between different flows, certain amount of buffering within the switch may be needed. The location of the buffers, their sizes, and their management, depend on the specific architecture of the switch.

Switching cells in parallel is a natural approach to build switches with very high external line rate, and with large number of ports. A prime example of this approach is the *parallel packet switch* (in short, PPS) model [3], which is based on the *inverse multiplexing for ATM (IMA)* technology [1]. A PPS is a three-stage Clos network, with $K < N$ switches in its center stage, also called *planes*. Each plane is an $N \times N$ switch operating at rate $r < R$, and is connected to all the input-ports on one side, and to all the output-ports on the other side (see Figure 1). The *speedup* S of the switch is the ratio of the aggregate capacity of the internal traffic lines, connected to an input- or output-port, to the capacity of its external line, namely, $S = \frac{Kr}{R}$. An *input-buffered* PPS [4] has finite buffers in its input-ports in addition to buffers in its output-ports.

A key issue in the design of a PPS is its *demultiplexing algorithm*, balancing the load of switching operations among the middle-stage switches, and by that utilizing the parallel capabilities of the switch.

*A full version of the paper is available as Technical Report CS-2004-02, Department of Computer Science, The Technion.

†Part of the work was performed while this author was at Dune Networks.

SPAA'04, June 27–30, 2004, Barcelona, Spain.
ACM 1-58113-840-7/04/0006.

Demultiplexing algorithms can be classified according to the amount and type of information they use. The strongest type of demultiplexing algorithms are *centralized algorithms*, using global information about the status of the switch. Unfortunately, these algorithms must operate in a speed proportional to the aggregate incoming flows rate, and therefore, they are impractical. At the other extreme, *fully-distributed demultiplexing algorithms* rely only on the local information in the input-port, and are common in contemporary switches. A middle ground is what we call *u Real-time distributed (u-RT) demultiplexing algorithms* , in which a demultiplexing decision is based on the local information and global information older than u time slots.

2. EVALUATING PPS PERFORMANCE

Switch architectures are evaluated by their ability to provide different QoS guarantees. Important figures are the queuing delay of cells (i.e., delay resulting from queuing cells within the switch) and the switch's throughput. Another interesting performance number is the *per-flow delay jitter* (or *cell delay variation*), namely, the maximal difference in queuing delay between two cells in the same flow.

The performance of a PPS is measured by comparison to an ideal *work-conserving (greedy)* switch, operating at rate R. A work-conserving switch guarantees that if a cell is pending for output port j at time-slot[1] t, then some cell leaves from output-port j at time-slot t. This property prevents an output-port from being idle unnecessarily, and by that, maximizes the switch throughput and minimizes its average cell delay.

The switch used for the comparison is called a *reference* switch, and it receives exactly the same stream of flows as the PPS. We assume that this reference switch minimizes the queuing delay of cells (or minimizes the delay jitter, in case we measure the relative delay jitter). A primary candidate for a reference switch is an output-queued switch operating at rate R. This is the reason this comparison is sometimes referred to as the ability of the PPS to mimic an output-queued switch [3, 4].

The *relative queuing delay* neglects the different propagation delays over the PPS and the reference switch, or the different number of stages. It captures the influence of the parallelism of the PPS on the performance of the switch, depending on the different demultiplexing algorithms.

3. OUR RESULTS

This paper proves lower bounds on the relative queuing delay and the relative delay jitter of the PPS when the switch is not flooded. We employ (R, B) *leaky-bucket constrained flows* to restrict the arrival of cells to the switch: In every time interval of length τ, the number of cells arriving to the switch and sharing the same input-port or the same output-port is bounded by $\tau R + B$, where B is a fixed *burstiness* factor [2].

A PPS without buffers at the input-ports with fully distributed demultiplexing algorithm induces the highest relative queuing delay and relative delay jitter. If some plane is utilized by all the demultiplexors, we prove a lower bound of $\left(\frac{R}{r} - 1\right) N$ time slots on the relative queuing delay and relative delay jitter. Even in the unrealistic and failure-prone case where the demultiplexing algorithm statically partitions the planes among the demultiplexors, the relative queuing delay and relative delay jitter are at least $\left(\frac{R}{r} - 1\right) \frac{N}{S}$ time-slots. Both lower bounds employ leaky-bucket flows with no bursts.

A bufferless PPS with u-RT demultiplexing algorithm (for any u) has relative queuing delay and relative delay jitter of at least $\left(1 - \overline{u}\frac{r}{R}\right) \frac{\overline{u}N}{S}$ time-slots, under leaky-bucket flows with burstiness factor of $\overline{u}^2 \frac{N}{K} - \overline{u}$, where $\overline{u} = \min\{u, \frac{1}{2}\frac{R}{r}\}$. In contrast, there is a bufferless PPS with centralized demultiplexing algorithm with zero relative queuing delay, provided that the switch has speedup $S \geq 2$ [3].

An *input-buffered PPS* can support more elaborate demultiplexing algorithms, since an arriving cell can either be transmitted to one of the middle stage switches, or be kept in the input-buffer. Under a u-RT demultiplexing algorithm, a switch with speedup $S \geq 2$ and input-buffers larger than u, can employ a centralized algorithm (e.g., [3]). In contrast, a fully-distributed demultiplexing algorithm introduces relative queuing delay and relative delay jitter of at least $\left(1 - \frac{r}{R}\right) \frac{N}{S}$ time-slots, for any buffer size under leaky-bucket flows with no bursts.

The lower bounds are obtained when cells that are supposed to leave the reference switch one after the other, are concentrated in a single plane. This concentration scenario is described by the next lemma:

LEMMA 1. *Assume output-port j's buffer is empty at time t, and that m cells destined for the same output-port j arrive to the switch during time-interval $[t, t+s)$, out of them $c \leq m$ are sent through the same plane. Assume also that the incoming traffic is (R, B) leaky-bucket, and no cells destined for j arrive to the switch during time interval $[t+s, t+m)$. Then:*
(1) The relative queuing delay of the PPS is at least $c\frac{R}{r} - (s + B)$ time-slots.
(2) The relative delay jitter of the PPS is at least $c\frac{R}{r} - (s+B)$ time-slots.

Our lower bound results show that the PPS architecture does not scale with increasing number of external ports. This is significant since great effort is currently invested in building switches with a large number of ports (where $N = 512$ or even 1024). Note that large relative queuing delays usually imply that the buffer sizes at the middle-stage switches or at the external ports should be large as well, so that the cells can be queued.

4. REFERENCES

[1] The ATM Forum. *Inverse Multiplexing for ATM (IMA) specification*, March 1999.

[2] A. Charny. *Providing QoS guarantees in input buffered crossbar switches with speedup*. PhD thesis, MIT, September 1998.

[3] S. Iyer, A. Awadallah, and N. McKeown. Analysis of a packet switch with memories running at slower than the line rate. *INFOCOM* 2000, pp. 529–537.

[4] S. Iyer and N. McKeown. Making parallel packet switches practical. *INFOCOM* 2001, pp. 1680–1687.

[1] A *time-slot* is the time required to transmit a cell at rate R.