Available online at www.sciencedirect.com
SCIENCE DIRECT®
Theoretical Computer Science 324 (2004) 161 – 182
Theoretical Computer Science
www.elsevier.com/locate/tcs

Harmonic buffer management policy for shared memory switches☆

Alexander Kesselman*, Yishay Mansour
School of Computer Science, Tel-Aviv University, Tel-Aviv 69978, Israel

Abstract

In this paper we consider shared-memory switches. We introduce a novel general non-preemptive buffer management scheme, which considers the queues ordered by their size. We propose a new scheduling policy, based on our general scheme, which we call the Harmonic policy. We analyze the performance of the Harmonic policy by means of competitive analysis and demonstrate that its throughput competitive ratio is at most $\ln(N)+2$, where N is the number of output ports. We also present a lower bound of $\Omega(\log N/\log\log N)$ on the performance of any online deterministic policy. Our simulations also show that the Harmonic policy achieves high throughput and easily adapts to changing load conditions.

Keywords: Competitive analysis; Shared memory switches; Non-preemptive policies; Buffer management

1. Introduction

The Internet is built around a large variety of transmission and switching systems and the information transfer is aggregated into packets, which are individually forwarded and switched toward their destinations. The main tasks of a router are to receive a packet from an input port, find its destination port using the routing table, transfer the packet to the output port via the switch fabric, and finally transmit it on the output link. If a burst of packets destined to the same output port arrives, not all packets can be transmitted on the fly and thus some of them need to be buffered. A critical aspect of the switch architecture is placement of buffers. In the output queuing (OQ) architecture, packets arriving from input ports immediately cross the switching fabric,

☆ A preliminary version of this work appeared in INFOCOM 2002.
* Corresponding author.
E-mail addresses: alx@cs.tau.ac.il (A. Kesselman), mansour@cs.tau.ac.il (Y. Mansour).

doi:10.1016/j.tcs.2004.05.014

and join a queue at the switch output port. It is well-known that the OQ architecture allows one to maximize throughput and control packet latency accurately. The shared memory (SM) switch architecture is an OQ architecture in which output queues are dynamically allocated from a shared memory pool.

The main benefit of SM switches is their flexibility. At the one extreme, they can be used for complete sharing, where the memory is used as one buffer to serve all the output ports. The main benefit of complete sharing is that it takes advantage of statistical multiplexing. However, complete sharing may perform poorly under overload conditions [10]. The problem is that a single output port can take over most of the memory, preventing packets destined for other output ports from gaining access, which causes the total switch throughput to drop. At the other extreme, shared memory can also simulate complete partition, i.e. a dedicated buffer for each output port. In this case the flow of each output port is fully protected from the flows of other output ports. This comes at the cost of underutilization of resources, thus risking higher loss during bursts. The flexibility of shared memory allows one to specify many other policies on the spectrum between complete sharing and complete partition. The goal of a buffer management policy is to achieve as much of the benefits of both schemes while suffering as little as possible from their weaknesses.

The buffer management policies are traditionally classified into two categories: preemptive and non-preemptive, according to whether they utilize the preempt action in which a packet that has been accepted into the buffer can be later dropped in order to free space for newly arriving packets. The tradeoff here is between ease of implementation and hardware (where non-preemptive policies have an advantage) and higher performance (where preemptive policies have an advantage). Both types of policies have been widely considered in the networking literature. For a good survey of shared-memory buffer management policies the reader can refer to [2].

The main class of non-preemptive scheduling policies are static threshold schemes. Irland [10] considers some simple policies of this type. In sharing with maximum queue lengths (SMXQ) scheme, each output queue has a static bound on its length and a packet is accepted if there is a free space in the buffer and the corresponding bound is not violated. In some schemes, like sharing with a maximum queue and minimum allocation (SMQMA) due to Kamoun and Kleinrock [11], each port always has access to a minimum allocated space. The main problem of the static threshold schemes is that they are not adaptive. When many queues are active and the sum of their thresholds exceeds the buffer capacity, the buffer may fill up completely, even though all queues are obeying the threshold constraints. Thus, some queues can be starved, which leads to underutilization of the switch. On the other hand, when very few queues are active, they are denied access to the idle buffer space beyond the sum of their thresholds. This creates higher packet loss rate for the active queues (see [5]).

Another class of non-preemptive policies includes dynamic threshold schemes. In the Dynamic Threshold (DT) policy due to Choudhury and Hahne [5], the threshold on a queue length at any instant of time is proportional to the current amount of