The Impact of Message Scheduling on a Packet Switching Interconnect Fabric

Ludmila Cherkasova, Vadim Kotov, Tomas Rokicki
Hewlett-Packard Laboratories
1501 Page Mill Road, Palo Alto, CA 94303
e-mail: {cherkasova,kotov,rokicki}@hpl.hp.com

Abstract

The impact of different message scheduling strategies on the performance of a packet-switched network is explored. We present a new scheduling technique, called Alpha scheduling, that can combine the bandwidth-fairness of round robin scheduling with nearly the optimal performance of shortest-first scheduling. We investigate the trade-offs involved in using different routing strategies while intelligently scheduling the packets from various messages for injection into an interconnect.

1 Introduction

Evaluation of interconnect performance generally focuses on fixed-size packet latency as a function of traffic load. To an application, however, it is the latency of the variable-length messages, rather than the individual packets, that is important. We present a new message scheduling technique, called Alpha scheduling, that can combine the bandwidth-fairness of Round Robin scheduling with nearly the optimal performance of shortest-first scheduling under bursty traffic conditions and different routing strategies. Our primary result is that using an appropriate strategy to insert the packets comprising messages into an interconnect can have a tremendous impact on the performance of that interconnect.

First, we present the performance results of Alpha scheduling, assuming the delay of the interconnect is a simple probability distribution. This isolates the effects of Alpha scheduling from its interaction with a particular interconnect. The results of this model indicate that selection of such an injection strategy can increase the effective performance of the “abstract” interconnect by a factor of two or three over naive FIFO or Round Robin packet insertion.

We also validated the Alpha scheduling performance advantages by applying it to the PO2 interconnect as outlined in [CKR94]. The PO2 topology is the same as that of the Mayfly [Davis92]: the elements are combined in a wrapped hexagonal mesh topology to form a low latency, high capacity interconnection fabric for scalable parallel processing systems containing up to hundreds of processing elements (PEs). PO2 supports the transfer of messages which may vary in length but are physically transferred as a series of fixed-length packets.

We investigated the trade-offs involved in using different routing strategies while intelligently scheduling the packets from various messages for injection into the interconnect. In particular we compare deterministic and two forms of adaptive strategies under three different message scheduling algorithms: FIFO, Round Robin and Alpha scheduling. Our simulations indicate that when messages are injected into an interconnect in FIFO manner, adaptive routing can improve the average message latency 1.5 to 2.5 times against the deterministic one. However, for some types of bursty traffic with a high volume of short messages, Alpha scheduling improves the average message latency 2 to 5 times. In addition, it makes the different routing strategies be practically indistinguishable from a performance perspective. For many workloads, intelligent injection not only improves performance more than adaptive routing does, but it also makes deterministic routing perform as well as adaptive routing.

For complete version of the paper see [CKR94].

2 Alpha Scheduling Strategy

We propose a scheduling strategy that lies between FIFO and shortest-first, based on the value of a coefficient. The messages are stored in a priority queue. Three parameters control the ordering of messages in the queue:

- The node parameter \( c \) is a “clock” that starts at zero and increments for each packet inserted into the interconnect through the current node. It is easy to keep this value bounded without changing the scheduling solution.
- The message parameter \( l \) is the number of packets in the message that have not yet been sent. Initially this is just the length of the message. As each packet is sent out, the message priority is decremented by \( \alpha \) to keep the head message priority up to date. Another strategy is to recalculate the head message priority before preempting it during the scan for insertion of a new message.
- The tuning parameter \( \alpha \) controls the balance between fairness and latency minimization.
Messages are inserted into the delivery queue with a priority of 
\[ c + \alpha t. \]

Messages with the lowest priorities get delivered first. A new message inserted into the queue with a priority lower than that of the sending message preempts the sending message.

If \( \alpha = 0 \), then this strategy is simply FIFO.

If \( \alpha = \infty \), then this strategy is simply shortest-packet first; this is optimal for latency.

If \( \alpha = 1 \) or some other finite positive value, then the strategy will not allow any single application to be delayed indefinitely by the other applications, no matter what their message stream looks like. Larger \( \alpha \) provides better average latency; smaller \( \alpha \) provides better fairness.

The Alpha scheduling simulation consists of three main components: a simplified model of the interconnect, an instantiation of the queue and its strategy, and a model to generate messages from a specific traffic pattern. Since we are only interested in the impact of message scheduling, we simplified our model of the interconnect to be a service queue with an average delay of one. This is the default time unit for our simulation. For the probability distribution function, we use the sum of a constant 0.5 plus a negative exponential with an average of 0.5 to reflect the fact that the port has a specific maximum bandwidth, and that the dead time between packets can vary greatly.

For a default traffic distribution, we assume 10% of the messages to be long, 20-packet messages, and the remaining 90% to be from one to five packets in length. The average message length is therefore 4.7 packets. Given a traffic density \( u \) between zero and one, we generate new messages using a negative exponential distribution with an average of 4.7/u.

Our primary simulation results for this model are summarized here:

- The effects of message scheduling increase with traffic load.
- Round Robin and FIFO scheduling can always be outperformed with a judicious selection of the \( \alpha \) parameter. A value of 10 will outperform both Round Robin and FIFO scheduling for traffic loads up to and including 98% of utilization for our traffic load. Other traffic loads show similar results.
- The \( \alpha \) parameter trades long-message latency for short-message latency. Higher \( \alpha \) gives better short-message latency and better average latency; lower \( \alpha \) decreases the worst-case message latency.

The primary effect of increasing \( \alpha \) is to insert more short messages before long messages, thus trading off long message latency for short message latency. We assume that short message latency is extremely important, and that short messages will outnumber long messages significantly. Yet, long message latency is of some importance and should factor into our calculations. Table 1 summarizes how average message latency changes with traffic load for the various strategies.

We validated the Alpha scheduling performance advantages by applying it to the PO2 interconnect. The backpressure flow control mechanism in PO2 changes the interconnect behaviour in a particular way that causes Alpha scheduling to have additional positive performance implications. In general, Alpha scheduling allows short messages to interrupt longer messages in such a way that overall queue waiting time is decreased. Indeed, it is easy to prove that short messages have a latency close to optimal. The trade-off is that the latency of long messages increases slightly. In the presence of the backpressure control mechanism, the latency increase is very small. If a long message is being inserted through a congested region, backpressure will quickly stall the queue, rejecting further packets from that message, so the long message would be delayed anyway. Injecting a few short messages significantly decreases their waiting time, without affecting the overall latency of the long message. Effectively, this automatic fragmentation of long messages by short ones acts to decrease traffic burstiness and randomize the traffic pattern by interleaving "chunks" of long message with short messages, decreasing the congestion caused by a few coinciding long messages. Thus, using Alpha scheduling minimizes the latency of short messages without a significant penalty for interrupting the transfer of long messages.

### References


<table>
<thead>
<tr>
<th>Strategy</th>
<th>Traffic Load</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>50</td>
</tr>
<tr>
<td>Round Robin</td>
<td>8.6</td>
</tr>
<tr>
<td>FIFO</td>
<td>9.6</td>
</tr>
<tr>
<td>( \alpha = 1 )</td>
<td>8.2</td>
</tr>
<tr>
<td>( \alpha = 3 )</td>
<td>7.1</td>
</tr>
<tr>
<td>( \alpha = 10 )</td>
<td>6.8</td>
</tr>
<tr>
<td>( \alpha = 30 )</td>
<td>6.8</td>
</tr>
<tr>
<td>( \alpha = 100 )</td>
<td>6.8</td>
</tr>
<tr>
<td>Shortest First</td>
<td>6.8</td>
</tr>
</tbody>
</table>

Table 1: Overall latency for various injection strategies and workloads.