Improving Optical Data Router Performance through Prime Packet Recycling

Joel W. Gannett and George Clapp
Telcordia Technologies, Inc., 331 Newman Springs Road, Red Bank, New Jersey 07701-5699
{gannett, clapp}@research.telcordia.com

Abstract: Data in the Optical Domain Networks (DOD-N) can suffer high packet drops owing to small optical buffer capacities. We present a technique to reduce packet drops that relies on the relative primeness of two integers.

© 2006 Optical Society of America
OCIS codes: (060.4250) Networks; (060.4510) Optical Communications

1. Introduction

Data in the Optical Domain Networks (DOD-N) seek to achieve superior bandwidth and utility by circumventing optical/electronic/optical (O/E/O) conversions in the data path. The advantages of DOD-N networks include reduced power dissipation in network elements and the avoidance of potential electronic bottlenecks. Unfortunately, current and foreseeable all-optical packet buffer technologies allow for only limited packet storage, which can result in degraded performance for DOD-N routers owing to high packet drop ratios.

We present a method here that uses easy-to-compute packet recycling to improve the drop ratio for a given buffer size. This method applies to certain router architectures and relies on the relative primeness of two integers. It requires for computation nothing more than the integer modulus and integer division operations.

2. Load-Balanced Router Architecture

Our method was designed to solve the packet storage problem in a so-called load-balanced router architecture, which has been described in detail elsewhere [1, 2]. Basically, the load-balanced architecture (Fig. 1) has two stages of router fabric, both of which undergo simple, deterministic cycling.

In the load-balanced architecture, time is divided into slots that are the proper duration for transmitting the network's fixed-size packets across the fabric. In time slot 0, which is the initial time slot, a packet arriving at input port \( j \) is transmitted to middle port \( j \), and a packet leaving middle port \( j \) is transmitted to output port \( j \).

In the next time slot (i.e., time slot 1), input \( j \)'s packet is transmitted to middle port \( j+1 \), while middle port \( j \) transmits its packet to output port \( j+1 \), for \( 0 \leq j \leq N-2 \), where \( N \) is the number of input, output, or middle ports. If \( j =
Recycle?  

Recycle?  

The attractiveness of the load-balanced architecture is the simplicity of its scheduling algorithm. No complex calculations or complex decisions need to be executed to schedule the packet transits across the fabric. Instead, the scheduling is simple, repetitive, and deterministic. This makes it amenable to high-speed DOD-N routing.

3. Packet Recycling

The routing function of the load-balanced router occurs in the middle stage, where a packet must be held in a buffer until a later time slot when the second-stage fabric is configured correctly to send the packet to its proper destination output port. Unlike conventional router architectures that may require that the packet be held for an indefinite period of time before it is scheduled for transmission across the fabric, the load-balanced architecture requires that the packet be held only for a certain definite number of time slots that is known the moment the packet arrives at the middle stage. This holding time is a function of the desired output port, which, in turn, is a function of the eventual destination of the packet. This allows the use of delay-line type packet storage in the middle stage, which is fortunate as indefinite-hold-time storage is difficult to implement all-optically. See Fig. 2, where a delay-line-style middle-stage packet buffer of size \( M = 5 \) is depicted. This buffer could be implemented with five fiber delay loops in series, each providing a delay of one time slot, along with a control mechanism directing which of the five fiber loops an arriving packet must enter. The integer label \( d \) on each buffer stage means that a packet arriving at the beginning of the time slot that is placed into stage \( d \) will be on the verge of exiting the right-hand end of the buffer at the end of the \( d \)-th subsequent time slot.

Consider an arriving packet that must be released to the second-stage fabric at the end of the \( k \)-th subsequent time slot. Our first choice would be to store it at position \( k \) in the buffer; however, if that were our only choice, then this packet would have to be dropped if there was a contending packet at position \( k+1 \) in the buffer at the beginning of the time slot, since the latter packet would have priority for moving into buffer position \( k \). A key observation is that if a packet must be released to the second-stage fabric at the end of the \( k \)-th subsequent time slot, then it would also work to store the packet \( k + 1 \) time slots, or \( k + 2N \), or \( k + 3N \), etc., since the fabric cycle time is \( N \). One downside would be the increased delay the packet would suffer in transiting the router, but this would usually be preferable to dropping the packet.

In optical technology, unfortunately, the buffer size \( M \) tends to be small while we desire to have the number of ports \( N \) as large as possible. So it is probably impractical to implement buffers of size \( pN \), where \( p \) is an integer greater than 1, which would have provided delays up to \( pN – 1 \) time slots. Instead, if we use the recycling option depicted in Fig. 2, we could recycle the packet through the buffer one or more times to achieve longer delays. Consider, for example, \( M = 5 \) and \( N = 8 \). Instead of placing the packet initially in position 3 for transmission at the end of the 3rd subsequent time slot, we could place it in position 1 and recycle it twice, so that it comes out at the end of the 11th subsequent time slot. Since 11 = 3 + 8, the packet will be launched into the second-stage fabric toward the correct output port. An initial placement at position 1 with two recyclings works because 11 = 1 + 2×5.

As a second example, suppose with \( M = 5 \) and \( N = 8 \) we wish to store the packet for 7 time slots. Note that such a storage time would be needed in practice because \( 7 < N \), but would be impossible to attain without recycling as \( 7 > M \). With recycling, however, we can attain the family of desired storage times by either storing in position 2 and recycling 1 time, or position 0 with 3 recyclings, or position 3 with 4 recyclings, or position 1 with 6 recyclings, or position 4 with 7 recyclings.

These initial positions and recyclings can be calculated easily, but the method works only if \( M \) and \( N \) are relatively prime, which means simply that \( M \) and \( N \) have no common factors. In the above example, for instance, \( M = 5 \) (which is a prime number) and \( N = 8 \), which has factors 2×2×2. Since \( M \) and \( N \) have no common factors, they are relatively prime. The same would be true if \( M = 9 = 3×3 \) (not a prime number) and \( N = 8 \) (also not a prime number). The numbers 9 and 8 do not have any common factors and so are relatively prime. However, if \( M = 5 \)
prime) and \( N = 10 = 2 \times 5 \) (not prime), then our method would not work as \( M \) and \( N \) would then both have the common factor 5.

**Method for Calculating Initial Buffer Positions and Number of Times to Recycle:** Suppose that the buffer size \( M \) and the fabric cycle time \( N \) are relatively prime. Suppose we wish to hold a packet in the middle stage of the load-balanced architecture router for a minimum of \( k \) time slots, with the understanding that \( k + N, k + 2N, \) and \( k + 3N \), etc., are also acceptable holding times. Let \( m \) denote an integer with \( 0 \leq m \leq M - 1 \). Then the packet can be stored initially in position \((k + m \times N) \% M\) and subsequently recycled \((k + m \times N) / M\) times, where \% denotes the integer modulus operation and / denotes the integer division operation.

Note that as the integer parameter \( m \) sweeps over all its possible values, all positions in the buffer are accounted for. Hence, all unoccupied positions in the buffer are available for storing an arriving packet. Table I below shows examples for two out of the eight possible values for \( k \).

**Table I. Examples with Buffer Size \( M = 5 \) and Fabric Cycle Time \( N = 8 \)**

<table>
<thead>
<tr>
<th>Time Slot Storage Parameter ( k = 3 )</th>
<th>Time Slot Storage Parameter ( k = 7 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial Storage Position</td>
<td>Number of Recyclings</td>
</tr>
<tr>
<td>((3 + 0 \times 8) % 5 = 3)</td>
<td>((3 + 0 \times 8) / 5 = 0)</td>
</tr>
<tr>
<td>((3 + 1 \times 8) % 5 = 1)</td>
<td>((3 + 1 \times 8) / 5 = 2)</td>
</tr>
<tr>
<td>((3 + 2 \times 8) % 5 = 4)</td>
<td>((3 + 2 \times 8) / 5 = 3)</td>
</tr>
<tr>
<td>((3 + 3 \times 8) % 5 = 2)</td>
<td>((3 + 3 \times 8) / 5 = 5)</td>
</tr>
<tr>
<td>((3 + 4 \times 8) % 5 = 0)</td>
<td>((3 + 4 \times 8) / 5 = 7)</td>
</tr>
</tbody>
</table>

Fig. 3 shows the simulated performance of packet recycling in a five-router network. Packets were not stored that would have exceeded the storage time limit shown on the horizontal axis. The graph for \( M = 512 \) shows no recycling effect because 512 and 128 are not relatively prime.

**3. Concluding Remarks**

Our simulations show that packet recycling can indeed reduce the drop ratio, although excessive recycling can make performance worse rather than better. Hence, one of the challenges in using this technique is to find an optimum amount of recycling. It was discovered through simulation that this optimum depends on the offered traffic load. In other words, an automatic control system should sense the amount of traffic in the network and use this information in adjusting the maximum storage time threshold for best performance.

**References**
