1 Abstract

An increasing number of high performance internetworking protocol routers, LAN and ATM switches use a switched backplane based on a crossbar switch. Most often, these systems use input queues to hold packets waiting to traverse the switching fabric. It is well known that if FIFO input queues are used to hold packets then, even under benign conditions, HOL blocking limits the achievable bandwidth to approximately 58.6% of the maximum. HOL blocking can be overcome by the use of virtual output queuing, which is described here.

An *iSLIP* scheduling algorithm is used to configure the crossbar switch, deciding the order in which packets will be served. An iterative, round-robin algorithm, *iSLIP* can achieve 100% throughput for uniform traffic, yet is simple to implement in hardware.

2 Introduction

In a crossbar switch, we need scheduling algorithm that configures the fabric during each cell time and decides which inputs will be connected to which outputs; this determines which of the N^2 VOQ's are served in each cell time. At the beginning of each cell time, a scheduler examines the contents of the N^2 input queues and finds a conflict free match M between inputs and outputs. This is same as finding a bipartite matching on a graph with max. size.

The problem is that for our application it is too complex to implement and takes too long to complete.It can cause some queues to be starved of service indefinitely. Furthermore, when the traffic is nonuniform, it cannot sustain very high throughput. For practical high-performance systems, we desire algorithms with the following properties.

- *High Throughput.*
- *Starvation Free.*
- *Fast.*
- *Simple to Implement.*

2.1 Parallel Iterative Matching (PIM)

PIM uses randomness to avoid starvation and reduce the no. of iterations needed to converge on a maximal-sized match. The steps of PIM are:

Step 1: Request. Each unmatched input sends a request to every output for which it has a queued cell.

Step 2: Grant. If an unmatched output receives any requests, it grants to one by randomly selecting a request uniformly over all requests.

Step 3: Accept. If an input receives a grant, it accepts one by selecting an output randomly among those that granted to this output.

By considering only unmatched inputs and outputs, each iteration only considers connections not made by earlier iterations.Using randomness comes with its problems, however.

First, it is difficult and expensive to implement at high speed; each arbiter makes a random selection among the members of a time-varying set. Second, when the switch is oversubscribed, PIM can lead to unfairness between connections. Finally, PIM does not perform well for a single iteration; it limits the throughput to approximately 63%, only slightly higher than for a FIFO switch.

2.2 Round Robin Matching (RRM)

Cells are scheduled by round-robin arbiters at each output, and at each input. RRM potentially overcomes two problems in PIM: complexity and unfairness. Implemented as priority encoders, the round-robin arbiters are much simpler and can perform faster than random arbiters. The rotating priority aids the algorithm in assigning bandwidth equally and more fairly among requesting connections. It has 3 steps:

Step 1: Request. Each input sends a request to every output for which it has a queued cell.

Step 2: Grant. If an output receives any requests, it chooses the one that appears next in a fixed, round robin schedule starting from the highest priority element.The output notifies each input whether or not its request was granted. The pointer g_i to the highest priority element of the round-robin schedule is incremented (modulo N) to one location beyond the granted input

Step 3: Accept. If an input receives a grant, it accepts the one that appears next in a fixed, round-robin schedule starting from the highest priority element. The pointer a_i to the highest priority element of the round-robin schedule is incremented (modulo N) to one location beyond the accepted output.

Synchronization of the grant pointers limits performance with random arrival patterns. For Bernoulli arrivals, this synchronization phenomenon leads to a maximum throughput of just 50%.

2.3 iSLIP Matching

The *iSLIP* algorithm improves upon RRM by reducing the synchronization of the output arbiters. *iSLIP* achieves this by not moving the grant pointers unless the grant is accepted. It is identical to RRM except for a condition placed on updating the grant pointers. The Grant step of RRM is changed to:

Step 2: Grant. If an output receives any requests, it chooses the one that appears next in a fixed round-robin schedule, starting from the highest priority element. The output notifies each input whether or not its request was granted. ***The pointer*** g_i ***to the highest priority element of the round-robin schedule is incremented (modulo N) to one location beyond the granted input if, and only if, the grant is accepted in Step 3.***

This small change to the algorithm leads to the following properties:

Property 1: Lowest priority is given to the most recent connection because when the arbiters move their pointers, the most recently granted (accepted) input (output) becomes the lowest priority at that output (input). If input *i* successfully connects to output *j* both a_i and g_i are updated and the connection from input to output becomes the lowest priority connection in the next cell time.

Property 2: No connection is starved because an input will continue to request an output until it is successful. The output will serve at most N-1 other inputs first, waiting at most N cell times to be accepted by each input. Therefore, a requesting input is always served in less than N^2 cell times.

Property 3: Under heavy load, all queues with a common output have the same throughput. This is a consequence of Property 2.

3 Question

The purpose of this assignment is to understand the performance of queuing in a packet switch. Your program should implement different types of scheduling mechanisms, as described below.

Inputs

The command line will specify the following:

- Number of switch input and output ports
- Buffer size
- Packet generation probability
- queue scheduling technique being used

For e.g., the input can be given as below:

./routing -N switchportcount -B buffersize -p packetgenprob -queue INQ/KOUQ/iSLIP -K knockout -out -outputfile -T maxtimeslots

- The switch is assumed to have N input and output ports. Ports are numbered from 0 through N-1. You can consider the default value of N as 8.
- All packets are of same length. Time is slotted. One time slot equals the transmission time of one packet.
- Default buffersize B is 4, i.e, each port will hold upto B fixed length packets.
- Default *packetgenprob* is 0.5. It denotes the probability that an input port will generate a packet in a given slot.
- The *-queue* argument specifies the queue type; the output file will contain the output generated by the program. Default is : INQ.
- Default value of *maxtimeslots* is 10000. It specifies the simulation time.

Switch Operation

The program will consist of three phases:

- Phase 1 - Corresponds to traffic generation.
- Phase 2 - Corresponds to packet scheduling.
- Phase 3 - Corresponds to packet transmission.

All three phases will take place at the beginning of each time slot. Initially, all the packet queues are assumed to be empty.

Traffic Generation

In this phase, each port will generate a packet with probability *packetgenprob*. A packet's destination port is selected randomly with uniform probability from the set of all output ports. The start time of each packet is randomly set between t + 0.001 and t + 0.01. For delay calculation purposes, ignore this offset and assume that the arrival time is t.

Packet delay is calculated as the difference between transmission completion time and the packet arrival time.

Scheduling

In this phase, the packets generated in the previous phase are handled.

- **INQ:-** For each packet generated, if there is no contention for its desired output port, it is selected for transmission and placed in the corresponding output port's buffer. For packets contending for the same output port, one of the packets is randomly selected for transmission and placed in the corresponding output port's buffer; the other packets are queued at the corresponding input port.
- **KOUQ:-** A maximum of K packets (per output port) that arrive in a given slot are queued (based on packet arrival time) at the corresponding output port. If two or more packets have the same arrival time, the packets can be queued in any order. If more than K packets arrive in a slot for a particular output port, then K packets are randomly selected for buffering, and the remaining packets are dropped. The default value is K = 0.6N.
- **iSLIP:-** Implement the *iSLIP* scheduling algorithm with a Virtual Output Queue (VOQ).

Transmission

At each output port, the packet at the head of the queue is transmitted. In case of KOUQ, the packet at the head of the queue will have lowest arrival time.

Outputs

The performance metrics to be measured:

- **Average packet delay:** The mean packet delay computed for all transmitted packets.
- **Average link utilization:** Link utilization for each link, is defined as the fraction of time a link has been used for transmitting a packet, with respect to the entire simulation duration. Average link utilization would be its mean value.
- **KOUQ drop probability:** The probability per slot that more than K packets were generated for an output port. For e.g., if in a given slot, more than K packets were generated for 3 out of 8 output ports, then the probability for that slot is 0.375. You should report the average probability over all the slots for the simulation duration.

The program, upon termination, will give the following line as output (may be separated by tabs), that will be appended to the specified output file.

N	p	Queue Type	Avg PD	Std Dev of PD	Avg link utilization

4 References:

https : *//ieeexplore.ieee.org/document/*769767