Exercise Set 8: Input Queueing, VOQ Crossbar Scheduling, and their Simulation

Assigned: Sat. 4 April 2015, Due: Wed. 22 April 2015

[Lectures: 4.2 Input Queueing, 4.4 CIOQ] [print version, in PDF]

8.1 Saturation Throughput in a 3x3 Input Queued Switch

As we saw in class, a simplified formulation for computing the saturation throughput of a 2x2 input queued switch (single queue per input, i.e. *without* VOQ's) is the following. The head-of-line cells of the two input queues may have the following destination combinations: *(i)* "00", meaning that they are both destined to output 0; or *(ii)* "01", meaning the first is destined to output 0 and the other is destined to output 1; or *(iii)* "10"; or *(iv)* "11". If we assume that the switch is saturated, the queues are always non-empty. If we assume that arrivals are i.i.d. (independent identically distributed) with uniformly distributed destinations, then the probability of each of the above combinations is 0.25, and it is independent of the combination in the previous time slot and of the service decision made in the previous time slot. Combinations 01 and 10 yield 2 outgoing cells per time slot for the two outputs (average throughput 1.0 per output), while combinations 00 and 11 yield only 1 outgoing cell per time slot for both outputs (average throughput 0.5 per output). Hence, the average (saturation) throughput per output over a long time window will be: 0.25*0.5 + 0.25*1.0 + 0.25*1.0 + 0.25*0.5 = 0.75.

Using the same technique, calculate the saturation throughput of a 3x3 input queued switch (single queue per input). The head-of-line cells of the three input queues may form 27 combinations. List these combinations; compute the probability of each; compute the average per-output throughput of the switch in each of these cases (and briefly explain); and, finally, compute the average saturation throughput.

8.2 Simulation of PIM and iSLIP under Uniform Traffic

This exercise concerns the evaluation of the performance of the PIM [1] and iSLIP [2] scheduling algorithms for VOQ crossbars, using computer simulation. The simulator which you will use is **BookSim**. BookSim is a modern cycle-time based network-on-chip (NoC) simulator, written in C++. The user can configure various parameters such as: topologies, sub-networks, routing algorithms, buffering schemes, router micro-architectures, and traffic patterns. The BookSim can be found at http://nocs.stanford.edu/cgi-bin/trac.cgi/wiki/Resources/BookSim and the GitHub for its latest version at https://github.com/booksim/booksim2.

You will simulate a single-stage (VOQ) crossbar under randomly-destined traffic, comprising unit-length packets (cells). In BookSim, you can run a simulation specifying a configuration file in the following way.

```
> ./booksim config.txt injection_rate=0.7
```

In addition, through the command line, you may pass parameter values, which will override the corresponding values in the configuration file -- in the example above, we override the "injection_rate", which is the load at each input.

Here is a sample configuration file.

```
> cat config.txt

topology = fly; // k-ary n-fly = ~ banyan
k = 16;         // 16 ports per switch
```

```
n = 1;          // one stage

router = iq;   // switch type input queueing
noq = 1;       // use VOQs at each input

num_vcs = 16;  // number of queues per input; set it equal to k for noq = 1


// if buffer_policy = private & buf_size = -1, then vc_buf_size = vc_buf_size
// if buffer_policy = shared  & buf_size >= 0, then buf_size = buf_size

buffer_policy = private; // or shared
buf_size = -1;           // -1 for buffer_policy = private;
vc_buf_size = 1024;      // size of each VOQ with buffer_policy = private


internal_speedup = 1.0;  // crossbar internal speedup -- expects a real number
output_buffer_size = -1; // means unlimited output buffer size; used when spee

sw_allocator = islip;   // pim
alloc_iters = 1;        // iterations of scheduling PIM/iSLIP algorithms; warn

// traffic pattern
packet_size = 1;        // each packet consists of 1 flit
traffic = uniform;      // Bernoulli iid uniformly-destined traffic.
injection_rate = 0.7;

 // misc
sample_period  = 100000;
routing_function = dest_tag;
use_read_write = 0;
routing_delay  = 0;
vc_alloc_delay = 1;
sw_alloc_delay = 1;
st_final_delay = 1;
private_buf_size = -1;
```

In this configuration, each VOQ has separate (private) buffer space for 1024 packets.

To generate a load-latency curve for one switch configuration, run multiple simulations for increasing load values between [0.1:0.99]; for each run, collect the "Network latency average" output of the simulator, and create a load/latency (2d) matrix, which you can plot using e.g. gnuplot.

Generate the load-latency curves for PIM and iSLIP, for 16x16, 32x32, and 64x64 crossbars. Compare and comment on the curves you obtained.

Next, considering the 32x32 switch, obtain the load-latency curves for PIM and iSLIP for 1, 2 and 4 scheduling iterations (parameter "alloc_iters"). Are the results improving with more iterations? If not, check whether the simulator really utilized the alloc_iters parameter value that you specified. If it didn't, try to manually change it in the C++ files of PIM and iSLIP, which are located in directory "src/allocators".

Next, consider a 32x32 switch using one iterations of iSLIP (alloc_iters = 1), under full input load ("injection_rate" = 1.0). In this test, assume that the buffer space at each input of the switch is shared by the local (at this input) VOQs. To do so, set "buffer_policy = shared" and play with the shared buffer size (parameter "buf_size"). Plot the "Accepted packet rate average" for increasing buffer size {1,2, 8, 32, 128, 1024}. Why iSLIP cannot achieve full throughput for all buffer sizes? Repeat the experiment using PIM instead of iSLIP. Is PIM worse than iSLIP? Finally, repeat the aforementioned experiments using an internal speedup of 2. Comment on the results. Which one of the following do you think is more cost-effective for a 32x32 switch chip

implementation, large input buffers, multiple scheduling iterations or internal speedup?

References

[1] T. Anderson, S. Owicki, J. Saxe, C. Thacker: "High-Speed Switch Scheduling for Local-Area Networks", ACM Trans. on Computer Systems, vol. 11, no. 4, Nov. 1993, pp. 319-352.

[2] N. McKeown: "The iSLIP Scheduling Algorithm for Input-Queued Switches", IEEE/ACM Trans. on Networking, vol. 7, no. 2, April 1999, pp. 188-201.

Up to the Home Page of CS-534

Last updated: 4 April 2015 by N. Chrysos; earlier versions by G. Passas, A. Psathakis, and M. Katevenis.