CS-534: Packet Switch Architecture
Spring 2013

Department of Computer Science

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

Assigned: Thu. 2 May 2013 (should have been: 19/4, wk. 8) - Due: Wed. 22 May 2013 (week 11 [should have been: 13/5 - wk.10])

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 and the next exercise concern the evaluation of the performance of the PIM [1], iSLIP [2], and DRRM [3] scheduling algorithms for VOQ crossbars, using machine simulation. You may implement and evaluate these algorithms using any simulator that you are familiar with (e.g. the ns simulator, or your ad hoc simulator), but we can offer the following suggestions/recommendations:

- A good, modern network-on-chip (NoC) simulator is **BookSim**, a cycle-time based 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. It can be found at: http://nocs.stanford.edu/cgi-bin/trac.cgi/wiki/Resources/BookSim
- An older and *simpler* simulator, developed by the High Performance Networking Group at Stanford University, was "**SIM** - A Fixed Length Packet Simulator". Although BookSim is in general preferable, for this particular exercise we suggest that you use SIM, due to SIM's simplicity. SIM used to be found at: http://klamath.stanford.edu/tools/SIM/ (but may no longer be there - and we have no new reference for it). A local copy of SIM exists on the `csd.uoc.gr` machines; the manual is at: **`~hy534/sim_manual.pdf`** and the source, input, and other related files are at: **`~hy534/simv2.35.tar.gz`** which include an adaptation of the Makefiles and other files for compilation in our local machines environment. SIM is a slotted-time (cell-time)

ATM switch simulator, written in ANSI C. It provides implementations for iSLIP, PIM, and many other scheduling algorithms - but not for DRRM. So, for PIM and iSLIP you may just run simulations using the ready implementations of SIM, while, if you decide to also do the optional exercise for DRRM, you will have to write your own implementation.

We are concerned with (VOQ) crossbars with a single priority level, unicast traffic, and unit-length packets (cells). Traffic may be feasible (Bernoulli iid uniform) or infeasible (some outputs are hot spots). SIM implements Bernoulli iid uniform traffic, but not hot-spot traffic; thus, you also have to submit your own implementation of hot-spot traffic. SIM also implements multiple priority levels and multicast, but you will not be using this functionality in this exercise set.

Generate the load-delay curve of PIM and iSLIP with one iteration of the matching steps in a 4x4 crossbar with a single priority level and unicast Bernoulli iid uniform traffic. To do so, run a simulation for each load value, measure the average delay of cells in each simulation run, and plot the resulting load-delay curve. You can run a simulation in this way:

```
> ./sim -f config.txt
```

where config.txt is a configuration file. For example, using the following configuration file, you can measure the average cell delay in a 4x4 crossbar scheduled by iSLIP with one iteration of the matching steps, when traffic load is 0.7 (70%). Average cell delay is reported in SIM's output as "Total Latency over all cells".

```
> cat config.txt

Numswitches 1 Switch 0
Numinputs  4
Numoutputs 4
InputAction  defaultInputAction
OutputAction defaultOutputAction
Fabric crossbar
Algorithm islip -n 1
0 bernoulli_iid_uniform -u 0.7
1 bernoulli_iid_uniform -u 0.7
2 bernoulli_iid_uniform -u 0.7
3 bernoulli_iid_uniform -u 0.7
Stats
Arrivals
Departures
Latency
Occupancy
Histograms
Arrivals
Departures
Latency
Occupancy
```

Also generate PIM and iSLIP curves for 16x16, 64x64, and 256x256 crossbars and repeat for iSLIP and PIM with 2, 4, 6, and 8 iterations of the matching steps.

When running your simulations, consider and answer the following questions: *(i)* We are interested in average cell delay when the switch is stable. Practically, stability can be decided by checking the delay of cells: If during a simulation run delay keeps increasing linearly with simulated time, the switch is unstable --a direct consequence of Little's law. How much time do you need to simulate to decide stability? Is this related to traffic load or switch size? How? How is stability related to the RAM utilization of your computer? In SIM, the simulated time when simulation stops is determined by the constant DEFAULT_SIMULATION_LENGTH in file sim.c. *(ii)* Which is the running time complexity of your simulations with respect to simulated time, switch size, and number of iterations?

8.3 Implementation of DRRM and Hot-Spot Traffic

----- *This exercise is OPTIONAL* -----

Optionally also implement DRRM [3] on your simulator, and repeat exercise 8.2 for that scheduling algorithm. To implement DRRM you have to add to SIM a modification of iSLIP. iSLIP is implemented in file ALGORITHMS/islip.c. The main loop of the algorithm is in lines 172-218. You have to modify mainly this part, reversing the order of arbitration (selection) --in DRRM, first arbitrate the inputs and then the outputs. DRRM operates in the following two steps:

1. **Request:** If an input has cells for any output, it chooses the output that appears next in a fixed, round robin schedule starting from the highest priority element and sends a request to this output. The pointer to the highest priority element of the round-robin schedule is incremented (modulo N) to one location beyond the requested output if and only if the request is granted in the next step 2.
2. **Grant:** If an output receives any request, it grants the one that appears next in a fixed, round-robin schedule starting from the highest priority element and the pointer to the highest priority element of the round-robin schedule is incremented (module N) to one location beyond the granted input.

How does DRRM compare to iSLIP? Compare DRRM to iSLIP under infeasible traffic. An example is the following. Inputs 1, 2, 3, and 4 send traffic to output 0, at a rate of 0.25 each. Input 0 sends traffic to outputs 0 and 1: it sends to output 0 at a rate of 0.25, and to output 1 at a rate L. This pattern overloads output 0, but not output 1. Plot the delay of cells from input 0 to output 1, as L varies from 0.05 to 0.75. Observe and explain.

To implement the above traffic patttern, add to SIM a modification of file TRAFFIC/bernoulli_iid_uniform.c. The main part for traffic generation is in lines 397-436: Variable "psend" determines the probability of generating a cell and variable "output" the destination output of the generated cell. Notice that you have to measure the delay of cells from input 0 to output 1 only (not all cels). So, you have to selectively stamp cells. This can be done by conditioning the update of the total latency variable in file latencyStats.c

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.

[3] J. Chao, "Saturn: A Terabit Packet Switch using Dual Round-Robin", IEEE Commun. Mag., vol. 38, Dec. 2000, pp. 78-84.

Up to the Home Page of CS-534

Last updated: 2 May 2013, by G. Passas, A. Psathakis, and M. Katevenis.