June 21st 2010

Exam of Switching architectures

Rules for the exam. It is forbidden to use notes, books or calculators. When needed, use approximations. Report the solution of each problem on a different paper sheet.

Time available: 70 minutes.

PROBLEM A
Describe in pseudo-code an algorithm to schedule the transmissions of unicast and multicast packets in a slotted input queued switch, of size $N \times P$. Assume that the queue structure at each input port is the following: i) a single FIFO queue for all unicast packets destined to a particular output; ii) a single FIFO queue for all the multicast packets. At each timeslot, let $Q[i][j]$ be the size of the queue for unicast packets at input $i$ and destined to output $j$. Let $M[i]$ be the size of the queue for multicast packets at input $i$. Let $X$ a matrix describing the switching configuration chosen in the current timeslot, based on the state of the queues. More precisely, $X[i][j]$ is a boolean variable, which assumes the value true iff the crosspoint from input $i$ to output $j$ is active, i.e. a packet is sent from input ”$i$” to output $j$ in the current timeslot.

1. Compute the total number of queues for each input and in the whole switch.

2. Write in pseudo-code a maximal scheduling algorithm that allows fanout splitting and serves the multicast packets at higher priority with respect to unicast packets. Assume that a function destInMCQueue($j, i$) that returns true iff output $j$ belongs to the fanout set of the packet at the head of the multicast queue $M[i]$ at input $i$ is available.

3. Does the algorithm achieve 100% throughput under any admissible traffic? Why?

PROBLEM B
Consider a $2000 \times 2000$ switch built just using basic modules $10 \times 10$.

1. Design a minimum-cost strictly non-blocking network. Draw the structure of the network and compute the total number of basic modules.

2. Design a minimum-cost non-interruptible, rearrangeable switching network. Draw the structure of the network and compute the total number of basic modules.

3. Are the two networks equivalent in terms of cost (i.e., number of basic modules) and control (i.e., configuration algorithm)? Why?

PROBLEM C
Prove that the weight of a greedy maximum weight matching (GWM) is at least equal to half the weight of the maximum weight matching (MWM). In other words, $W(GWM) \geq \frac{1}{2} W(MWM)$.

In the proof, denote by $E$ the set of edges in the bipartite graph, by $G$ the sub-set of edges selected by the GWM, and by $M$ the sub-set of edges selected by the MWM scheduler.
SOLUTIONS

Problem A

The total number of queues for each input is $P + 1$, and for the whole switch is $N(P + 1)$. The switch cannot obtain the maximum throughput because of any of the following reasons: (i) the queueing is not optimal and suffers the HoL blocking problem for multicast traffic, (ii) the scheduling policy is not optimal. In general, an input queued switch cannot obtain the maximum throughput under any admissible multicast traffic because of intrinsic architecture limitations, highlighted by specific arrivals patterns.

// initialize the data structures
for j=1...P // for each output port
  output_reserved[j]=false
for i=1...N // for each input port
  X[i][j]=false
// scheduler decision
for i=1...N // for each input port
  // try to serve the multicast traffic
  multicast_served=false
  if (M[i]>0) // check if the mc queue is non-empty
    for j=1...P // for each output port
      if (destInMCQueue(j,i) AND (output_reserved[j]==false))
        // it means that the output is present in the
        // fanout set and it has not been reserved so far
        output_reserved[j]=true
        X[i][j]=true
        multicast_served=true
  // try to serve a unicast queue only if it is available
  if (multicast_served==false)
    // M[i] has not served; now look at unicast traffic
    for j=1...P
      if ((Q[i][j]>0) AND (output_reserved[j]==false))
        X[i][j]=true
        output_reserved[j]=true
        break

Problem B

To design a strictly-non-blocking network,

$$C_{2000,SNB} = 200C_{10 \times 19} + 19C_{200,SNB} + 200C_{19 \times 10}$$

where

$$C_{10 \times 19} = 2C_{10}$$
$$C_{200,SNB} = 20C_{10 \times 19} + 19C_{20,SNB} + 20C_{10 \times 19}$$

and

$$C_{20,SNB} = 4C_{10}$$

since 4 crossbar $k \times k$ can be always combined to build a crossbar $(2k) \times (2k)$. Hence,

$$C_{200,SNB} = (20 \times 2 + 19 \times 4 + 20 \times 2)C_{10} = 156C_{10}$$

and, finally,

$$C_{2000,SNB} = (200 \times 2 + 19 \times 156 + 200 \times 2)C_{10} = 3764C_{10}$$

To design a non-interruptible, rearrangeable (NIR) network,

$$C_{2000,NIR} = 250C_{8 \times 10} + 10C_{250,REAR} + 250C_{10 \times 8}$$

where

$$C_{8 \times 10} = C_{10}$$

Hence,

$$C_{250,REAR} = 25C_{10} + 10C_{25,REAR} + 25C_{10}$$

where, using the classical Clos construction:

$$C_{25,REAR} = 3C_9 + 9C_3 + 3C_9$$

where

$$C_9 = C_{10} \quad C_3 = C_{10}/3$$
Finally,

\[ C_{25,REAR} = 9C_{10} \]

and

\[ C_{250,REAR} = 140C_{10} \]

and

\[ C_{2000,NIR} = 1900C_{10} \]

The first network has a complexity almost twice than the second one, but its control algorithm is trivial. The control algorithm for the second network is a variant of Paul algorithm, in which the two additional medium-stage modules are used to exploit multipath and avoid interruptions.

**Problem C**

The solution has been provided during the class.