4.3 Virtual Output Queue (VOQ) Input Q’ing and the Crossbar Scheduling Problem

- Crossbar Switch with one Buffer Memory per Input Line
- Throughput per Buffer Memory = 1 (incoming) + 1 (outgoing)
- Multiple (one per output) Queues per Buffer Memory:
  - “Virtual Output Queues – VOQ”
    - $N$ queues per input, $N^2$ queues total, for $N \times N$ switch
- Crossbar Scheduling, per cell-time:
  - pairings (“marriages”) between inputs and outputs – each input specifies a subset of the outputs that it accepts to be married to
  - interdependent decisions – difficult problem!
Crossbar Scheduling: Parallel Iterative Matching (PIM)

**Request Phase:**
All inputs send their requests in parallel.

**Grant Phase:**
Each output, independently, grants to one of the requests that it received.

**Accept Phase:**
Each input accepts one of the grants that it received.

---

**First Iteration**

Unmatched inputs resend their requests.

Unmatched outputs (rcv'd no accept) grant to one of the received requests.

Unmatched inputs accept one of the received grants.

---

**Second Iteration**

- Original PIM proposal: outputs grant randomly among requesting inputs, inputs accept randomly among granting outputs.

---

4.3 VOQ, Crossbar Scheduling
4.3 VOQ, Crossbar Scheduling

### Maximum Matching

Cannot add any new connection without breaking some already made connection(s)

### Maximum Matching

Maximum possible number of connections for the given request pattern
**iSLIP: Most Popular, Practical Crossbar Scheduler**

- Practical variation of PIM
- Widely used in commercial switch products
- Nick McKeown: “The iSLIP Scheduling Algorithm for Input-Queued Switches”, IEEE/ACM Tr. on Networking, April 1999
- Performance properties:
  - performs well under uniform and heavy load, when most VOQ’s are non-empty and matching almost “rotate” among inputs & outputs
  - adds delay under medium loads, until most VOQ’s become non-empty
  - does not perform very well under “unbalanced” traffic (each input preferentially sends to one or a few “favored” output(s) of itself)
iSLIP → variation of PIM

- fewer iterations
- better fairness

Nick McKeown: IEEE/ACM TON April 1999

was used in Cisco GSR-12000, TinyTera, Abirio/PMC-Sierra, etc.

(a) REQUEST: like PIM

(b) GRANT:

- PIM: each output randomly grants to a requesting input
- iSLIP: grant in a Round-Robin fashion to a requesting input
  - top priority = next of: previous grant
  - NO!!! see below
  - previous grant that was accepted!

(c) ACCEPT:

- PIM: each input randomly accepts one of the granting outputs
- iSLIP: accept in Round-Robin priority a granting output
  - with top priority = next of previous accept

- if we were granting round-robin after the previous grant, without regard as to whether the grant was accepted or not, then the grant pointers way get synchronized in a "bad way" and stay that way for a long time, resulting in very poor perf.

- perform well even with a single iteration!
  (versus 2-4 iterations of PIM)
  reason: under heavy load, with all inputs requesting all outputs, pointers get out of sync, desynchronized, and scheduler degenerates into time division multiplexing, so can reach even 100% throughput under uniform load

- more iterations improve the delay
- has good fairness properties
- is relatively easy to implement
Fig. 21. Interconnection of $2N$ arbiters to implement iSLIP for an $N \times N$ switch.
**4.3 VOQ, Crossbar Scheduling**

- Cell Arrivals increment $Q_{ij}$
- When $Q_{ij} \geq 1$ a Request $R_{ij}$ is issued
- Requests issued decrement $Q_{ij}$ (pending decision on cell transfer)
- When pending decision is resolved:
  - Successful grants leave $Q_{ij}$ as is (already decremented)
  - Failed grants re-increment $Q_{ij}$ (restore unaccepted request)

Assumption for this scheme to work: fixed-size cells

⇒ Requests for cells and actual cell transfers are interchangeable
with each other: if the "first" request, on behalf of the
"first" cell in the queue is not granted, then the "second"
request (issued on behalf of the "second" cell) will result
in transferring the "first" cell, if granted.

**Ref:**
Oki, Reja-Cossa, Chao: "A Pipeline-Based Approach for Minimal
Size-Sharing Scheduling in Input-Buffered Switches" IEEE Communication

---

http://archvlsi.ics.forth.gr/~kateveni/534