The iSLIP scheduling algorithm for IQ switches

- iSUP algorithm
  - Is it a queuing mechanism?
    - No
  - Does it assume any queueing mechanism?
    - Yes – virtual output queues
  - How does iSLP compare with LQF/OCF?
    1. simple to implement \(\rightarrow\) good point
    2. sub-optimal under non-inform traffic?

- Switch architecture assumptions
  - Cross-bar backplane \(\Rightarrow\) “centralized” scheduler
  - Fixed size cells

- Is centralized control desirable?
  - Ans. No. The two things, important during a switch design, are:
    1. implementation complexity
    2. control complexity \(\Rightarrow\) self-routing desirable.

- Three demerits of Cross-bar switches:
  1. \(N^2\) complexity – may be reduced using Clo’s networks
  2. Guaranteed QoS not possible
    - Con’t control “when” a cell should depart
    - Use TSI (time-slot interchanger) or “speedup”
  3. HOL blocking because I/P Queues are usually used.

Maximal match versus maximum match

- Maximal match \(\leq\) maximum match
- Maximal match is incremental
  - Previous connections are kept while searching/making new connections
- Which one is easier to implement?
  - Ans. Maximal match

- Parallel iterative matching (PIM)
  - Uses randomness to avoid starvation
  - Three steps (in each iteration)
    1. Request [made by inputs]
    2. Grant [made by outputs]
    3. Accept [made by inputs]
  - 3 merits of PIM
    1. a maximal match would be achieved in, on average, O(log N) iterations
    2. no starvation \(\Rightarrow\) all inputs will be “granted”
    3. no need to keep track of previous state.
3 demerits of PIM
(a) difficult to implement ⇔ randomization
(b) unfair for oversubscribed switch (fig.3)
(c) For one iteration, throughput is limited to 63.2%.

**Basic RRM** (round-robin matching)

- Almost same as PIM except that Grants and Accepts are according to round-robin rather than random.
  
  (See Fig. 4)

- Example in Fig. 4 found a match of size 2 – not the maximum size match

- RRM is superior to PIM in the following aspects:
  (i) It is less complex compared to PIM
  (ii) It is fair compared to PIM

- Problem with RRM
  (i) Throughput can be limited to smaller values – primarily because granting arbiters become “synchronized”.
  ⇒ Many outputs try to grant to the same input. (see Fig. 6 and Fig.7)

**iSLIP**

- Addresses the “synchronization” issue of grant arbiters without adding much to the complexity.
- IDEA: “DO NOT MOVE THE GRANT POINTERS UNLESS GRANT IS ACCEPTED”

**3 Properties of iSLIP**

(i) lowest priority given to most recently made connection (both grant and accept pointers are updated when connection is made)

(ii) No connection is starved. Why?

- A cell will have to wait at most $N^2$ slots.

(iii) under heavy load, all queues with a common output have the same throughput

⇒ average latency is proportional to the burst length.

**iterative iSLIP:**

(algorithm will converge in at most $N$ iterations)

- more iterations lead to a greater match
• \( \log_2 N \) (approx.) iterations are sufficient

**Variations of iSLIP**

• Prioritized iSLIP to deal with various service classes

• A compromise between maximum size and maximum weight matchings – Divided queue size (or latency) into bins (quantized intervals) and then apply prioritized iSLIP on those