Programming scheduling algorithms for Input-Queued switches

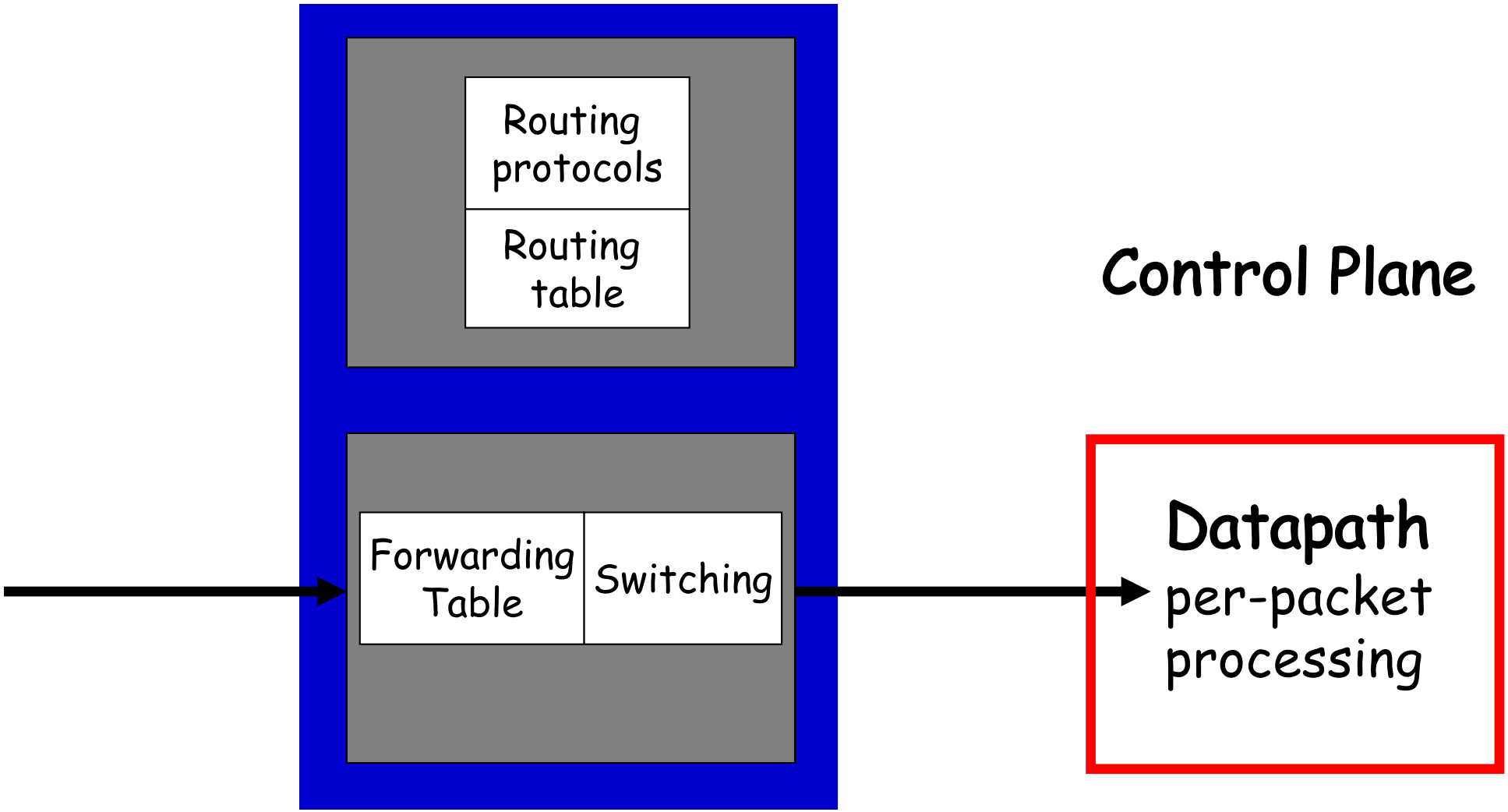
Class on

Switching technologies for data centers
2021/22

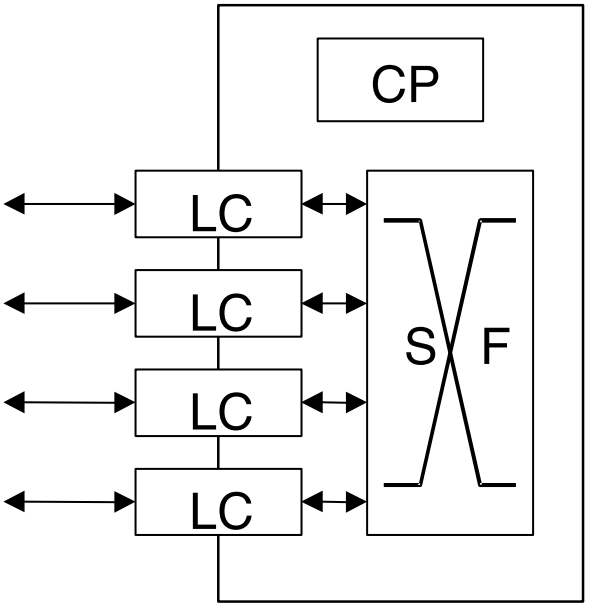
Paolo Giaccone

Telecommunication Network Group
Dept. of Electronics and Telecommunication
Politecnico di Torino
http://www.tlc-networks.polito.it

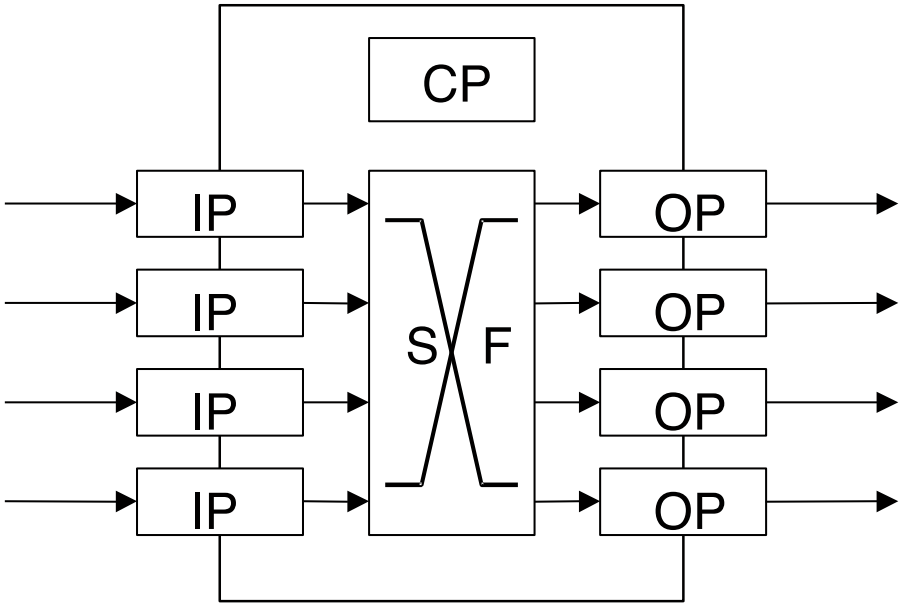
Basic router architecture



Hardware architecture



physical structure



logical structure

Hardware architecture

Main elements

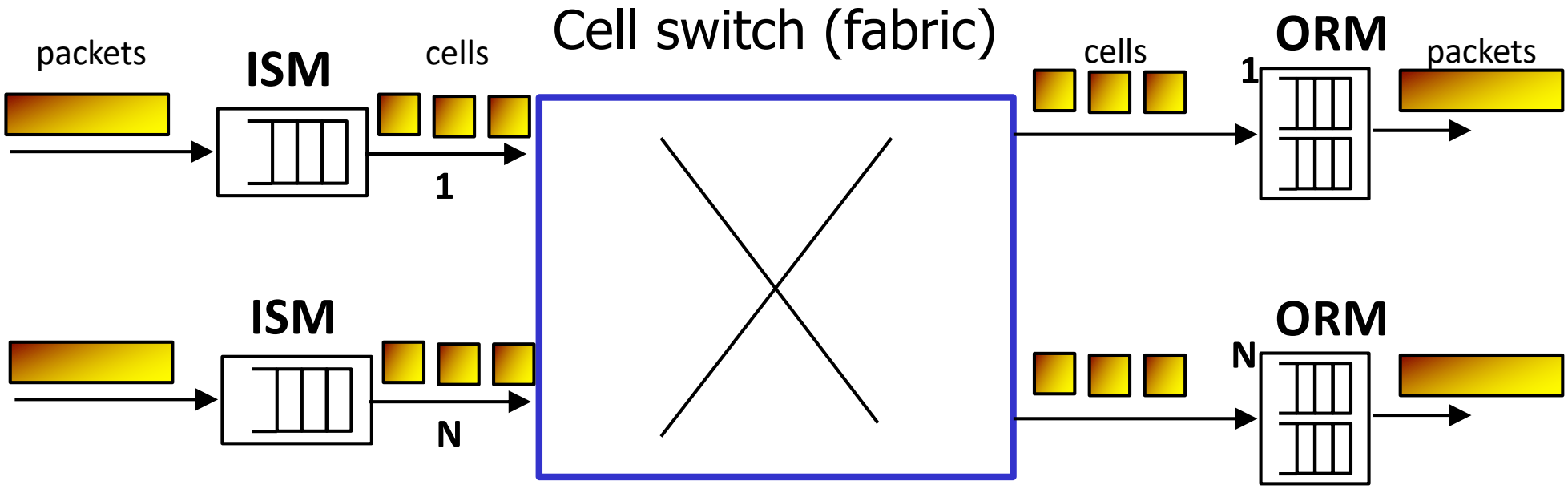
- line cards
  - support input/output transmissions
  - adapt packets to the internal format of the switching fabric
  - support data link protocols
  - In most architectures
    - store packets
    - classify packets
    - schedule packets
    - support security
- switching fabric
  - transfers packets from input ports to output ports

Hardware architecture

- control processor/network processor
  - runs routing protocols
  - computes and stores **routing tables**
  - manages the overall system
  - sometimes
    - store packets
    - classify packets
    - schedule packets
    - support security
- forwarding engines
  - inspect packet headers
  - compute the packet destination (lookup)
    - Searching **routing or forwarding** (cache) tables
  - rewrite packet headers

Cell-based routers

- ISM: Input-Segmentation Module
- ORM: Output-Reassembly Module
- *packet*: variable-size data unit
- *cell*: fixed-size data unit

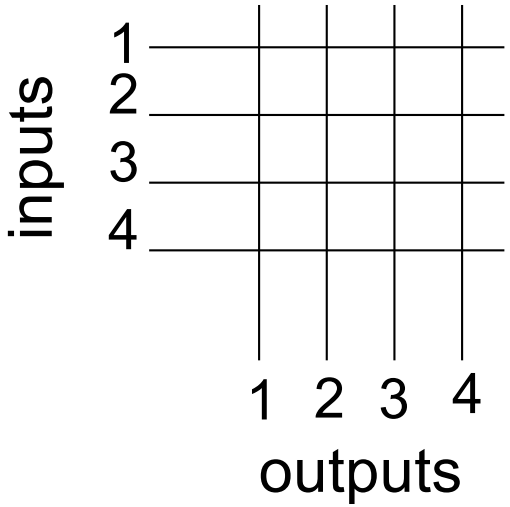


Switching fabric

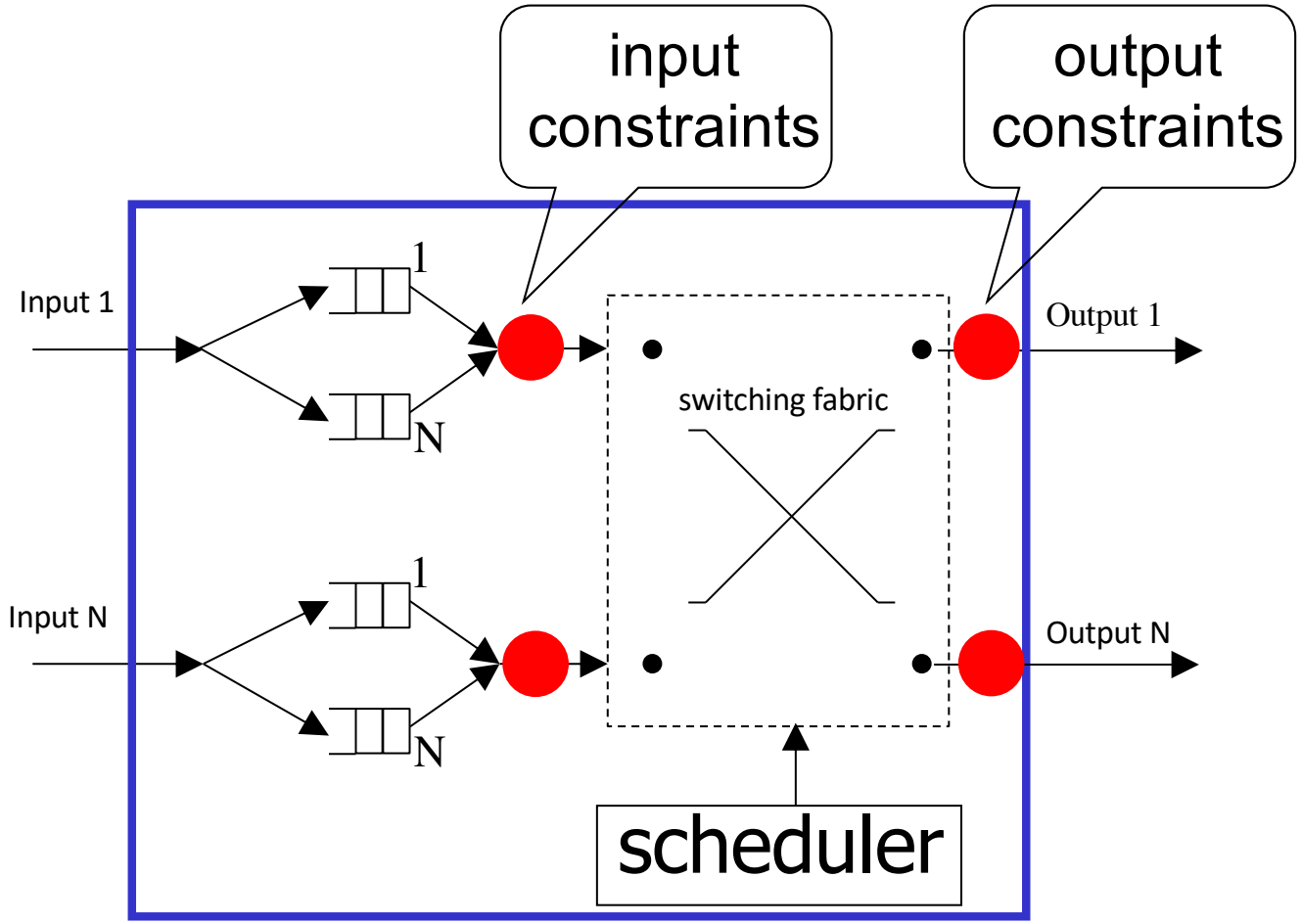
- Our assumptions:
  - bufferless
    - to reduce internal hardware complexity
  - non-blocking
    - given a non-conflicting set of inputs/outputs, it is always possible to connect inputs with outputs

Switching fabric

- Examples:
  - bus
  - shared memory
  - crossbar
  - multi-stage
    - rearrangeable Clos network
    - Benes network
    - Batcher-Banyan network (self-routing)
- Switching constraints
  - at most one cell for each input and for each output can be transferred



IQ switches with VOQ

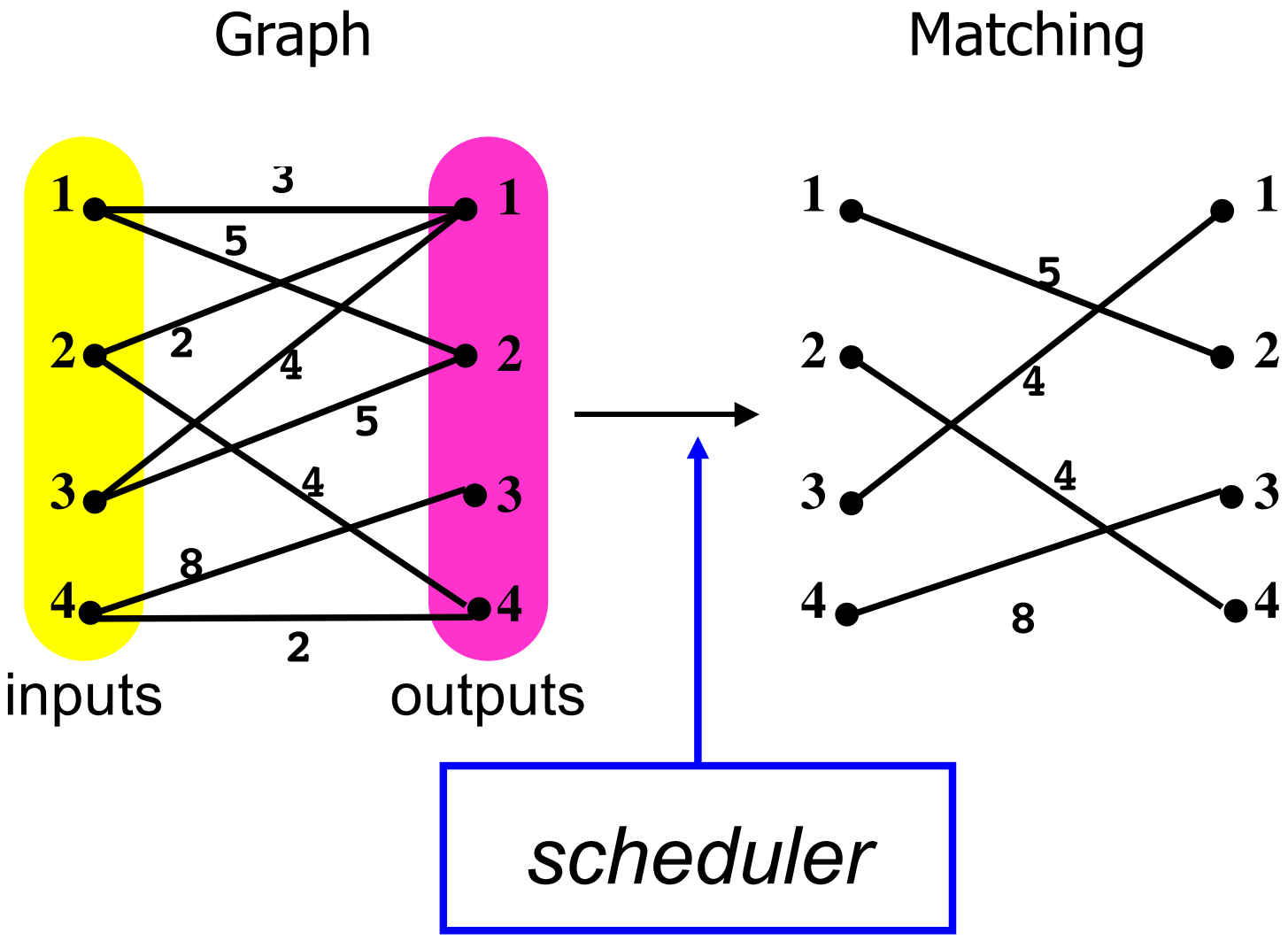


Note: from now on, we always assume VOQ at the switch inputs

Scheduling in IQ switches

- Scheduling can be modeled as a matching problem in a bipartite graph
  - the edge from node *i* to node *j* refers to packets at input *i* and directed to output *j*
  - the weight of the edge can be
    - binary (not empty/empty queue)
    - queue length
    - HOL cell waiting time, or cell age
    - some other metric indicating the priority of the HOL cell to be served

Scheduling in IQ switches



Maximum Weight Matching

- Maximum Weight Matching (MWM)
  - among all the possible N! matchings, selects the one with the highest weight (sum of the queue lengths)
    - MWM is generally not unique
  - optimal in terms of performance
    - maximum throughput and minimum delay
  - too complex to be implemented in hardware at high speed
    - the best MWM algorithm requires $O(N^3)$ iterations, and cannot be implemented efficiently, since it is based on a flow augmentation path algorithm

Maximum Size Matching

- Maximum Size Matching (MSM)
  - among all the possible matchings, selects the one with the highest number of edges/packets (like MWM with binary edge weights)
    - MSM is generally not unique
  - the best MSM algorithm requires $O(N^{2.5})$ iterations, and cannot be implemented efficiently, since it is based on a flow augmentation path algorithm

Maximum Size Matching

- MSM maximizes the instantaneous throughput
- MSM may not yield 100% throughput
  - short term decisions can be inefficient in the long term
  - non-binary edge weights allow MWM to maximize the long-term throughput

Approximations of MSM and MWM

- Motivation
  - strong interest in scheduling algorithms with
    - very low complexity
    - high performance
- Usually
  - implementable schedulers (low complexity)
    $\Rightarrow$ low throughput, long delays
  - theoretical schedulers (high complexity)
    $\Rightarrow$ high throughput, short delays

Programming the scheduling algorithm

- Input data structure:

  `int VOQ[N][N]`

  - `VOQ[i][j]` is the # of enqueued packets from input i to output j
- Output data structure:

  `int matching[N]`

  - `matching[i]=j` means that input i is connected to output j
  - `matching[i]=-1` means that input i is not connected

Programming the scheduling algorithm

- Auxiliary data structure

  `boolean output_reserved[N]`

  - `output_reserved[j]=true` iff output j has been already matched with some input

- Initialize

```
for (i=0; i<N; i++) {
        matching[i]=-1; // input i not connected
        output_reserved[i]=false; // output i available
}
```

Programming the scheduling algorithm

- Scheduling approximating the MSM

```
for (i=0; i<N; i++) {// for each input
    for (j=0; j<N; j++) {// for each output
      if ((VOQ[i][j])>0 &&
(output_reserved[j]==false)) { // check if the VOQ is
not empty and the corresponding output is yet available
          matching[i]=j; // input i not connected
          output_reserved[j]=true; // reserve output j
          break; // consider next input
       }
    }
}
```

Other programming variants

- the code for approximating the MSM can be modified
  - to give higher priority to longest queues or to other metrics
  - to support priority-based policies
  - to support multicast traffic
    - requires some minor modifications

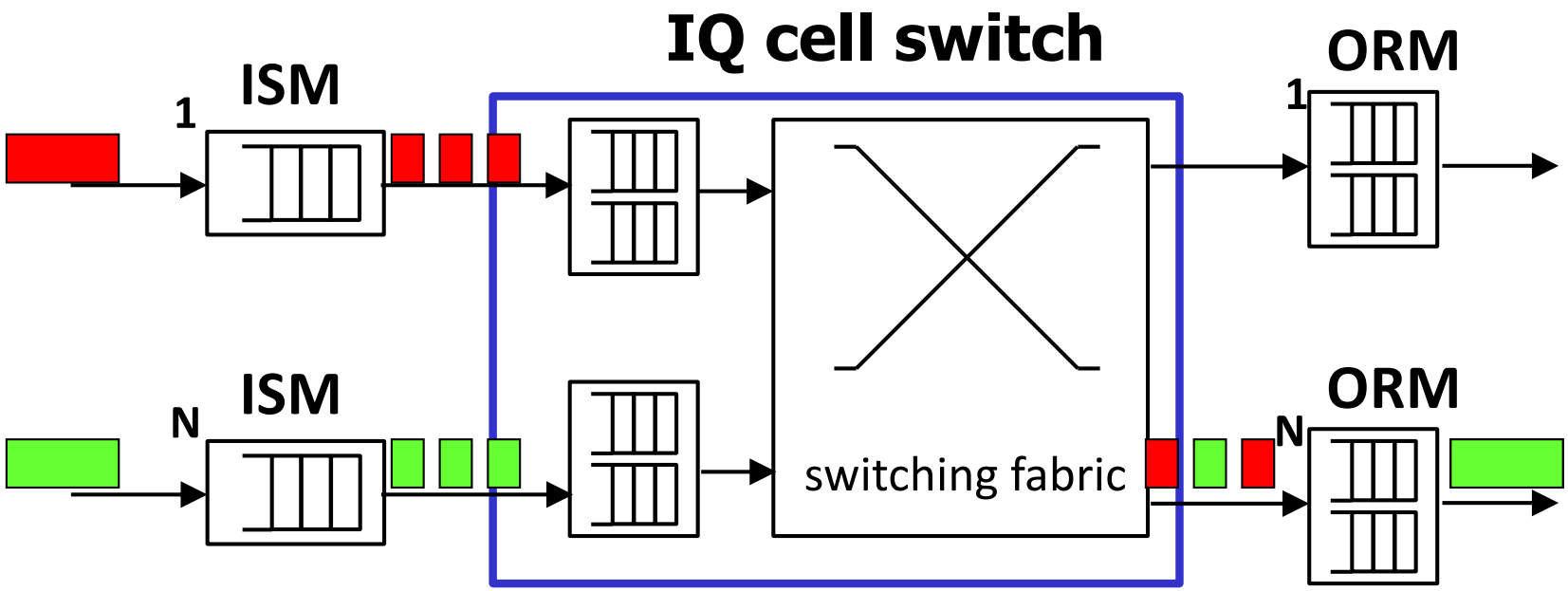
Routers and switches

- IP routers deal with variable-size packets
- Hardware switching fabrics often deal with fixed-size cells

Question:

- how to integrate an hardware switching fabric within an IP router?

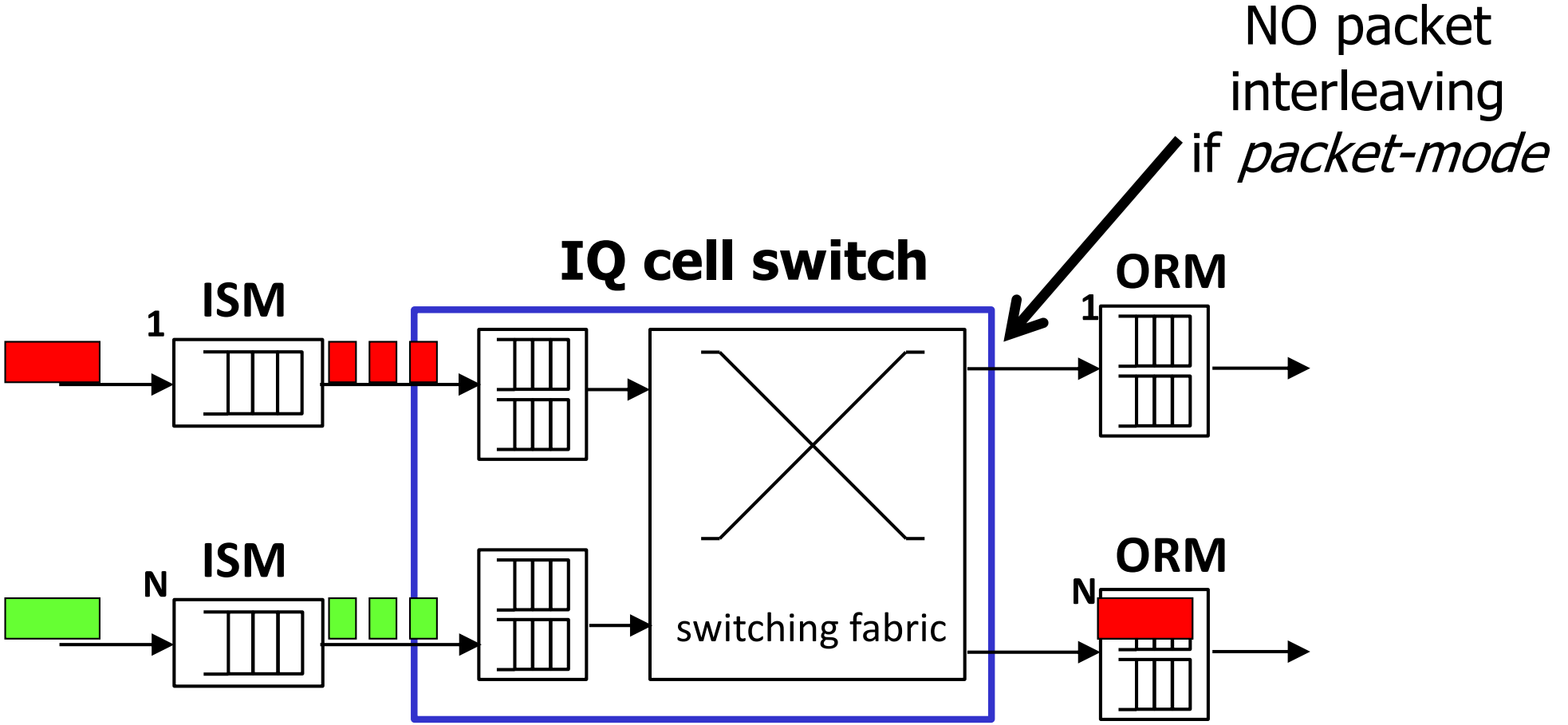
Router based on an IQ cell switch: *cell-mode*



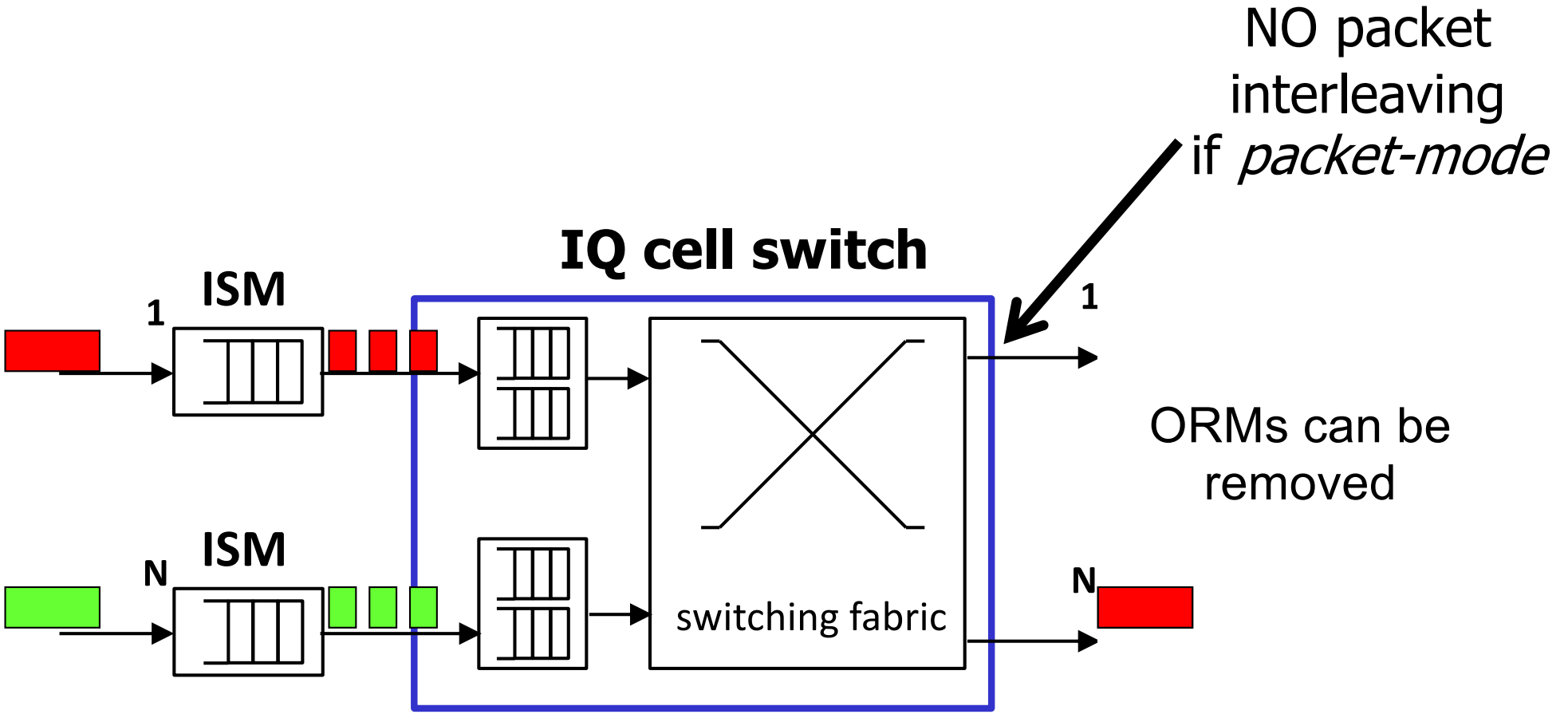
Cell-mode scheduling

- Scheduling algorithms work at cell level
  - pros:
    - 100% throughput achievable
  - cons:
    - interleaving of packets at the outputs of the switching fabric

Router based on an IQ cell switch: *packet-mode*



Router based on an IQ cell switch: *packet-mode*



Packet-mode scheduling

- Rule: packets transferred as *trains of cells*
  - when an input starts transferring the first cell of a packet comprising k cells, it continues to transfer in the following k-1 time slots
- Pros:
  - no interleaving of packets at the outputs
  - easy extension of traditional schedulers
- Cons:
  - starvation due to long packets
    - inherent in packet systems without preemption
    - negligible for high speed rates

Programming packet mode

- simple variant of scheduler for IQ switch
  - block the matching[i]=j until the last cell of the packet
    - e.g., using a binary flag
  - compute the matching on only the unblocked input and output pairs
    - i.e., on a subset of the rows and columns of VOQ