The Internet is a mesh of routers

- **Access router:**
  - high number of ports at low speed (kbps/Mbps)
  - several access protocols (modem, ADSL, cable)

- **Enterprise router:**
  - medium number of ports at high speed (Mbps)
  - several services (IP classification, filtering)

- **Core router:**
  - moderate number of ports at very high speed (Gbps)
  - very high throughput

Hardware architecture

Interconnections among main elements - I

Interconnections among main elements - II
Hardware architecture: main elements

- **Line cards**
  - support input/output processing and rx/tx
  - store packets
  - adapt packets to the internal format of the switching fabric
  - support data link protocols
  - classify packets
  - schedule packets
  - support security
- **Switching fabric**
  - transfers packets from input ports to output ports

Hardware architecture: main elements

- **Control processor/network processor**
  - runs routing protocols
  - computes routing tables
  - manages the overall system
- **Forwarding engines**
  - compute the packet destination (lookup)
  - inspect packet headers
  - rewrite packet headers

Cell-based synchronous routers

- **Packet**: variable-size data unit
- **Cell**: fixed-size data unit

- **ISM**: Input-Segmentation Module
- **ORM**: Output-Reassembly Module

Switching fabric

- **Our assumptions**:
  - Bufferless
    - to reduce internal hardware complexity
  - Non-blocking
    - it is always possible to transfer in parallel from input to output ports any non-conflicting set of cells

Switching fabric

- **Examples**:
  - Buses
  - Shared memory
  - Crossbar
  - Multi-stage
    - rearrangeable Clos network
    - Benes network
    - Batcher-Banyan network (self-routing)
  - Buffered cross-bars (not considered)

Memory and speedup

- **Input queues**
- **Output queues**

Symbols:
- $S_{in}$: input switch
- $S_{out}$: output switch
- $S$: switching fabric
- $N$: number of inputs or outputs

Diagram:
- Cells switch (fabric)
- Packet: variable-size data unit
- Cell: fixed-size data unit
- ISM: Input-Segmentation Module
- ORM: Output-Reassembly Module
- Switching fabric examples: Buses, Shared memory, Crossbar, Multi-stage (Clos, Benes, Batcher-Banyan, not considered)

Diagram:
- Memory and speedup
- Input/output queues
- Switching fabric
- $S_{in}$, $S_{out}$
- $N$ inputs, $N$ outputs
Router/switch architectures

Speedup

- The speedup (increase in speed with respect to line speed) determines switch performance:
  - \( S_{in} = \) reading speed from input queues
  - \( S_{out} = \) writing speed to output queues
- The speedup is also a technological constraint
- Maximum speedup factor:
  - \( S = \max(S_{in}, S_{out}) \)

Faster and faster

- Need for high performance routers
  - to accommodate the bandwidth demands for new users and new services
  - to support QoS
  - to reduce costs
- Moore’s law (electronic packet processing power) is too slow with respect to the increase in link speed
- The bottleneck is memory speed

Single packet processing

- The time to process one packet is becoming shorter and shorter
- Worst case: 40-Byte packets (ACKs)
  - 3.2 \( \mu \)s at 100 Mbps
  - 320 ns at 1 Gps
  - 32 ns at 10 Gps
  - 3.2 ns at 100 Gbps
  - 320 ps at 1 Tbps

Switches with queues at outputs

- OQ (Output Queued)
  - The switching fabric is able to transfer to any output all cells received in one time slot
  - 100% throughput
  - Optimal average delay
  - Speedup \( N \) with respect to line speed is required in switching fabric speed and in output port memory access

Switches with queues at inputs

- IQ (Input Queued)
  - Switching constraints
    - at most one cell for each input and for each output can be transferred
  - Advantages:
    - Switching fabrics and memories less costly
    - No speedup required in the switching fabric
    - Memory access speed equal to line speed
    - Speedup=1
  - Only viable solution for very high speed devices

IQ switches

- Two problems:
  - Define scheduling algorithms to avoid output contention
    - Select in each time slot data to be transmitted from inputs to outputs
    - Constraint: in each time slot
      - At most 1 data from each input (no speedup in reading)
      - At most 1 data to each output (no speedup in writing because no memory is available)
IQ switches

- Two problems:
  - Memory architecture:
    - If using FIFO queues, HoL (Head of the Line) blocking
    - If choosing cyan packet from queue N, the red packet in queue 1 is blocked by the cyan packet which is at the head of queue 1 even if red output port N is free
  - For uniform unicast traffic throughput limited to 58.6%

Memory architecture in IQ

- To avoid HoL blocking phenomenon, a more complex memory architecture is needed

- Two possible solutions:
  - p-window queueing
  - VOQ (Virtual Output Queueing)

Memory architecture

- p-window queueing:
  - p is the window size
  - The first p cells of each queue are considered for scheduling
  - Higher complexity
    - Scheduler deals with pN cells
    - Non FIFO queues
  - HoL blocking reduced, completely eliminated only for \( P \to \infty \)

- VOQ (virtual output queueing):
  - At each input data are stored in separate queues according to data destination (N queues for each input)
  - \( N^2 \) queues in total
  - Eliminates HoL blocking and it permits to achieve 100% throughput with a proper scheduling algorithm

Scheduling problem modelling

- Problem to be solved by the scheduling can be represented using a bipartite graph

- An edge \( i \to j \) exist if there is at least a data at input \( i \) willing to reach output \( j \)
- Edges may be weighted
  - Weight
    - Binary
    - Queue length
    - HoL cell age or waiting time
    - Other metrics
- The scheduling algorithm tries to determine, in each time slot, a matching over the bipartite graphs.
- Select at most N edges with constraints
  - At most one edge for each input
  - At most one edge for each output
Router/switch architectures

Scheduling problem modeling

• Another possible representation is based on a (Request Matrix RM), which stores the information related to data transfer request.

\[
\begin{array}{cccc}
0 & 5 & 0 & 0 \\
3 & 0 & 4 & 0 \\
1 & 0 & 0 & 2 \\
0 & 0 & 0 & 8 \\
\end{array}
\]

0: no data from input to output
1: at least one data from input to output (may be weighted)

• Matching $\Rightarrow$ it is a permutation matrix i.e., a matrix such that the row and column sum is at most equal to 1

\[
\begin{array}{cccc}
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 \\
\end{array}
\]

Scheduling in IQ switches

Traffic description

• $A_{ij}(n) = 1$ if a packet arrives at time $n$ at input $i$, with destination reachable through output $j$
• $\lambda_{ij} = E[A_{ij}(n)]$
• An arrival process is admissible if:
  - $\sum \lambda_{ij} < 1$
  - $\sum \lambda_{ij} < 1$
  • no input and no output are overloaded on average
  • OQ switches exhibit finite delays (for admissible traffic)
• Traffic matrix: $\Lambda = [\lambda_{ij}]$

Scheduling policies: objective

• Let us consider a N x N IQ
• Denote by $i$ the input port index and by $j$ the output port index
• Goal: assuming infinite buffer size, transfer any admissible traffic pattern with no losses
• Solutions are known
  - If traffic pattern is known in advance
    • TDM of Birkhoff von Neumann algorithm
  - For admissible unknown traffic pattern
    • Maximum Weight Matching
    • Maximum Size Matching
• Several heuristics are proposed for unknown traffic pattern
  - iSLIP, iLQF, iOCF, 2DRR (WFA), MUCS, RPA, and many others

Scheduling uniform known traffic

• A number of algorithms give 100% throughput when admissible traffic is uniform
  - For example:
    • TDM and a few variants
    • iSLIP (see later)

Example of a TDM schedule for a 4x4 switch

\[
\begin{array}{cccc}
\cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot \\
\end{array}
\]

Birkhoff - von Neumann theorem

• Any doubly stochastic matrix $\Lambda$ can be expressed as convex combination of permutation matrices $\pi_n$:

\[
\Lambda = \sum_{n} a_n \pi_n
\]

• with
  - $a_n \geq 0$
  - $\sum_n a_n = 1$
Scheduling non-uniform known traffic

• Thanks to the Birkhoff–von Neumann theorem

• If the traffic is known and admissible, 100% throughput can be achieved by a TDM scheme using:
  – for a fraction of time \( a_1 \) matching \( M_1 \) \( (\pi_1) \)
  – for a fraction of time \( a_2 \) matching \( M_2 \) \( (\pi_2) \)
  – for a fraction of time \( a_k \) matching \( M_k \) \( (\pi_k) \)

MSM: Maximum Size Matching

• MSM maximizes the number of data transferred in a single time slot, i.e. select the maximum number of edges
  – Instantaneous throughput maximization.

• Asymptotic computational complexity is \( O(N^{2.5}) \)

• Non optimal algorithm
  – Some admissible traffic pattern cannot be scheduled, i.e. it does not always achieve 100% throughput.

MSM: example

\[
\begin{array}{ccc|c|c|c}
I_1 & I_2 & I_3 & O_1 & O_2 & O_3 \\
\hline
1 & \lambda & 0 & 0 \\
2 & 0 & \lambda & 0 \\
3 & \lambda & \lambda & 0 \\
\end{array}
\]

\( \lambda = 0.5 \)

Three MSM are possible in overload, i.e. when all queues are full

When the first matching is chosen, the maximum throughput cannot be achieved

MWM: Maximum Weight Matching

• A weight is associated with each edge

• The MWM, among all possible \( N! \) matchings, selects the one with the highest weight (sum of edge metrics)

MWM: Maximum Weight Matching

• MWM does not maximize instantaneous throughput (worse than MSM)

• It was demonstrated that a MWM algorithm
  – in IQ switches with VOQ architecture
  – under admissible traffic
  – with infinite queue size
  – when using as weight either the queue length or the age of the HoL data
    achieves 100% throughput

• Asymptotic computational complexity \( O(N^3) \)

• With finite queue size, it behaves similarly to MSM

• Problems with delays and possible starvation

Practical solutions

• Need to define heuristics
  – with reasonable complexity
  – that can be implemented in hardware

• Any scheduling algorithm defines three aspects:
  – A method to compute the weights to be associated with each edge (metric):
    – Approximate MSM
    – Binary (queue occupancy)
    – Approximate MWM
    – Queue length (it is an indication of the fact that the queue, which is associated with an input/output pair, is suffering)
    – Age of HoL data
    – Interface load
    – Ad hoc metrics to select critical edges/nodes
Practical solutions

- A heuristic algorithm to determine a matching
- A contention resolution algorithm among edges with the same metric:
  - round-robin (initial choice state dependent)
  - sequential search (initial choice non state dependent)
  - random

\[\text{i-SLIP}\]

- Iterative algorithm
- It defines a heuristic algorithm which determines, with a proper number of iterations, a maximal size matching (i.e., a matching that cannot be further extended with other edges selection)
- Metric is the queue occupancy
- To solve contentions, it exploits an arbiter for each input and for each output

\[\text{i-SLIP: counters}\]

- Each input (output) has a pointer associated with to solve contentions
- The output pointer is incremented, modulo N, by one unit beyond the index of the input to which the grant was issued
- The input pointer is incremented, modulo N, by one unit beyond the index of the output from which an accept was received

\[\text{i-SLIP: properties}\]

- For uniform overload the bipartite graph is a full mesh the higher the number of alternatives, the easier is to determine a good matching. iSLIP degenerates to a TDM scheme

For one iteration unsatisfactory

\[\text{i-SLIP: properties}\]

- IT 1
  - All outputs grant input 1
  - Input 1 accepts output 1

- IT 2
  - Outputs grant input 2
  - Input 2 accepts output 2
Router/switch architectures

### i-SLIP: properties

- At the end:
  - Each iteration has a computational complexity of $O(N^2)$, but it can be easily made parallel.
  - Worst case in one iteration: 1 edge is selected.
  - When executing N iterations, the matching is maximal (depends on the choice made but cannot be extended).
  - However, the computational complexity is $O(N^2)$.
  - Experimental results show that $\log_2 N$ iterations are in general enough to obtain good performance.
  - Performance drops if pointers are badly synchronized.
  - iSLIP was implemented on a single chip in the Cisco 12000 router.

### iSLIP: extensions

- Use the same heuristic algorithm (3-phase) but with different metrics:
  - Queue length: $L_{QF}$
  - HoL cell age: $C_{OF}$
- Input send requests containing the weight.
- Contentions are solved using the weight first, only for equal weights the choice is random.
- Does not exploit pointer synchronization to obtain good performance, rather the edge weight.
- OCF has better delay properties (never starves data), but the increase in complexity is significant and makes the algorithm practically unfeasible.

### 2DRR: Two Dimensional Round Robin

- Operates on the request matrix.
- Extension of the WFA (Wave Front Arbiter), very easily implementable in hardware.
- Definitions:
  - Generalized diagonal is a set of N elements of a matrix $N \times N$ such that two elements do not belong to the same row or column.
  - A set of N diagonal is said to be covering if each element of the matrix belongs to one and only one diagonal.
  - In each time slot, the algorithm goes through N iterations.
- At the beginning, all links (input-output connections) are enabled.
- At each iteration, a given generalized diagonal is chosen:
  - Only enabled links may be selected if the are covered by the elements belonging to the chosen diagonal.
  - If a link from input i to output j is selected, all requests issued by i or sent to j are disabled for the current time slot (cannot be chosen in the matching).
- In N iterations, all N generalized diagonal are considered and the request matrix is fully covered.
Router/switch architectures

2DRR: example

2DRR: Two-Dimensional Round Robin

• At each time slot, a different covering set of generalized diagonal is chosen, to improve fairness
  – Indeed, edges covered to the first diagonal chosen are more likely selected
  – Round robin over different sets of covering diagonal and round robin on each element in the set
• Emulates a MSM
• Not easy to extend to other metrics
• Asymptotic computational complexity $O(N^2)$

Traffic scenarios

• Uniform traffic
  – Bernoulli i.i.d. arrivals
  – usual testbed in the literature
    • “easy to schedule”
  – Diagonal traffic
    – Bernoulli i.i.d. arrivals
    – critical to schedule, since only two matchings are good

Uniform traffic

• Comparison between MWM, iSLIP, iLQF, RPA

Issues in IQ switches

• RPA achieves 98% throughput, iLQF 87%, iSLIP 83%
• Signalling:
  – Signalling bandwidth required to transfer weights from inputs to the controller may be significant with respect to the available bandwidth in the switching fabric
  – The more complex the adopted metric, the larger the signalling bandwidth required
  – Differential signalling may be adopted
• Multiple classes:
  – Given K classes, first the VOQ architecture must be extended, by using KN queues at each input
  – Scheduling algorithms must be extended to support priorities
Issues in IQ switches

• QoS (fair queueing)
  - Scheduling for QoS (need to serve the most urgent packet) has a difficulty interaction with the scheduling to transfer data from inputs to outputs
  - Need to balance performance and fairness
  - No ideal optimal solution known

• Frame scheduling
  - Operate on a frame of length F slot, and compute a schedule on the frame and not on a slot by slot basis
  - Scheduling algorithm executes only at frame boundaries
  - Relatively easy to provide QoS guarantees for each input-output pair
  - Delay increases at low loads

Example of a critical traffic pattern

Router/switch architectures

Issues in IQ switches

• Multicast:
  - 2nd possible different multicast flows
  - May be treated as unicast through input port replication (often named multicopy)
    - At each input a number of copies equal to the packet fanout are created, for the proper outputs, and inserted in the proper VOQ
    - Speedup required
    - Increases the instantaneous input load
    - May lead to low throughput (unable to sustain a single broadcast flow)
  - Scheduling for multicast must be defined to exploit switching fabric multicast capabilities
    - Balance fanout splitting and no-fanout splitting
    - Often a single FIFO for multicast is proposed (Hol blocking, less critical with respect to unicast)
  - Critical traffic patterns when few inputs are active

• Variable packet length support
  - May introduce packet scheduling instead of cell scheduling
    - Packets transferred as trains of cells
      - An edge is selected when the first cell of a packet arrives and is kept in all the following matchings until the last cell of the packet is transferred
  - It avoids reassembly machines at outputs
  - Same throughput guarantees
  - Packet delay may be larger or shorter
  - Depends on packet length distribution

Issues in IQ switches

• MC-VOQ architecture
  - 2nd separate queues at each input
  - Best possible solution (no Hol blocking)
    - An optimal scheduling was defined (only theoretically)
    - Implies re-enqueuing and out-of-sequence
  - However, admissible traffic pattern exist that cannot be scheduled in an IQ switch regardless of the queue architecture and of the scheduling algorithm
  - Scalability problem
  - Number of queues
  - Scheduling algorithm
  - Manage a finite number of queues
  - CIOQ switches (a moderate speedup helps a lot)

References