Architectures for IP routers and scheduling algorithms

Class on
Switching technologies for data centers
2018/19

Andrea Bianco, Paolo Giaccone

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

- IP routers
- OQ routers
- IQ routers
  - Scheduling
  - Optimal algorithms
  - Heuristic algorithms
  - Packet-mode algorithms
  - Networks of routers
  - QoS support
- CIOQ routers
- Multicast traffic
- Conclusions
The slides marked RWP are reproduced with permission of Prof. Nick McKeown from the Electrical Engineering and Computer Science Dept. of Stanford University (CA, USA)
Outline

- IP routers
- OQ routers
- IQ routers
  - Scheduling
  - Optimal algorithms
  - Heuristic algorithms
  - Packet-mode algorithms
  - Networks of routers
  - QoS support
- CIOQ routers
- Multicast traffic
- Conclusions
“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:
- low number of ports at very high speed (Mbps/Gbps)
- very high throughput
- few services
Basic architecture

Control Plane

Datapath
per-packet processing

Routing protocols
Routing table

Forwarding Table  Switching
Basic functions

- **Routing**
  - computation of the output port of an incoming packet (forwarding)
  - uses the routing tables computed by the routing protocols
  - can be a complex procedure:
    - very large routing tables
    - dynamic variation of routes in the Internet
Basic functions

- **Switching**
  - transfer of packets from input ports to output ports
  - solution of the contentions for output ports
    - queueing methods
      - where to store
    - scheduling methods
      - what to transfer
Faster and faster

➢ Need for high performance routers
  ▪ to accommodate the bandwidth demands for new users and new services
  ▪ to support QoS (over-provisioning)
  ▪ to reduce costs with respect to a cloud of smaller size routers (maybe)
    • a smaller number of fibers is needed
    • a smaller number of devices (but it is less costly?)
    • may be more energy-hungry
  ▪ to ease of the management task
Packet processing grows similarly to link speed and router bandwidth

- packet processing (CPU-based) grows as Moore law
  - 2x / 18 months (i.e. 1.6x / 12 months)
- link speed follows Butter’s law
  - 2x / 9 months
- premium end-users’ connection speed follow Nielsen's law
  - 1.5x / 12 months
- overall bandwidth for high-speed routers bandwidth grows similarly
  - 2.2x / 18 months
Memory as bottleneck

- Memory access time improves very slowly with respect to packet processing and communication speeds
  - 1.1x / 18 months
- Performance bottleneck is often memory speed
  - not enough fast memories to store the packets
- Also power consumption may limit the performance due to the hardware heating and thus to cooling problems
  - with CMOS technologies, each transmitted bit across a switching fabric requires some energy, due to the variation of the voltage level in the gate
  - power grows proportionally to the data rate
The time to process one packet is becoming shorter and shorter

- worst case: 40-Byte packets (ACKs) travelling over the Internet
  - 3.2 µs at 100 Mbps
  - 320 ns at 1 Gbps
  - 32 ns at 10 Gbps
  - 3.2 ns at 100 Gbps
  - 320 ps at 1Tbps
Hardware architecture

physical structure

logical structure
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
Interconnections among main elements - I

control processor & forwarding engine

switching fabric

line card 1

line card N
Interconnections among main elements - II
Interconnections among main elements - II

control processor

switching fabric

line card & forwarding engine

------

line card & forwarding engine
Cell-based routers

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

- *packet*: variable-size data unit
- *cell*: fixed-size data unit

Cell switch (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
Switching fabric

- We do not discuss switching fabrics with internal buffers
  - e.g.: crossbars with buffer at each crosspoint
  - very interesting tradeoff between implementation complexity and control complexity
Generic switching architecture

Input 1

S_{in}

switching fabric

S_{out}

Output 1

Input N

S_{in}

input queues

output queues

Output N
The speedup limits the switch performance

- $S_{\text{in}} =$ reading speed from input queues
- $S_{\text{out}} =$ writing speed to output queues

Because of memory bottleneck, the main performance limit may be the maximum speedup factor:

$$S = \max(S_{\text{in}}, S_{\text{out}})$$
Performance comparison

- The performance of different switching systems can be studied
  - with analytical models
    - introducing simplifying assumptions, but obtaining general results
  - with simulation models
    - obtaining more detailed results
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:
  - $\sum_i \lambda_{ij} < 1$
  - $\sum_j \lambda_{ij} < 1$
    - that is, no input and no output are overloaded on average
- traffic matrix: $\Lambda = [\lambda_{ij}]$
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

\[ \Lambda = \frac{\rho}{N} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \]

\[ \Lambda = \frac{\rho}{3} \begin{bmatrix} 2 & 1 & 0 & 0 \\ 0 & 2 & 1 & 0 \\ 0 & 0 & 2 & 1 \\ 1 & 0 & 0 & 2 \end{bmatrix} \]
Traffic scenarios

- **LogDiagonal traffic**
  - Bernoulli i.i.d. arrivals
  - more critical than uniform, less than diagonal traffic

\[ \Lambda = \frac{\rho}{2^N - 1} \begin{bmatrix}
8 & 4 & 2 & 1 \\
1 & 8 & 4 & 2 \\
2 & 1 & 8 & 4 \\
4 & 2 & 1 & 8
\end{bmatrix} \]
Outline

- IP routers
- OQ routers
- IQ routers
  - Scheduling
  - Optimal algorithms
  - Heuristic algorithms
  - Packet-mode algorithms
  - Networks of routers
  - QoS support
- CIOQ routers
- Multicast traffic
- Conclusions
Output Queued (OQ) switches

- $S_{in} = 1$  \quad $S_{out} = N$
- used for low bandwidth routers
  - no coordination among ports
  - work-conserving
    - best average delays
  - complete control of delays
    - support of QoS scheduling
- **Property**: OQ switches exhibit finite delays under any admissible traffic
Output Queued (OQ) switch

speedup N

Input 1 → Output 1
Input N → Output N

switching fabric
Note: OQ is optimal from the point of view of average delay and throughput.
Stability, throughput and delays

- Hp: stationary system, infinite queue
  \[ \lambda_{\text{in}} \rightarrow \square \rightarrow \lambda_{\text{out}} \]

- for a particular \( \lambda_{\text{in}} \)
  - stable \( \iff \) finite occupancy \( \iff \) finite delays \( \iff \)
  \[ \lambda_{\text{in}} = \lambda_{\text{out}} \]

- 100% throughput \( \iff \) stable under any \( \lambda_{\text{in}} \) admissible
Stability, throughput and delays

- \( \lambda_{\text{out}} \leq \lambda_{\text{max}} \)
- \( \text{stable}(\lambda_{\text{in}}) \iff \lambda_{\text{in}} = \lambda_{\text{out}} \Rightarrow \lambda_{\text{in}} \leq \lambda_{\text{max}} \)
  - hence, \( \lambda_{\text{max}} \) is
    - maximum achievable throughput
    - maximum offered load for stability
- \( \text{unstable} (\lambda_{\text{in}}) \iff \lambda_{\text{in}} > \lambda_{\text{max}} \)
  - queue grows with rate \( \lambda_{\text{in}} - \lambda_{\text{max}} \)
Outline

- IP routers
- OQ routers
- IQ routers
  - Scheduling
  - Optimal algorithms
  - Heuristic algorithms
  - Packet-mode algorithms
  - Networks of routers
  - QoS support
- CIOQ routers
- Multicast traffic
- Conclusions
Simple Input Queued (IQ) switches

- $S_{in} = 1 \quad S_{out} = 1$
- 1 FIFO queue for each input port
- throughput limitations
  - due to head of the line (HOL) blocking
- scheduling
  - to solve contentions for the same output
Head of the Line (HOL) Blocking

RWP
Simple IQ switch performance

![Graph showing Simple IQ switch performance with normalized load and delay metrics. The graph illustrates the performance of Simple IQ and OQ under uniform traffic with a notable point at 2 - \(\sqrt{2}\) ≈ 58%.]
Single IQ switch

- Using a simple Markov chain model
  - 2x2 $\rightarrow$ throughput 0.75
    - states: (2,0), (1,1)
  - 3x3 $\rightarrow$ throughput ?
    - states: (3,0,0), (2,1,0), (1,1,1)
Bufferless switch

Throughput = $1 - \left(1 - \frac{p}{N}\right)^N \rightarrow 1 - e^{-1} \approx 0.63$

- uniform i.i.d. Bernoulli arrivals
- input load $p$
Improving IQ switches performance

- Window/bypass schedulers
  - the first $w$ cells of each queue contend for outputs
  - HOL blocking is reduced, not eliminated
  - $w = 1$ means FIFO at each input
  - higher complexity
    - the scheduler deals with $wN$ cells
    - non-FIFO queues
Improving IQ switches performance

- Maximum throughput in an $N \times N$ switch with variable window size $w$

<table>
<thead>
<tr>
<th>$N$</th>
<th>$W=1$</th>
<th>$W=2$</th>
<th>$W=3$</th>
<th>$W=4$</th>
<th>$W=5$</th>
<th>$W=6$</th>
<th>$W=7$</th>
<th>$W=8$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.75</td>
<td>0.84</td>
<td>0.89</td>
<td>0.92</td>
<td>0.93</td>
<td>0.94</td>
<td>0.95</td>
<td>0.96</td>
</tr>
<tr>
<td>4</td>
<td>0.66</td>
<td>0.76</td>
<td>0.81</td>
<td>0.85</td>
<td>0.87</td>
<td>0.89</td>
<td>0.91</td>
<td>0.92</td>
</tr>
<tr>
<td>8</td>
<td>0.62</td>
<td>0.72</td>
<td>0.78</td>
<td>0.82</td>
<td>0.85</td>
<td>0.87</td>
<td>0.88</td>
<td>0.89</td>
</tr>
<tr>
<td>16</td>
<td>0.60</td>
<td>0.71</td>
<td>0.77</td>
<td>0.81</td>
<td>0.84</td>
<td>0.86</td>
<td>0.87</td>
<td>0.88</td>
</tr>
<tr>
<td>32</td>
<td>0.59</td>
<td>0.70</td>
<td>0.76</td>
<td>0.80</td>
<td>0.83</td>
<td>0.85</td>
<td>0.87</td>
<td>0.88</td>
</tr>
<tr>
<td>64</td>
<td>0.59</td>
<td>0.70</td>
<td>0.76</td>
<td>0.80</td>
<td>0.83</td>
<td>0.85</td>
<td>0.86</td>
<td>0.88</td>
</tr>
<tr>
<td>128</td>
<td>0.59</td>
<td>0.70</td>
<td>0.76</td>
<td>0.80</td>
<td>0.83</td>
<td>0.85</td>
<td>0.86</td>
<td>0.88</td>
</tr>
</tbody>
</table>
Improving IQ switches performance

- Virtual output queueing (VOQ)
  - one queue for each input/output pair
    - N queues at each input
    - $N^2$ queues in the whole switch
  - eliminates HOL blocking
  - used in high-bandwidth routers
    - scheduling implemented in hardware at very high speed
IQ switches with VOQ

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

- IP routers
- OQ routers
- IQ routers
  - Scheduling
  - Optimal algorithms
  - Heuristic algorithms
  - Packet-mode algorithms
  - Networks of routers
  - QoS support
- CIOQ routers
- Multicast traffic
- Conclusions
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

Graph

Matching

scheduler
Implementing schedulers

- Scheduling is a complex task
  - a scheduling algorithm can be implemented in hardware if:
    - it shows good performance for a wide range of traffic patterns
    - it can be efficiently parallelized
    - it can be efficiently pipelined
    - it requires few iterations (or clock cycles)
    - it requires limited control information
Many algorithms achieve 100% throughput under uniform traffic

- For example:
  - TDM and a few variants
  - iSLIP (see later)

Example of TDM for a 4x4 switch
Scheduling non-uniform traffic

- If the traffic is known and admissible, 100% throughput can be achieved by a TDM using:
  - for a fraction of time $a_1$ matching $M_1$
  - for a fraction of time $a_2$ matching $M_2$
  - for a fraction of time $a_k$ matching $M_k$
  - subject to $\sum_i a_i = 1$

- thanks to the Birkhoff - von Neumann theorem
Outline

- IP routers
- OQ routers
- IQ routers
  - Scheduling
  - Optimal algorithms
  - Heuristic algorithms
  - Packet-mode algorithms
  - Networks of routers
  - QoS support
- CIOQ routers
- Multicast traffic
- Conclusions
Maximum Weight Matching (MWM)

- among all the possible $N!$ matchings, selects the one with the highest weight (sum of the edge metrics)
  - MWM is generally not unique
- MWM is 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
  - cannot be parallelized and pipelined efficiently
- MWM has never been implemented in a commercial chipset
MWM is the optimal solution of the scheduling problem when the traffic is unknown, when the weight is either the queue length or the cell age:

- achieves **100% throughput** under any traffic
  - also under non-Bernoulli arrival processes, satisfying the law of large numbers
- achieves **low average delays**, very close to those of OQ switches
- possible starvation for lightly loaded packet flows
Let $T$ and $P$ be fixed

$D_t$ denotes the matching used at time $t$

The following variations of MWM also achieve 100% throughput:

- $D_t = \text{MWM}(t-P)$  
  MWM with pipeline degree $P$

- $D_t = \text{MWM}(\text{ceil}(t/T)\cdot T)$  
  MWM with latency $T$

- combinations of both

thus, it seems easy to achieve 100% throughput!
MWM with pipeline and latency

But:

- What about throughput?
  - 100% throughput 😊
    - but needs the computation of a MWM … 😞

- What about delays?
  - delays can be really bad! 😞
When scheduling in IQ switches, it is very difficult to achieve simultaneously:
- high throughput
- low delay
- limited implementation complexity
Maximum Size Matching (MSM)

- among all the possible matchings, selects the one with the highest number of edges (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
Instability of MSM

- Assume:
  - \( P(\text{arrival at } Q_{12}) = \lambda \)
  - \( P(\text{arrival at } Q_{11}) = P(\text{arrival at } Q_{22}) = 1-\lambda-\varepsilon \)
  - \( Q_{12} = B \gg 0 \quad Q_{11} = Q_{22} = 0 \)
  - in case of parity serve \( Q_{11} \) and/or \( Q_{22} \) instead of \( Q_{12} \)

- Observe:
  - \( Q_{12} \) is served only when \( A_{11} = 0 \) and \( A_{22} = 0 \), i.e. with probability:
    \[
    P(\text{serve } Q_{12}) = P(\text{no arrivals at both } Q_{11} \text{ and } Q_{22}) = [1-(1-\lambda-\varepsilon)]^2 = (\lambda+\varepsilon)^2
    \]
  - \( P(\text{serve } Q_{12}) < P(\text{arrival at } Q_{12}) \) if \( \varepsilon \) is small enough
  - Example: \( \lambda = 0.5; \ \varepsilon = 0.1; \)
    \( P(\text{serve } Q_{12}) = 0.36 \)

Note: this proof is due to I.Keslassy and R.Zhang, Stanford Univ.
MWM and MSM behave almost identically.

**Uniform Traffic**

![Graph showing mean delay vs normalized load for MWM and MSM](image)
MSM is somewhat inferior to MWM

LogDiagonal Traffic

MSM

MWM
Diagonal traffic

- MSM yields much longer delays than MWM at medium/high loads

Diagonal Traffic

![Graph showing mean delay vs. normalized load for MSM and MWM. MSM has significantly higher delay at medium and high loads compared to MWM.]
Outline

- IP routers
- OQ routers
- IQ routers
  - Scheduling
  - Optimal algorithms
  - Heuristic algorithms
  - Packet-mode algorithms
  - Networks of routers
  - QoS support
- CIOQ routers
- Multicast traffic
- Conclusions
Motivation

- strong interest in scheduling algorithms with
  - very low complexity
  - high performance

Usually

- implementable schedulers (low complexity)
  => low throughput, long delays
- theoretical schedulers (high complexity)
  => high throughput, short delays
Some implementable algorithms

- Approximate MSM
  - WFA, iSLIP, 2DRR, RC, FIRM and many others
- Approximate MWM with \( w_{ij} = X_{ij} \) (queue length)
  - iLQF, RPA, learning algorithms
- Approximate MWM with \( w_{ij} = \text{cell age} \)
  - iOCF
- Approximate MWM with \( w_{ij} = \sum_i X_{ij} + \sum_j X_{ij} \)
  - iLPF, MUCS
APPROXIMATIONS OF MAXIMUM SIZE MATCHING
Wave Front Arbiter

Requests

Match

RWP
Wave Front Arbiter

2N-1 steps

Requests

Match
Wrapped Wave Front Arbiter

N steps instead of 2N-1
iSLIP

- iSLIP means “iterative SLIP”
- iterates among the following 3 phases
  - Request
  - Grant
  - Accept
iSLIP

iSLIP demo

iSLIP

- 3 phases:
  - Request (from inputs to outputs)
    - each unmatched input sends a request to every output for which it has a cell
  - Grant (from outputs to inputs)
    - if an unmatched output receives requests, it sends a grant to one of the inputs
      - contentions solved by a round-robin mechanism
  - Accept (from inputs to outputs)
    - if an unmatched input receives grants, it selects a single output and it becomes matched to it
      - contentions solved by a round-robin mechanism
The round robin mechanism in iSLIP is designed so that, under uniform traffic, iSLIP emulates a dynamic TDM scheduler synchronized on the arrival pattern.
iSLIP is maximal
  • often, with log N iterations
  • always, with N iterations

iSLIP was implemented on a single chip in the Cisco 12000 router
APPROXIMATIONS OF MAXIMUM WEIGHT MATCHING
iLQF

- iLQF means “iterative Longest Queue First”
- iterates among the following 3 phases
  - Request
  - Grant
  - Accept
iLQF demo

3 phases:

- Request (from inputs to outputs)
  - each unmatched input sends all its queue lengths as requests to corresponding outputs

- Grant (from outputs to inputs)
  - if an unmatched output receives requests, it sends a grant to the input corresponding to the longest queue
    - contentions solved by random choice

- Accept (from inputs to outputs)
  - if an unmatched input receives grants, it selects the output with the longest queue
    - contentions solved by random choice
iLQF

- iLQF is maximal
  - often, with log N iterations
  - always, with N iterations
- iLQF is robust to non-uniform traffic
Uniform traffic

- comparison between MWM, iSLIP, iLQF, and RPA

![Graph showing the comparison between different traffic control algorithms (MWM, iSLIP, iLQF, RPA) under uniform traffic conditions. The x-axis represents the normalized load, and the y-axis represents the mean delay. The graph shows the performance of each algorithm at different load levels.](image-url)
LogDiagonal traffic

- iSLIP saturates close to 84% throughput
RPA achieves 98% throughput, iLQF 87%, iSLIP 83%
Outline

- IP routers
- OQ routers
- IQ routers
  - Scheduling
  - Optimal algorithms
  - Heuristic algorithms
  - Packet-mode algorithms
  - Networks of routers
  - QoS support
- CIOQ routers
- Multicast traffic
- Conclusions
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*

NO packet interleaving if *packet-mode*
Router based on an IQ cell switch: *packet-mode*

NO packet interleaving if *packet-mode*

ORMs can be removed
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
Questions

- can packet mode provide high throughput?
  
  YES! 😊

- what about delays?
  
  It depends…😊
Main theoretical results

- MWM in *packet-mode* yields 100% throughput
- *Packet mode* can provide shorter delays than *cell mode*, depending on the packet length distribution
Simulation scenario

- Router with ISMs and ORMs
- Uniform packet traffic
  - uniform packet load
  - uniform (1,192) packet size distribution
- Spotted packet traffic
  - non uniform packet load
  - bimodal (3,100) packet size distribution

\[
\Lambda_p = \begin{bmatrix}
1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\
0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\
1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\
1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\
1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\
0 & 1 & 1 & 1 & 0 & 1 & 0 & 1
\end{bmatrix}
\]
Uniform packet traffic

Packet mode and cell mode reach the same throughput
- Packet mode reaches higher throughput than cell mode
Effect of packet size distribution

- iSLIP delay\textsubscript{CM}/delay\textsubscript{PM} for different packet size distributions

At high load PM becomes better
Packet mode features

- Packet mode scheduling
  - is a feasible modification of schedulers
  - improves throughput
    - but it can generate some unfairness between long and short packets
      - inherent to all variable-packet networks without preemption
  - *may* give better packet delays than cell mode
    - depends on the packet size distribution
Outline

- IP routers
- OQ routers
- IQ routers
  - Scheduling
  - Optimal algorithms
  - Heuristic algorithms
  - Packet-mode algorithms
  - Networks of routers
  - QoS support
- CIOQ routers
- Multicast traffic
- Conclusions
Question:
- given a network of IQ switches running MWM and an admissible input traffic, is the network always stable?

NO! 😞

this is quite counterintuitive…but true
Consider the acyclic network of IQ routers in the following slide

- derived from well established results from adversarial queueing theory
- a very specific scenario, but comprises only few switches…
  - this situation may not be common, but cannot be excluded in real networks
Pathological network of IQ switches

Network with 8 switches and 4 flows
Instability of MWM

- If MWM is adopted at each IQ router, and the traffic is *admissible*, the system can be unstable under Bernoulli i.i.d. arrivals.
Instability of MWM

- MWM is too greedy, in the sense that it can create traffic bursts that are amplified by each scheduler.
- A server can be idling when large bursts (directed to it) are blocked because of the contentions upstream.
  - The problem arises when a packet flow is subject to priority changes along its path through the network.
Stability in networks of routers

- Global policies
  - “Oldest in the network” and many others
    - problem: requires global information about the network, and synchronized clocks at the ingress of the network
Stability in networks of routers

- Semi-local policies
  - MWM with local information about the router neighbors can achieve 100% throughput under i.i.d. Bernoulli arrivals
- Virtual network queue
  - The weights used by MWM are:
    - \( w_{ij} = \max\{0, X_{ij} - X_{\text{down-queue}(ij)}\} \)
    - where down-queue(ij) is the first downstream queue which is receiving packets from \( X_{ij} \)
Outline

- IP routers
- OQ routers
- IQ routers
  - Scheduling
  - Optimal algorithms
  - Heuristic algorithms
  - Packet-mode algorithms
  - Networks of routers
  - QoS support
- CIOQ routers
- Multicast traffic
- Conclusions
Problem:

- support rate guarantees, with admissible rate matrix

\[ R = \begin{bmatrix}
    r_{11} & \ldots & r_{1N} \\
    \ldots & \ldots & \ldots \\
    r_{N1} & \ldots & r_{NN}
\end{bmatrix} \]

with

\[ \sum_i r_{ij} \leq 1 \quad \sum_j r_{ij} \leq 1 \]
goal: find a sequence of matchings $M_k$ and their fraction of time $\phi_k$ such that the service given to all the queues satisfies $R$

$$R \leq \sum_{k} \phi_k M_k$$

$$\sum_{k} \phi_k = 1$$
Example:

\[ \phi_1 = \frac{1}{4}, \quad \phi_2 = \frac{1}{2}, \quad \phi_3 = \frac{1}{4} \]
How to decompose $R$?

- $R$ double substochastic
  - augmentation algorithm
  - $R'$ double stochastic such that $R' \geq R$
    - BvN algorithm
  - $R'$ decomposition
Augmentation algorithm

- If the sum of all elements in $R$ is less than $N$, then there exists an element $(i, j)$ such that $\sum_k r_{kj} < 1$ and $\sum_k r_{ik} < 1$

- Let $\epsilon = 1 - \max[\sum_k r_{kj}, \sum_k r_{ik}]$. Add $\epsilon$ to element $(i, j)$ of the matrix and obtain $R^*$. Now in $R^*$ the number of row sums and column sums that are < 1 is at least one less then in $R$

- Repeat this procedure until the matrix becomes double stochastic; this new matrix is $R'$
Let $R_1 = R'$ which is double stochastic. During iteration $i$:

- For a double $b$-stochastic matrix $R_i$ (i.e., all rows and columns sum to $b$) find a matching $M_i$ corresponding to all non-null elements of $R_i$.

- Compute $\phi_i$, the minimum value of $R_i$ corresponding to $M_i$, and

  $$R_{i+1} = R_i - \phi_i M_i$$

  which is still double $b'$-stochastic.

- If $R_{i+1} = 0$, then stop.

- Else, then a new iteration starts: $i = i + 1$. 
When the algorithm stops after $k$ iterations, we have a sequence of

$$\{(\phi_i, M_i)\}_{i=1}^k$$

From which we obtain the final **BvN decomposition**:

$$R \leq R' = \sum_{i=1}^k \phi_i M_i$$

where $\phi_i \in (0, 1]$ and

$$\sum_{i=1}^k \phi_i = 1$$
Outline

- IP routers
- OQ routers
- IQ routers
  - Scheduling
  - Optimal algorithms
  - Heuristic algorithms
  - Packet-mode algorithms
  - Networks of routers
  - QoS support
- CIOQ routers
- Multicast traffic
- Conclusions
CIOQ routers

VOQ

Input 1

Input N

Output 1

Output N

S

S

S

S

switching fabric

O₁

O_N
Question:

- if a low speedup $S$ is allowed (and queues are available at both inputs and outputs), is it possible to design simple scheduling algorithms, capable of achieving high throughput and low delay?

YES! 😊
a CIOQ switch achieves perfect OQ emulation if the departure order of all the packets from each output is the same as the emulated OQ

- it is impossible to distinguish, by observing arrivals and departures, if the switching architecture is CIOQ or OQ
- delays are perfectly controlled
  - easy to implement scheduling algorithms born for OQ (eg: WFQ)
Work conservation

- a CIOQ switch is **work-conserving** when each output is busy at the same time as the corresponding OQ switch
  - i.e., each output of the switch for which there are cells (either at the inputs or at the outputs) at the beginning of cell slot T is active at the end of the cell slot T
  - output never idling whenever a packet is present destined to it
  - good delay performance: same average delays as OQ
- note that OQ emulation implies work conservation but not vice versa
Speedup and performance

- speedup 4
  - exact OQ emulation

- speedup 2
  - exact OQ emulation
  - work conservation
    - same average delay than OQ
CIOQ routers with $S=2$

- If $S = 2$
  - easy to obtain 100% throughput
    - any maximal matching obtains 100% throughput
  - less easy to obtain work conservation
    - LOOFA algorithm
  - it is difficult to obtain perfect OQ emulation
    - stable marriage algorithm with special preference list
LOOFA

- Occupancy $o_j$: number of cells currently residing at the $j$-th output queue
  - at each time slot, $o_j$ is decremented by one because of departures

- Basic idea of LOOFA
  - Higher priority is given to outputs with lower occupancy, thereby attempting to maintain work-conservation for all outputs
If $S = 2$, during each of the two phases
   - each unmatched input selects a non-empty VOQ directed to the unmatched output with the lowest occupancy, and sends a request to that output
   - each unmatched output grants one request
     - the selection can be round robin, random, ...
   - repeat until the matching is maximal
LOOFA with $S=2$

- **TEO:**
  - LOOFA achieves work conservation if $S = 2$
OQ emulation with $S=4$

- **urgency** of a cell = departure time in OQ - current time
- MUCF (Most urgent cell first)

During each phase:
1. outputs request their most urgent cells from inputs
2. input grants output with the most urgent cell
3. loser output tries to obtain their next urgent cell
4. when no more matchings are possible, cells are transferred and the next phase starts
OQ emulation with $S=4$

OQ emulation and speedup 4

- **TEO:**
  - MUCF with speedup 4 obtains OQ emulation
CIOQ routers

- CIOQ are very promising architectures
  - many degrees of freedom in design
    - how to balance input/output buffers
    - how the buffers interact
      - e.g., by backpressure mechanisms
- Several currently designed architectures are supposed to be CIOQ
- Speedup $S$ is becoming closer and closer to 1 in practical implementations of new switching architectures ($\text{CIOQ} \rightarrow \text{IQ}$)
Outline

- IP routers
- OQ routers
- IQ routers
  - Scheduling
  - Optimal algorithms
  - Heuristic algorithms
  - Packet-mode algorithms
  - Networks of routers
  - QoS support
- CIOQ routers
- Multicast traffic
- Conclusions
Multicast traffic

Misleading idea:

- observe
  1. OQ can achieve 100% throughput under any admissible unicast and multicast traffic
  2. OQ can be perfectly emulated by CIOQ with $S = 2$
- then, with $S = 2$ it is possible to achieve 100% throughput for multicast traffic

WRONG! 😞

because observation 2 holds only for unicast traffic
Question:

- what is the minimum speedup required to achieve 100% throughput?

unknown! 😞
Multicast traffic

- Possible implementations
  - copy network before the switching fabric
    - a multicast cell with f destinations is treated as f cells
    - possible bandwidth inefficiency
  - dedicated queue
    - multicast packets are treated in some specific way
Multicast traffic: optimal queueing

- MC-VOQ queueing
  - best throughput performance
    - avoids HOL blocking
  - $2^N-1$ queues for each input, one for each fanout set
    - re-enqueuing process $\Rightarrow$ out-of-sequence problem
    - no re-enqueuing $\Rightarrow$ some throughput degradation
The optimal scheduling for multicast traffic can be defined similarly to unicast traffic:

- it is a sort of max flow algorithm on all \( N(2^N-1) \) queues

Many heuristics can be envisaged to approximate it.
3 main ingredients for IQ scheduling algorithms:

- Weight computation
- Matching computation
- Contention resolution
Weight computation
- obtains the priority of each input queue
- the metric can be related to queue length, waiting time of the cell at the HOL, …

Contention resolution
- whenever the selection is among situations with equal weights
- can be round robin, or random
Matching computation

- computes the matching, trying to maximize its total weight
- can be based on
  - an iterative search, like in iSLIP, iOCF, iLQF
  - a matrix greedy approach, like in MUCS, WFA
  - a reservation vector, like in RPA
  - a learning approach, like in APSARA
Summary

- Good IQ scheduling algorithms exist:
  - 100% throughput
  - short delay
  - limited complexity

- Performance differences are significant only close to saturation
Open questions concerning IQ schedulers:
- QoS guarantees
- stability of networks of switches
- multicast traffic
Router functions and architectures

Scheduling in IQ switches
References

Scheduling in IQ switches
References

Packet scheduling in IQ switches

Scheduling multicast traffic in IQ switches
References

Scheduling multicast traffic in IQ switches


QoS support in IQ switches


Advanced architectures derived from pure IQ

Scheduling in networks of IQ switches


