Queue and Flow Control Architectures for Interconnection Switches

1. Basic Concepts and Queueing Architectures
2. High-Throughput Multi-Queue Memories
3. Crossbars, Scheduling, & Combination Queueing
4. Flow and Congestion Control in Switching Fabrics

Manolis Katevenis

FORTH and Univ. of Crete, Greece

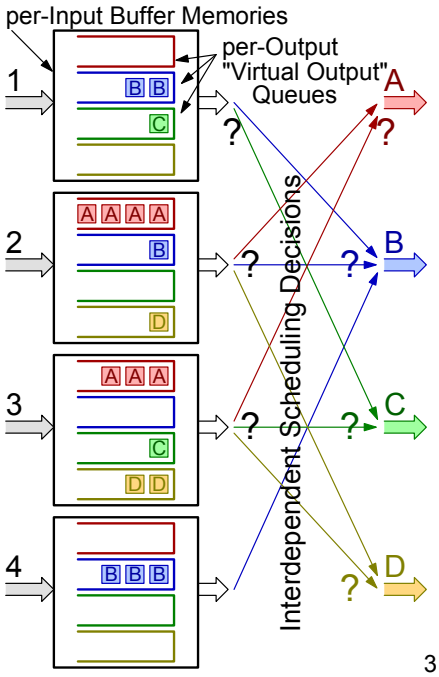
http://archvlsi.ics.forth.gr/~kateveni/534

3. Crossbars, Scheduling, & Combination Queueing

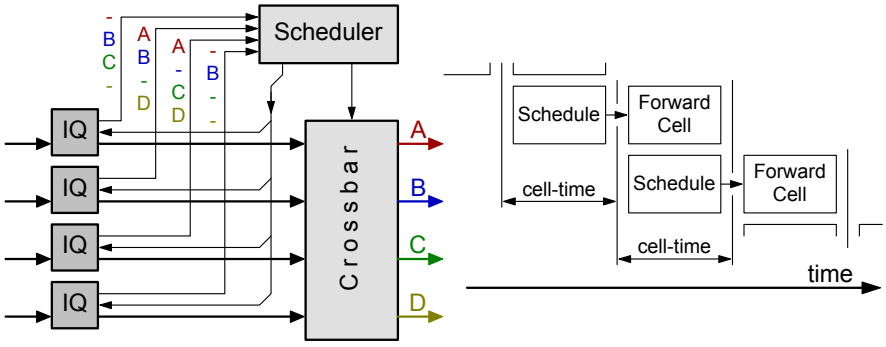
Table of Contents:

3.1 Input Q'ing w. VOQ (Virtual Output Queues)

- Input Queueing: most promising *scalable* switch architecture
  - provided multiple queues
  - provided efficient scheduler
- "Virtual Output Queues" (VOQ): per-output queues at the inputs
  - N queues/input (for $N \times N$ sw.)
  - N^2 queues total in all inputs
- Interdependent Scheduling
  - can each output decide alone? … no: may create input conflicts
  - can each input decide by itself? … no: may create output confl.

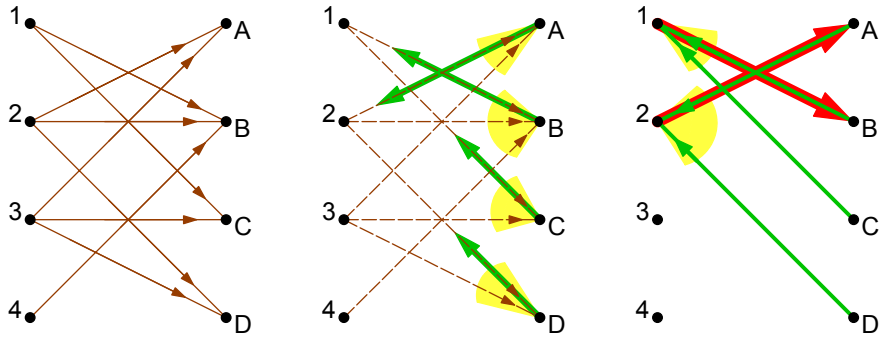


3.1 Crossbar Scheduling with VOQ's



- Example: 64 Byte = 512 bit cells, 10 Gbit/s line rate ⇒ 51 ns cell-time
  - if speedup of 2 to 3, like usually (see §3.2) ⇒ cell-time below 20 ns
- Most popular scheduler: *iSLIP* – McKeown, IEEE/ACM ToN, Apr. 1999
- ... based on the earlier "Parallel Iterative Matching" (PIM) scheduler – Anderson, e.a.: ACM Tr. on Computer Systems, Nov. 1993

Crossbar Scheduling: Parallel Iterative Matching (PIM)



Request Phase:

All inputs send their requests in parallel

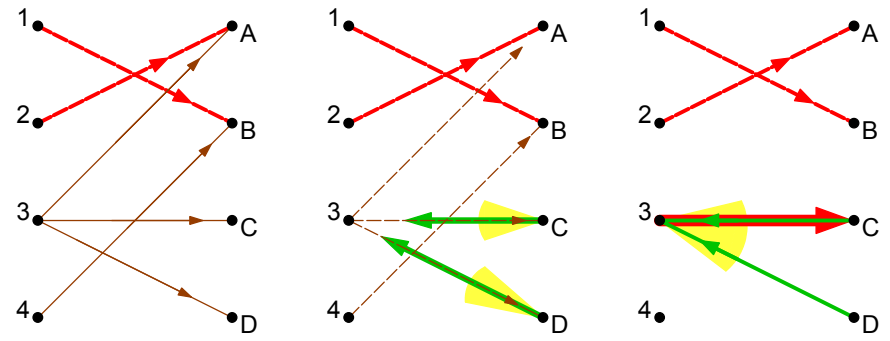
Grant Phase:

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



Request Phase:

Unmatched inputs (received no grant) resend their requests

Grant Phase:

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

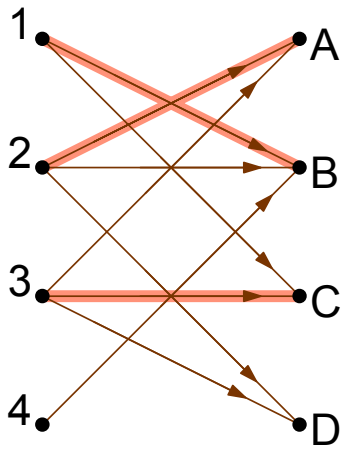
Accept Phase:

Unmatched inputs accept one of the received grants

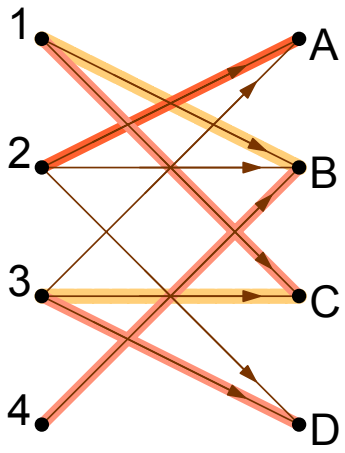
Second Iteration

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

Maximal Matching – Maximum Matching



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

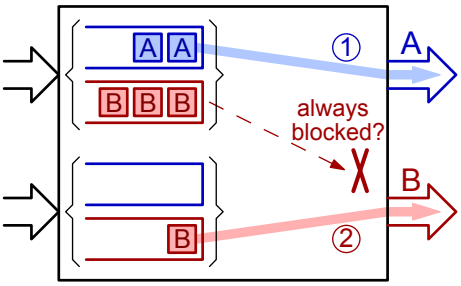


Maximum possible number of connections for the given request pattern

Maximum Matching is Complex

- *N*×*N* switch
- *R* requests; usually *R* is $O(N^2)$
- Deterministic Algorithm: $O(N \cdot (N+R)) \approx O(N^3)$
  - Tarjan: Data Structures & Network Algor., SIAM 1983
- Randomized Algorithm: $O(N+R) \approx O(N^2)$
  - Karp e.a: ACM STOC, 1990

… and may be unfair



- PIM progresses towards *maximal* matchings – not necessarily maximum.
- Original PIM, with random selections: each iteration resolves, on average, ≥3/4 of the remaining unresolved requests ⇒ $O(\log N)$ iterations.
- Hardware implementation of truly random selection, at high speed, with equal probability among (varying number of) choices, is unrealistic.

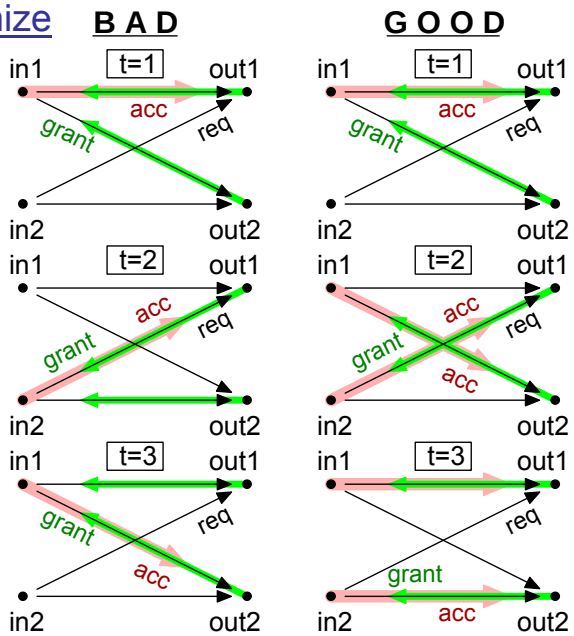
iSLIP: Practical, Popular Crossbar Scheduler

- Variation of PIM:
- Request Phase: same as PIM.
- Grant Phase:
  - PIM grants *randomly* to one of the requesting inputs.
  - iSLIP grants in *Round-Robin* order to the first requesting input after the previously marked input – careful which one that is: see next slide (the "Slip" idiom).
- Accept Phase:
  - PIM accepts *randomly* one of the granting outputs.
  - iSLIP accepts, in *Round-Robin* order, the first now-granting output after the output that was accepted last time.
- Relatively easy to implement, good fairness properties
- Was used in CISCO GSR-12000, Tiny Tera, Abrizio/PMC-Sierra, e.a.
  – Nick McKeown: IEEE/ACM ToN April 1999.

Slip to Desynchronize

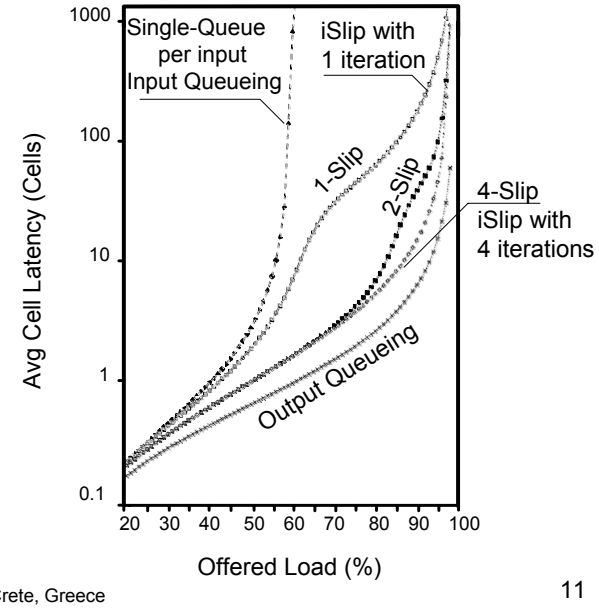
- Bad: grant round-robin to next request seen after last-time grant
- Good: grant rnd-robin to next request seen after last grant *that was accepted*

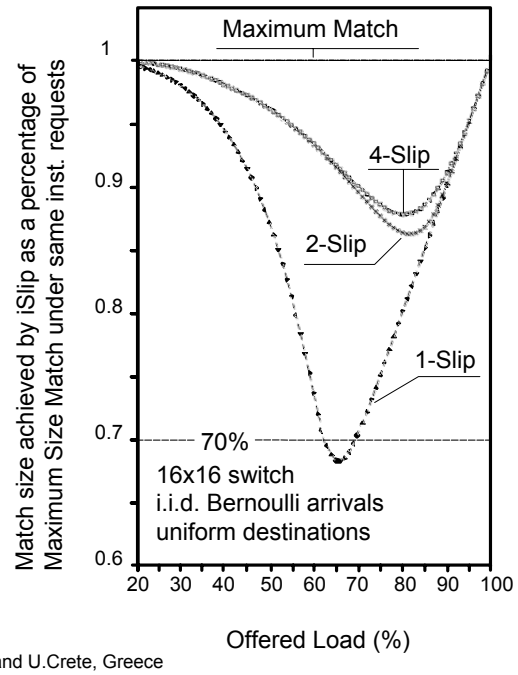
⇒ Under uniformly-destined traffic, when all VOQ's become non-empty, grant pointers get *desynchronized* and stay that way, moving all together, and yielding 100% output utilization



iSlip Performance under Uniform, Non-Bursty Traffic

- Source: McKeown, IEEE/ACM ToN, April 1999
- 16×16 switch
- i.i.d. Bernoulli arrivals (non-bursty traffic)
- uniformly distributed packet destinations
- *Beware!:* real traffic is very different from these assumptions – simulations under these assumptions are only useful to give the designer a "first, very rough idea"…

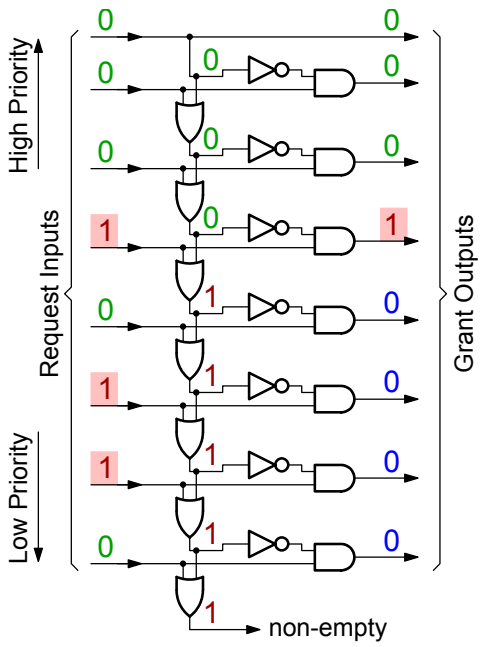




3.2 Linear Priority Circuit

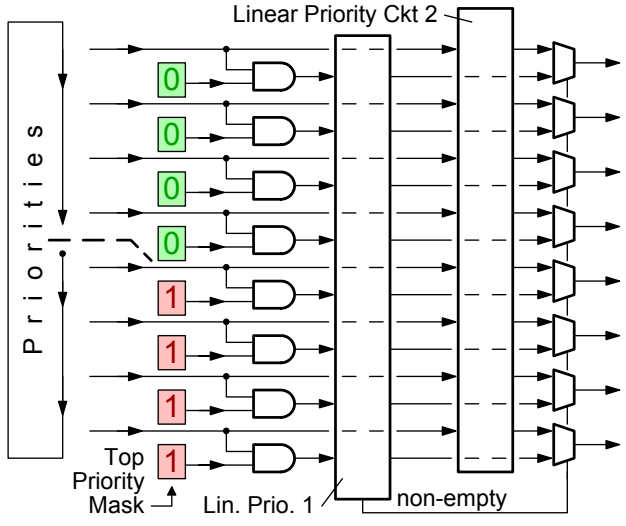
- *Input:* an ordered set of request flags
- *Output:* the same set of flags, where only the topmost asserted request remains ON, while all lower are cleared
- Output *i* is asserted iff input *i* is ON and none of the previous inputs, *0* to *i-1*, is ON:
  - Output[*i*] = Input[*i*] AND NOT ($OR_{k=0}^{i-1}$ Input[*k*])

⇒ *Delay* grows *logarithmicly* with the number of inputs
  - big OR functions built as trees of fixed-fan-in gates



Circular Priority

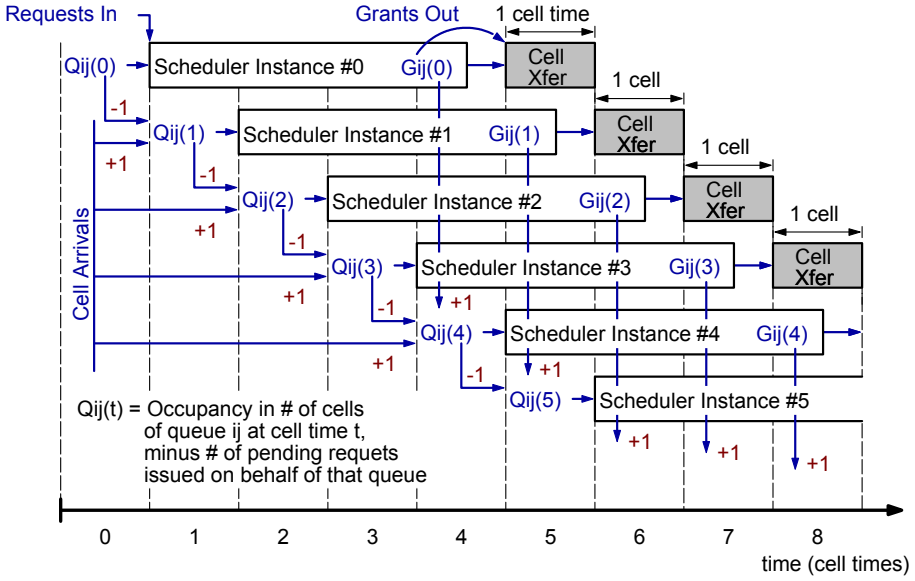
- Bit mask defines the top-priority position.
- First priority circuit finds topmost request in "half" the input request set, if any.
- Second priority ckt finds topmost request in other "half" of input set.



- Chose the first answer, if any, else chose the second

⇒ Circular priority (round-robin) delay grows w. logarithm of num. of inputs

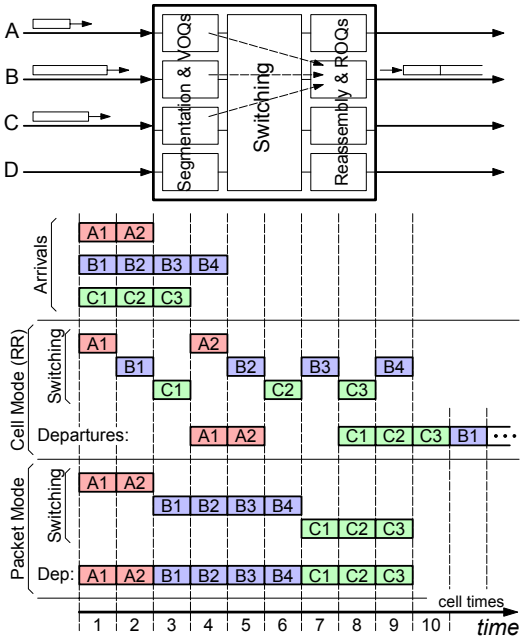
3.2 Pipelined Scheduling for Cell-based Crossbars



- Cell arrivals increment Qij (Qij = "guaranteed" min. occupancy of Q. i,j)
- When Qij ≥ 1 ⇒ issue a request Rij
- Requests issued decrement Qij, pending the transfer decision
- When a pending decision is resolved:
  - successful grants leave Qij as is (already decremented)
  - missed grants re-increment Qij (restore rejected request)
- Prerequisite: *fixed-size cells – interchangeable requests*
  - 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 –that was issued on behalf of the "second" cell– if granted, will result in transfering the "first" cell.
- Reference: Oki, Rojas-Cessa, Chao: "A pipeline-based approach for maximal-sized matching scheduling in input-buffered switches", IEEE Communications Letters, June 2001

3.2 Packet-Mode Scheduling

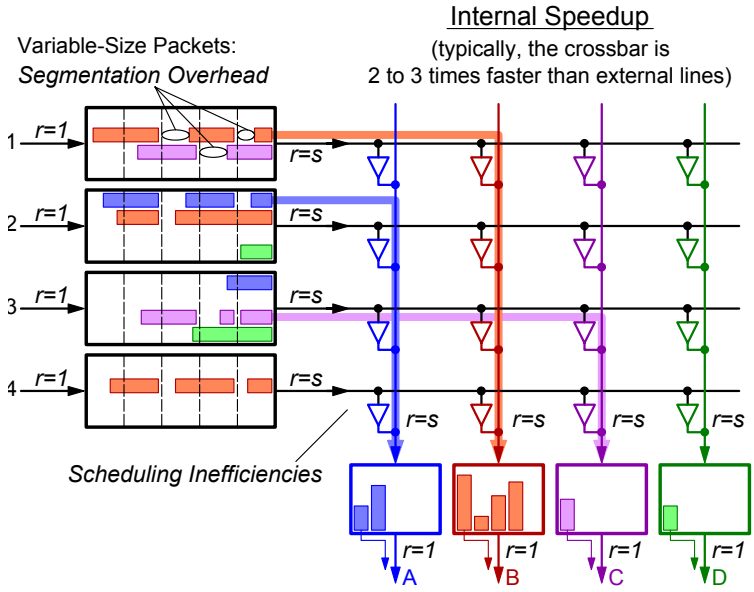
- Variable size packets segmented into fixed-size cells for VOQ and crossbar operation
- If cells of a packet are forwarded through the xbar non-consecutively:
  - increased delay
  - reassembly queues
  - no cut-through forward
- *Packet-mode* scheduling: forward all cells of a packet in consecutive cell-times



Packet-Mode Scheduling, Asynchronous Operation

- *Packet-Mode Scheduling:*
  - Maintain input-output pairings until full packet is forwarded;
  - Scheduler only considers candidacies for those inputs and outputs that are not currently busy in the "middle" of a packet;
  - Reduces delay, eliminates reassembly, allows cut-through;
  - Rare danger of pathological "locking": when a packet-mode "connection" completes, if all other ports are still busy and that flow's VOQ is non-empty, the same connection is made again.
  - Reference: Ajmone Marsan e.a.: IEEE/ACM ToN, Oct. 2002
- *Asynchronous Crossbar Operation:*
  - Schedule ports whenever freed – not necessarily in synchrony
  - Similar to packet-mode, but sched'ngTime >> pck-sz granularity
  - Reference: Passas, Katevenis: HPSR 2007

3.3 Combined Input-Output Queueing (CIOQ)



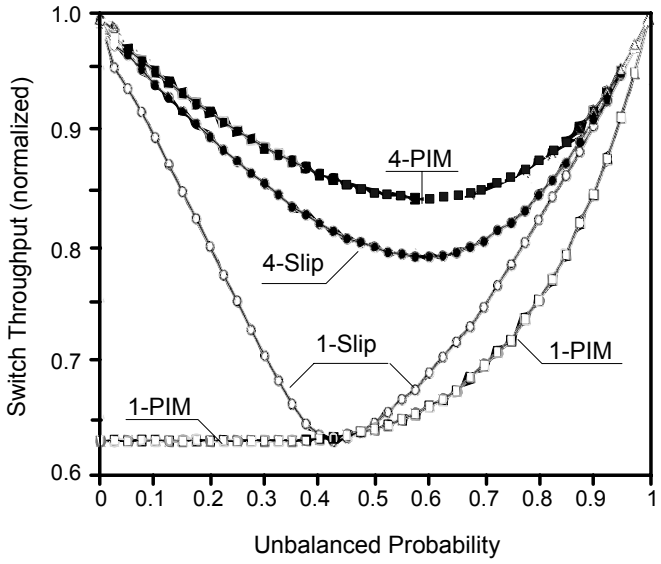
Internal Speedup – Combined Input-Output Queueing (CIOQ)

- Most widely used architecture in high-end internet switches
- Make the crossbar faster than the external lines, in order to:
  - compensate for the inefficiencies of the scheduler (e.g. unbal. traffic)
  - compensate for the segmentation overhead of variable-size packets
  - allow for separate (output) queues per QoS class
- Typical Speedup Factor values are between 2 and 3:
  - speedup of about 2 needed for variable-size packets (see § 2.2)
  - theoretical results: speedup of 2 suffices to emulate output queueing (using complex schedulers though – hard to totally unrealistic)
- The cost of Internal Speedup:
  - buffers at outputs too, increased throughput for crossbar & buffers
  - nowadays, increased throughput is too expensive (power consumpt'n) for off-chip communication ⇒ only use speedup *inside* switch chips, placing at least portion of input and output queues on-chip, with the rest of these queues on the line cards

Unbalanced Traffic: *simple example of hard traffic pattern*

- Each input has a "favored" output
  - "favored" input-output pairs are disjoint (they form a permutation)
- Each input sends traffic @ total rate = *load* as follows:
  - *(u × load)* to its favored output (u is the "unbalance factor), plus
  - *((1-u) × load)* to all outputs, uniformly distributed
  - ⇒ each output receives traffic @ total rate = *load*
- *"u"* is the "Unbalance Factor":
  - u = 0 % ⇒ totally uniform traffic (usually easy)
  - u = 100 % ⇒ totally directional traffic (permutation) (often easy)
  - u = intermediate ⇒ … usually hard traffic …

Crossbar Sched. Perf. under Unbalanced Traffic



- 32×32 switch
- Saturation Throughput simulations: load = 100%
- Source: Rojas-Cessa e.a: "CIXOB-k: combined input-crosspoint-output buffered packet switch", IEEE Globecom 2001

Can a CIOQ Sw. Emulate an Output Queued Switch?

- *Full Emulation:*

  consider a CIOQ switch (combined input-output queueing, with internal speedup), and an OQ switch, both as "black boxes". Consider precisely the same cells entering into both switches at precisely the same times. Full emulation is when the CIOQ switch will always forward to its outputs precisely the same cells as the OQ switch does, and at precisely the same times, for any arbitrary traffic pattern; i.e., an external observer is unable to tell which switch is which, no matter what traffic sequence (s)he injects.

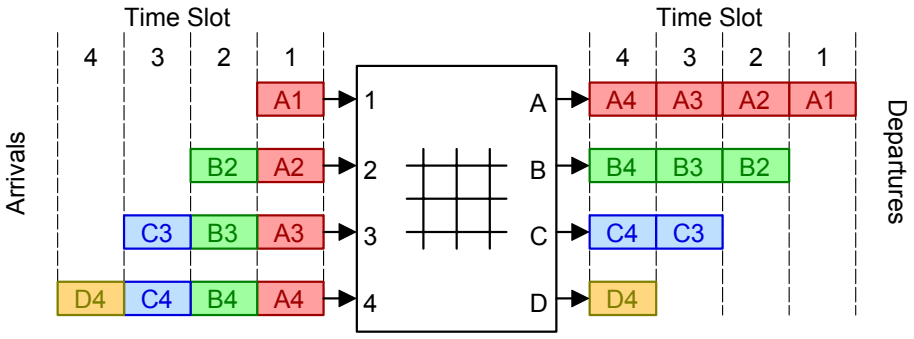
- *Work-Conserving Operation:*

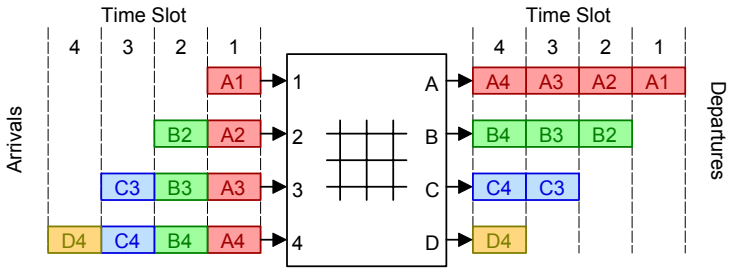
  no output port is ever left idle, except when there are no cells destined to it anywhere inside the switch. Hence, the outputs of the CIOQ switch will be busy (or idle) at precisely the same times as the corresponding OQ outputs, but *not necessarily* forwarding the exact same cell – may be forwarding another one of the cells destined to the same output (implies same *average* cell delay, due to "delay conservation" theorem for work-conserving switches).

Emulation of Output Q'ng by CIOQ with Speedup ≈ 2

- Results in IEEE JSAC, June 1999 (paper 1 by Chuang, Goel, McKeown, Prabhakar; paper 2 by Krishna, Patel, Charny, Simcoe):
- *Speedup = 2 - 1/N* is *necessary and sufficient* for a *N*×*N* CIOQ switch to *fully* emulate an OQ with FIFO output service
  - necessary: see next two slides
  - sufficient: need complex xbar scheduler – theoretical value only
- *Speedup = 2* is *sufficient* for CIOQ to *fully* emulate OQ with quite general service policies (PIFO – push-in first-out)
  - need complex crossbar scheduler – of theoretical value only
- *Speedup = 2* is *sufficient* for a CIOQ switch that uses *LOOFA* scheduler to be *Work-Conserving*
  - *LOOFA:* Lowest-Occupancy Output First Algorithm – maximal match where the shallowest output queues are connected first

Traffic pattern to prove the lower bound of s = 2 – 1/N of speedup that is necessary for CIOQ to emulate OQ





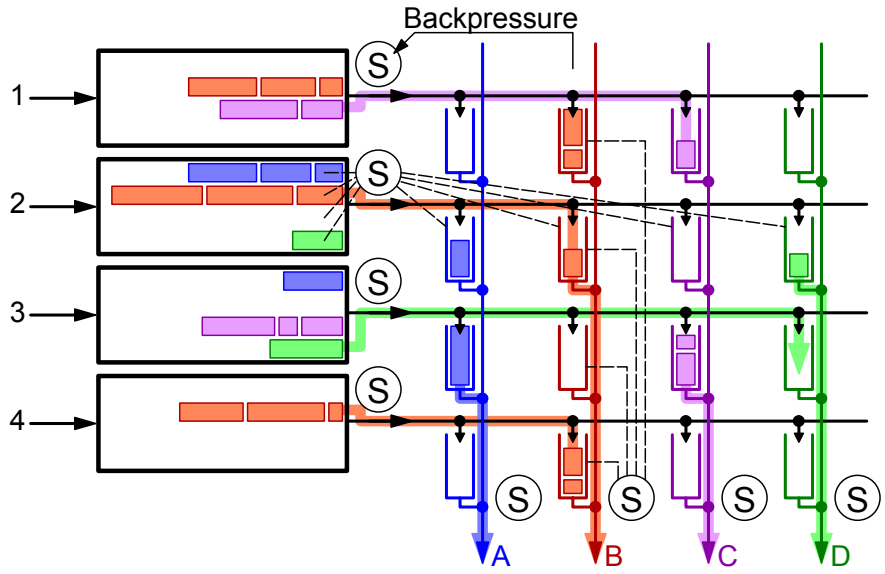
Phase	PA	PB	PC	PD
1	A1			
2				
3		B2		
4				
5			C3	
6				
7				D4

	PA	PB	PC	PD
	A1			
	A2			
		B2		
		B3		
			C3	
			C4	
				D4

	PA	PB	PC	PD
	A1			
	A2			
	A3	B2		
		B3		
		B4	C3	
			C4	
				D4

	PA	PB	PC	PD
	A1			
	A2			
	A3	B2		
	A4	B3		
		B4	C3	
			C4	
				D4

3.4 Buffered Crossbars (CICQ)



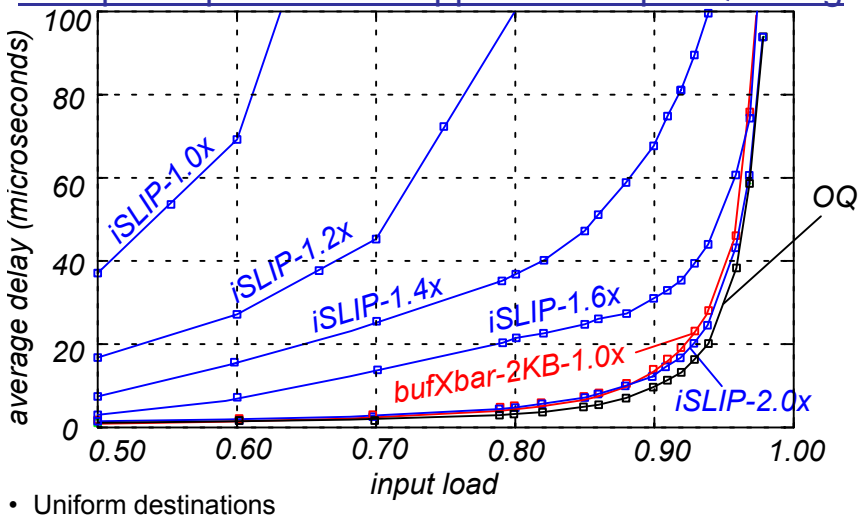
Buffered Crossbars, or Comb. Input-Crosspoint Q'ng

- *Small* buffers per crosspoint, large buffers per input
- Backpressure from crosspoint buffers to VOQ's at inputs
- Loosely-coupled, independent, single-resource schedulers
  - per-output schedulers decide which flow (crosspoint queue) to serve among the non-empty ones in each output's column
  - per-input schedulers decide which flow to serve among the ones with non-empty VOQ and with credits available in each input's row

⇒ Approximate "matchings" yield better scheduling efficiency

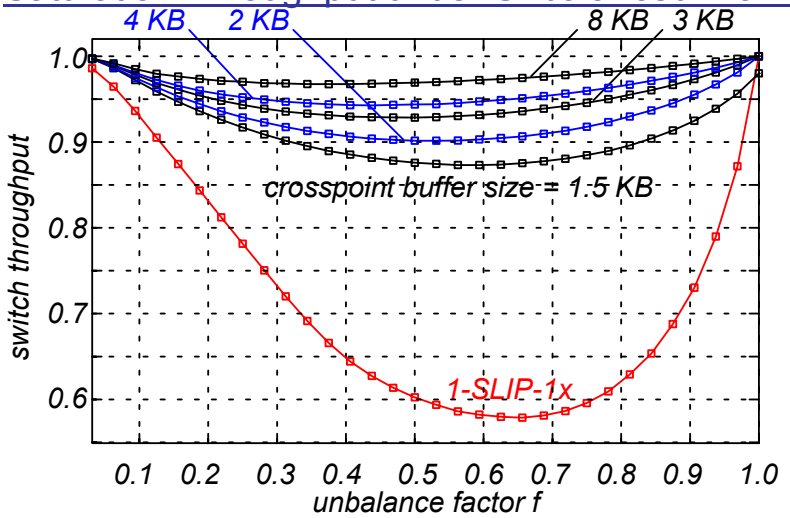
- in the short term, *(i)* multiple inputs may feed the same column (e.g. 2 and 4); *(ii)* multiple outputs may be fed by the same row (e.g. A and C)
- in the long run, these cannot persist, because *(i)* buffers in that column are filled faster than they get emptied, so they will fill-up; *(ii)* buffers in that row are being emptied faster than they get filled, so they will drain.

No Speedup needed to approach Output Queuing



- Uniform destinations
- Internet-style synthetic workload; 40-1500 byte packet sizes
- Unbuffered crossbar w. SAR: one-iteration iSLIP, 64-byte segments

Saturation Throughput under Unbalanced Traffic



- Poisson arrivals, Pareto sizes (40-1500)
- For iSLIP, packet sizes are multiples of 64 B (⇒ no SAR overhead)

Buffered Crossbars (CICQ) – References:

- D. Stephens, H. Zhang: "Implementing Distributed Packet Fair Queueing in a Scalable Switch Architecture", INFOCOM 1998
- T. Javidi, R. Magill, and T. Hrabik: "A High-Throughput Scheduling Algorithm for a Buffered Crossbar Switch Fabric", ICC 2001
- R. Rojas-Cessa, E. Oki, and H. Jonathan Chao: "CIXOB-k: Combined Input-Crosspoint-Output Buffered Switch", GLOBECOM 2001
- Abel, Minkenberg, Luijten, Gusat, Iliadis: "A Four-Terabit Packet Switch Supporting Long Round-Trip Times", IEEE Micro, Jan. 2003
- N. Chrysos, M. Katevenis: "Weighted Fairness in Buffered Crossbar Scheduling", IEEE Wrksh. High Perf. Switching & Routing (HPSR) 2003

⇒ M. Katevenis, G. Passas, D. Simos, I. Papaefstathiou, N. Chrysos: "Variable Packet Size Buffered Crossbar (CICQ) Switches", ICC 2004

- G. Passas, M. Katevenis: "Packet Mode Scheduling in Buffered Crossbar (CICQ) Switches", IEEE W.High Perf.Sw.Rtng (HPSR) 2006