

4.4 Internal Speedup & Related Topics

- Inefficiencies of Crossbar Scheduling
- Combined Input-Output Queueing (CIOQ) – Internal Speedup
- Variable-Size Packets – Segmentation and Reassembly
- Packet-Mode Scheduling
- Per-QoS-class output queues in CIOQ: outp. sched. for QoS
- Crossbars with Reconfiguration Overhead: Enveloppe Sched.
- Theoretical Questions:
 - can a CIOQ switch emulate Output Queueing?
 - with what internal speedup?

CS-534, Copyright Univ. of Crete

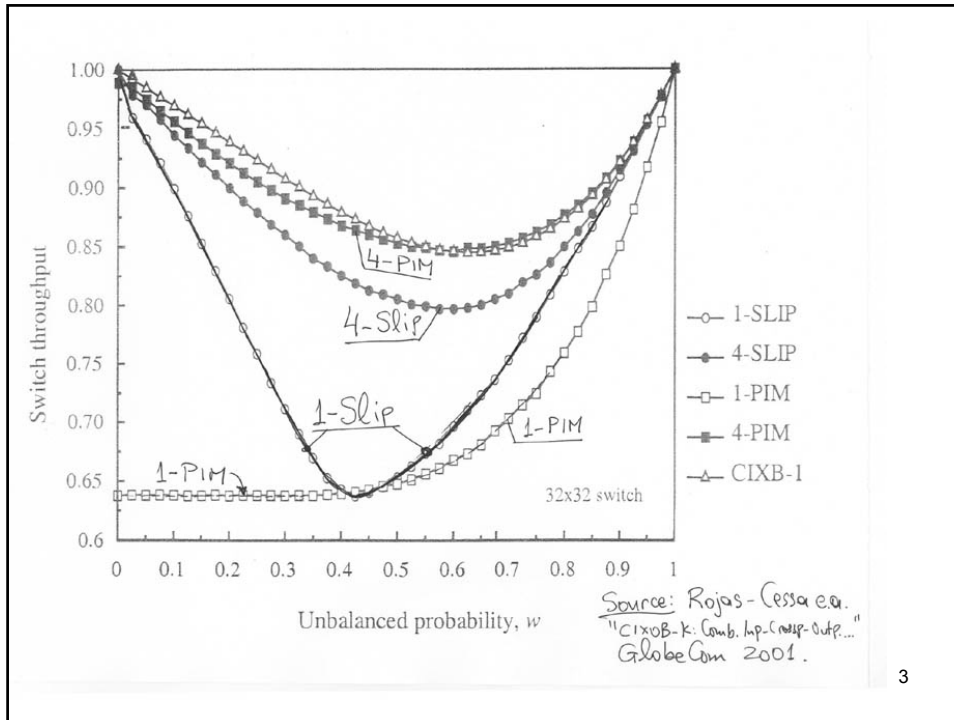
1

Unbalanced Traffic: a 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:
 - $(u \times load)$ to its favored output (u is the “unbalance factor”), plus
 - $((1-u) \times load)$ 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)
 - $u = \text{intermediate}$ ⇒ ... usually hard traffic ...

CS-534, Copyright Univ. of Crete

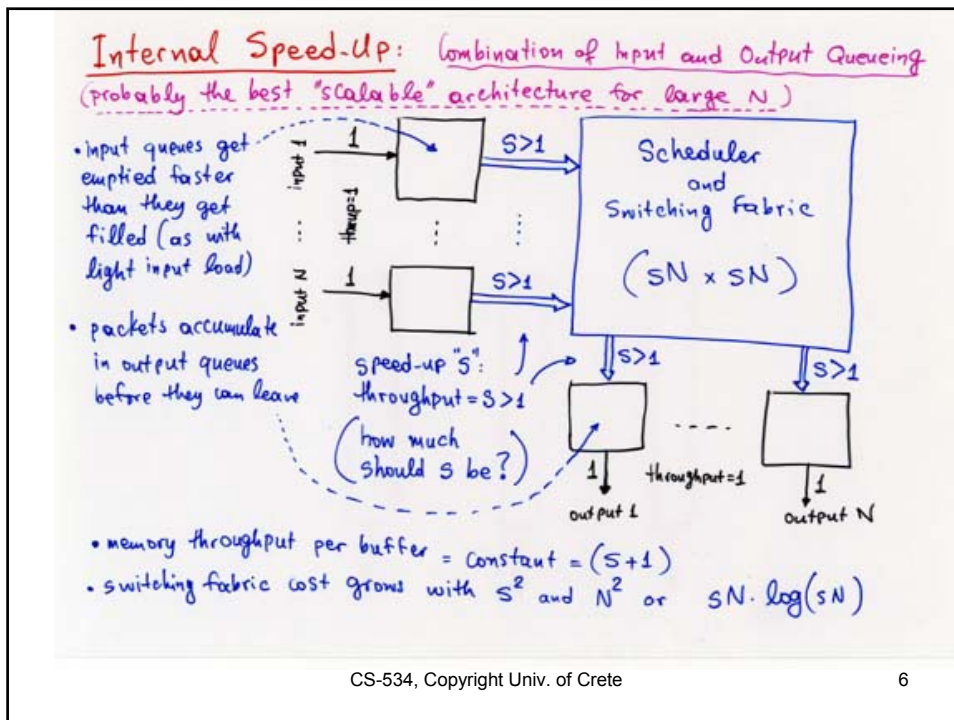
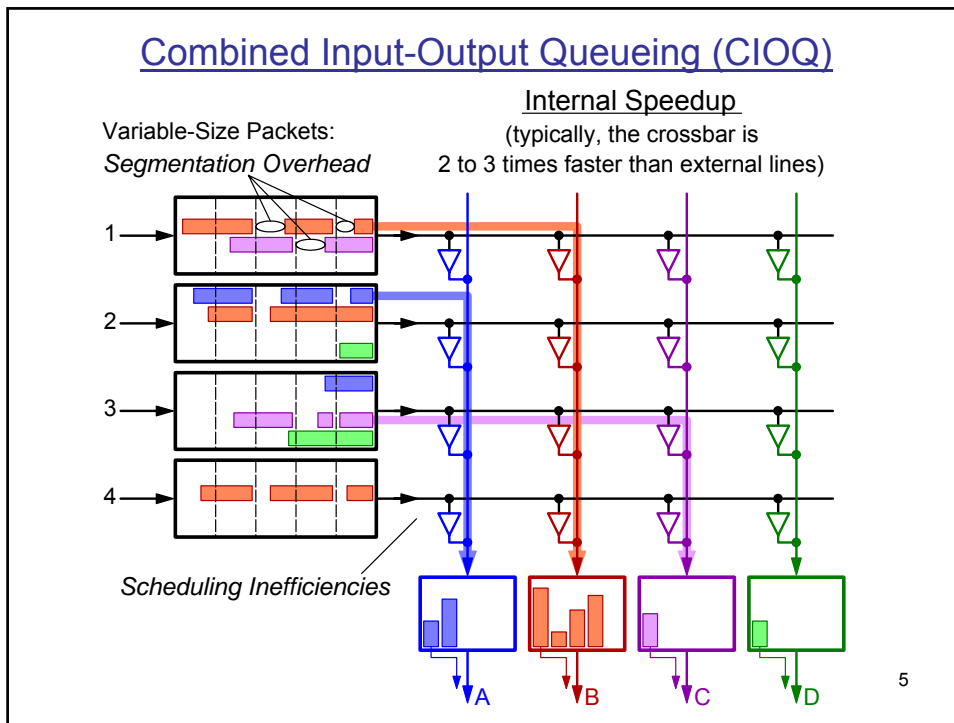
2



3

Internal Speedup Combined Input-Output Queueing (CIOQ)

- 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 exer. 4)
 - 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
 - for a given peak achievable internal throughput, the highest achievable line rate decreases in proportion to the speedup factor



Variable-Size Packets in Fixed-Size-Cell Crossbars

- Crossbars operate on a fixed-cell-time periodicity
 - need periodic time intervals at which to make scheduling decisions
 - need to reconfigure all input-output pairs at once (but see about “packet mode” scheduling, below)
 - we refer to “*unbuffered*” crossbars here – “buffered” crossbars, which contain small buffers at their crosspoints, can operate asynchronously (to be discussed later)
- Variable-size packets can be switched only after segmentation into fixed-size cells
 - Segmentation Overhead: partially filled last cell of packet
 - Reassemble packets after the crossbar – buffers & queues needed
 - Must store-and-forward – no cut-through possible any more

CS-534, Copyright Univ. of Crete

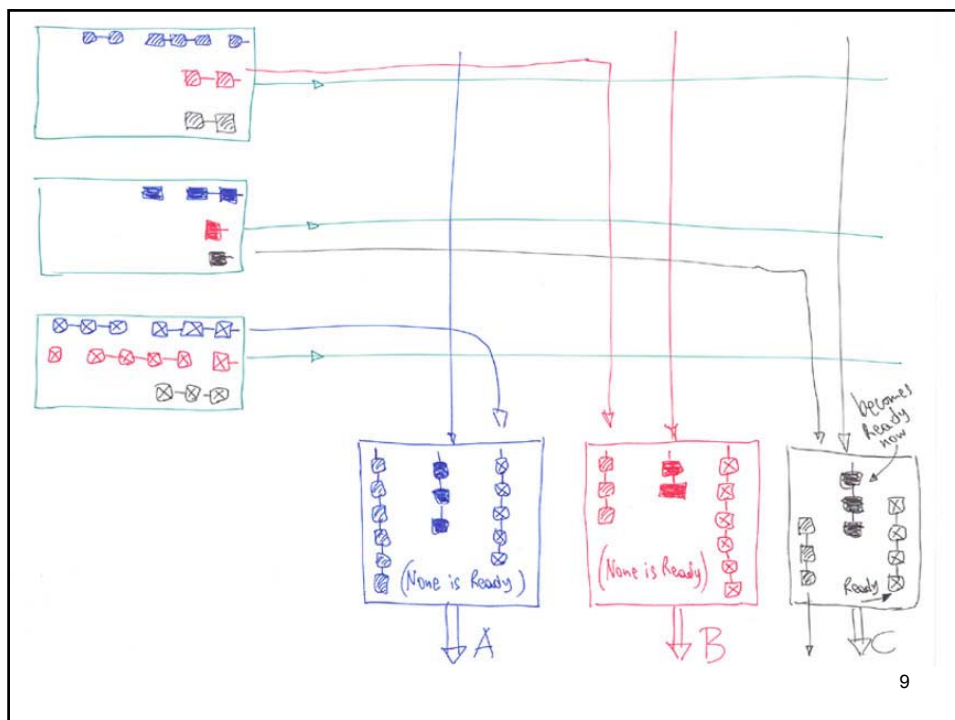
7

Segmentation and Reassembly (SAR): Reassembly Queues and Buffer Space needed

- One reassembly queue per input needed at each output
- Reassembly space per output, when this space is shared among all reassembly queues at that output:
 - a max-size packet per input may be almost complete at the output
 - after the first completed packet starts being forwarded to the output, outgoing rate \approx incoming rate \Rightarrow fixed total buffer occupancy
- Reassembly space, when partitioned per reassembly queue:
 - much more than with shared space:
 - after a first max-size packet, A, gets completed, and while it is being forwarded to the output (time \sim max.pck.size), one of the other reassembly queues (already containing almost one max. packet) may be receiving cells of total size \sim 1 max. packet, and so on...

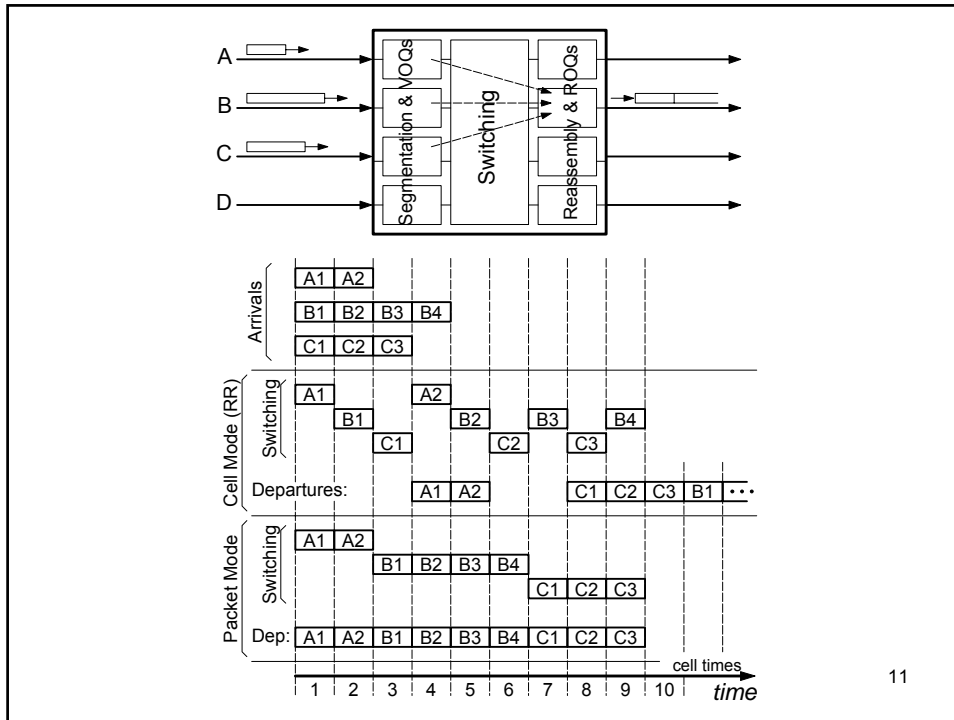
CS-534, Copyright Univ. of Crete

8



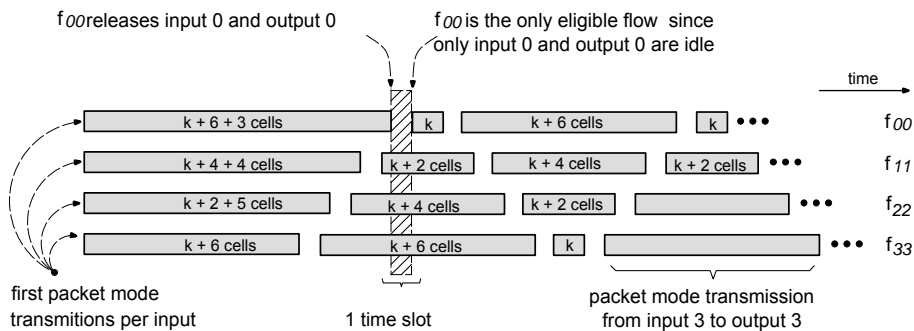
Packet-Mode Crossbar Scheduling

- Maintain input-output pairings until full packet is forwarded
- During each cell-time, scheduler only considers candidacies for those inputs and outputs that are not in the middle of a packet-mode "connection"
- No reassembly queues needed
- Cut-through forwarding is feasible
- *Reference: Ajmone Marsan e.a.: "Packet-Mode Scheduling in Input-Queued Cell-Based Switches", IEEE/ACM Tr. on Networking, Oct. 2002.*



11

Packet-Mode Scheduling: Danger of Pathological Locking in one Configuration



- during the cell-time when a packet transmission is completed at an input-output pair, if this is the only input-output pair that gets released and there are more packets in that VOQ, then the same pair will get re-connected \Rightarrow other flows are locked out

Switches with Reconfiguration Overhead: large, multi-packet "Enveloppes" in lieu of cells

- Reconfiguration Overhead
 - crossbars without central clock \Rightarrow rcvrs' resynchronization delay
 - optical crossbars
- Reduce reconfiguration frequency
 - \Rightarrow Increase cell time, i.e. increase cell size
 - call the large cells "Enveloppes"
 - Allow multiple segments from multiple packets in an envelope
 - enveloppes are formed in the VOQ's

CS-534, Copyright Univ. of Crete

13

Switches with Reconfiguration Overhead: Enveloppe Scheduling

- When to "release" an Enveloppe to the Switch?
 - partially filled enveloppes waste switch throughput
 - \Rightarrow wait for enveloppes to fill up
 - but if excessive waiting \Rightarrow too much delay introduced by the switch
 - \Rightarrow use some kind of timeout mechanism, or use a portion of the switch throughput to periodically "scan" all partially-filled VOQ's
- *Reference:* Kar e.a.: "Reduced Complexity Input Buffered Switches", Hot Interconnects 2000.

CS-534, Copyright Univ. of Crete

14

Can a CIOQ Switch with Internal Speedup Emulate an Output Queued Switch?

Recent theoretical results:
 IEEE JSAC, June 1999:
 (1) Chuang, Goel, McKeown, Prabhakar
 (2) Krishna, Patel, Charny, Simcoe

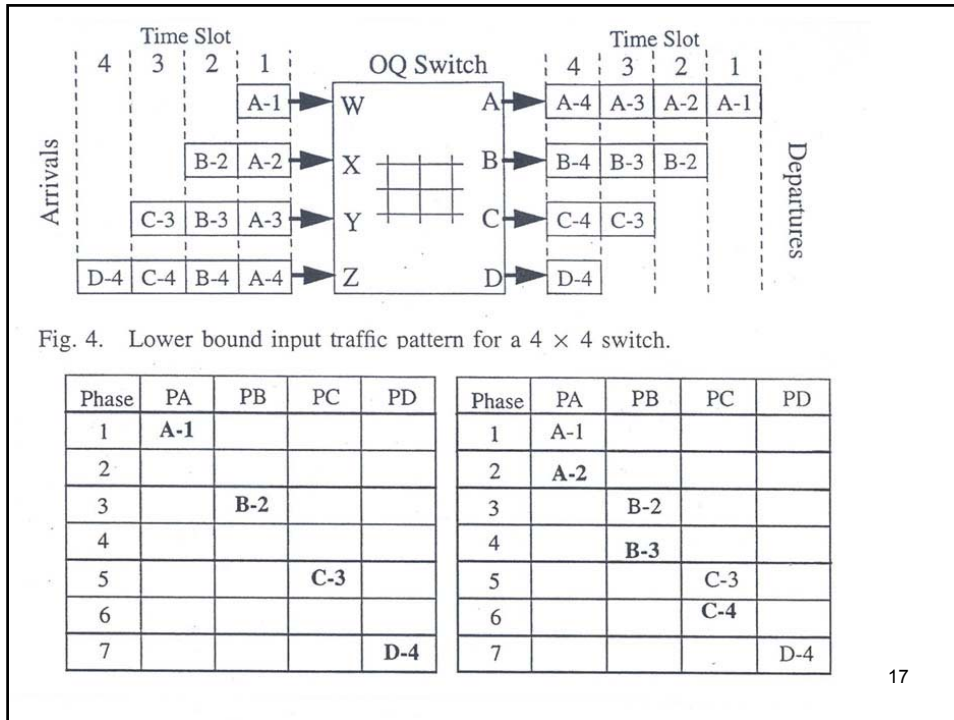
"CIOQ" = Combined
 Input & Output
 Queuing

- Full Emulation: consider a CIOQ switch and an OQ switch, with precisely the same cells entering into both switches, at precisely the same time. Full Emulation: precisely the same cells will depart from the CIOQ switch, as from the OQ switch, at precisely the same time.
- Work Conserving Operation: whenever cells destined to an output port exist in the switch, that output port will be busy transmitting one of them, i.e. an output port is never left idle, in the presence of traffic destined to it.

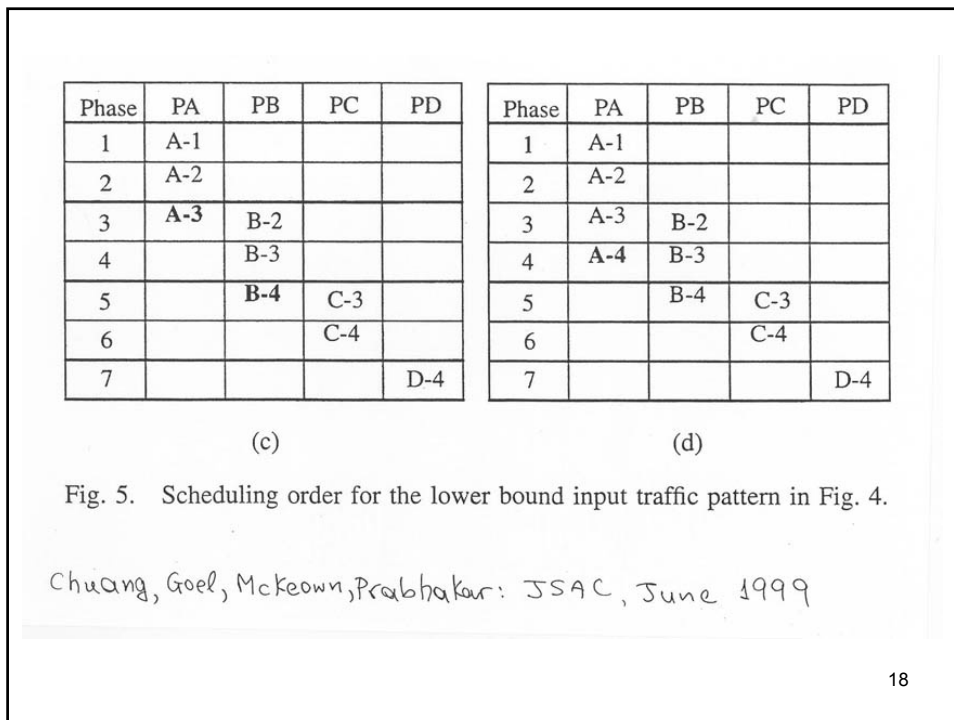
CS-534, Copyright Univ. of Crete 15

- $N \times N$ CIOQ switch to fully emulate an $N \times N$ OQ switch with FIFO output service policy:
 Speedup $S = 2 - \frac{1}{N}$ is necessary and sufficient
 see proof by counter example below ↑ need complex crossbar scheduler (theoretical value only)
- Speedup of $S = 2$ with a complex crossbar scheduling ← (based on:
 is sufficient to fully emulate an OQ switch with various (quite general) output scheduling policies ("PIFO - Push In First Out")
 "output cushion"
 "input thread"
 "slackness")
- speedup of $S = 2$ + LDDFA (Lowest Occupancy Output First Algorithm)
 crossbar scheduling is sufficient for Work Conserving CIOQ operation
 (LDDFA crossbar scheduling: maximal match that gives priority to those output ports that have the shallowest queues.

CS-534, Copyright Univ. of Crete 16



17



18

Summary of $m \times n$ Switch Architectures (link capacity = 1)

Architecture	maximum single-stream throughput	number of memories	Total memory throughput	Memory Space Utilization	Performance	Complexity Cost
Shared Buffer	$m+n$	1	$m+n$	BEST	BEST	high-thrup. single mem.
Output Queueing	$m+1$	n	$(m+1) \cdot n$	medium	BEST	large total mem. cost
Crosspoint Queueing	2	$m \cdot n$	$2 \cdot m \cdot n$	WORST	BEST	many memories
Internal Speed-up of s ($s > 1$)	$s+1$	$m+n$	$(s+1)(m+n)$	medium	very good (if...)	expensive switching fabric
inp. Buffering (Adv. inp. Q.)	2	m	$2m$	medium (sh. sp.) worst (part. sp.)	very good (if...)	scheduling algorithm
Input Queueing	2	m	$2m$	medium	WORST	(ok?)

CS-534, Copyright Univ. of Crete 19