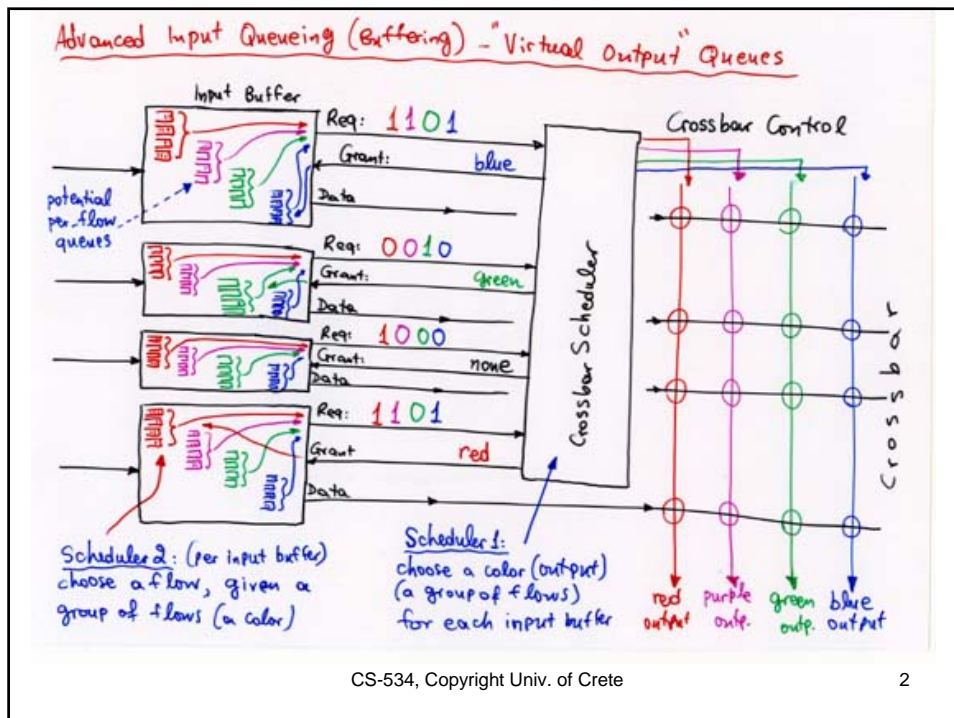## **4.3** Virtual Output Queue (VOQ) Input Q'ing and the Crossbar Scheduling Problem

- Crossbar Switch with one Buffer Memory per Input Line
- Throughput per Buffer Memory = 1 (incoming) + 1 (outgoing)
- Multiple (one per output) Queues per Buffer Memory:

  "Virtual Output Queues – VOQ"
    - *N* queues per input, N^2 queues total, for *NxN* switch
- Crossbar Scheduling, per cell-time:
    - pairings ("marriages") between inputs and outputs – each input specifies a subset of the outputs that it accepts to be married to
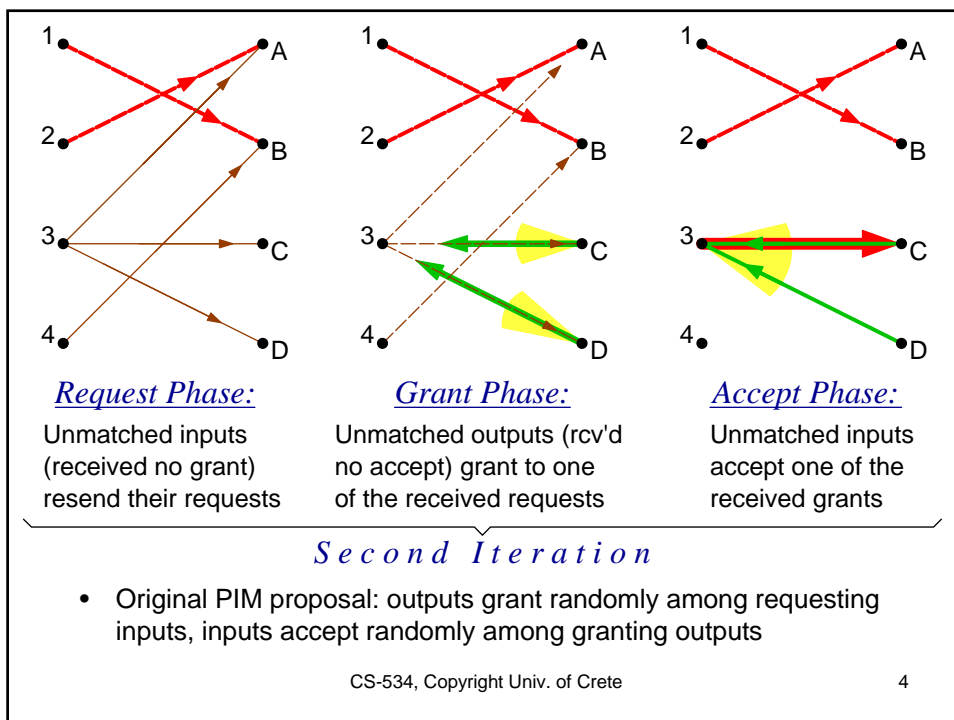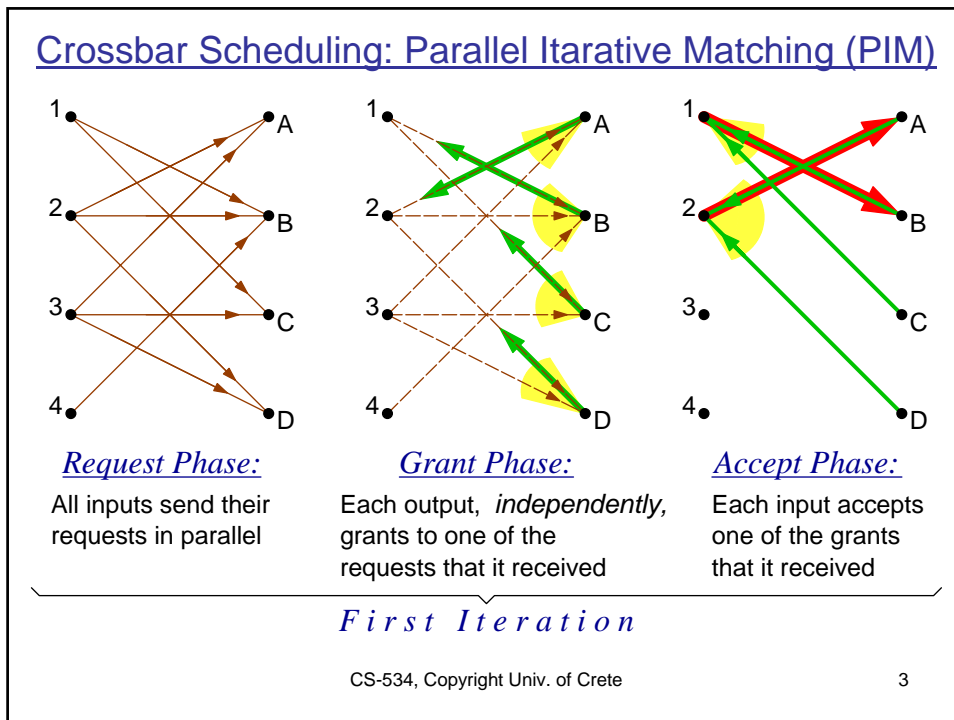   - interdependent decisions – difficult problem!

CS-534, Copyright Univ. of Crete   1

CS-534, Copyright Univ. of Crete   2

Crossbar Scheduling: Parallel Itarative 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

F i r s t   I t e r a t i o n

CS-534, Copyright Univ. of Crete   3

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

S e c o n d   I t e r a t i o n

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

CS-534, Copyright Univ. of Crete   4

Maximal Matching

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

Maximum Matching

Maximum possible number
of connections
for the given request pattern

CS-534, Copyright Univ. of Crete   5

Maximum Matching is complex ... and may be Unfair:

- NxN switch ?
- M requests } :

- $O(N \cdot (N+M))$ deterministic algorithm

  Tarjan: Data Structures & Network Algorithms
  SIAM, 1983

- $O(N+M)$ randomized algorithm

  Karp e.a.: ACM STOC, 1990

VOQ's

never served!

① ②

CS-534, Copyright Univ. of Crete   6

Performance of PIM:

- each iteration resolves, on average, $\geq \frac{3}{4}$ of the remaining unresolved requests
  $\Downarrow$
  $O(\log N)$ iterations

- 16×16 switch:

#iterations	1	2	3	4
% matched	64-87%	88-100%	97-100%	99.9-100%

- delay (VOQ w. PIM) $\approx$ 2× delay (Output Queueing)

- how to implement random selection ???

Fairness Issue:

1/4 grant probability per request

5/16
5/16
5/16
1/16
5/16
5/16
5/16

1/4 accept probability per grant

(weighted random selection ???)

---

iSlip: Most Popular, Practical Crossbar Scheduler

- Practical variation of PIM
- Widely used in commercial switch products
- Nick McKeown: "The iSLIP Scheduling Algorithm for Input-Queued Switches", IEEE/ACM Tr. on Networking, April 1999
- Performance properties:
  - performs well under uniform and heavy load, when most VOQ's are non-empty and matching almost "rotate" among inputs & outputs
  - adds delay under medium loads, until most VOQ's become non-empty
  - does not perform very well under *"unbalanced"* traffic (each input preferentially sends to one or a few "favored" output(s) of itself)

iSLIP → variation of PIM ⟨ → fewer iterations
                                → better fairness

Nick McKeown: IEEE/ACM ToN April 1999
• was used in CISCO GSR-12000, Tiny Tera, Abrizio/PMC-Sierra, e.a.

(a) REQUEST: like PIM

(b) GRANT: • PIM: each output randomly grants to a requesting input
            • iSLIP: grant in a Round-Robin fashion
                     to a requesting input;
              top priority = next of : ~~previous grant~~ NO.!!-see below
                              previous grant that was accepted!

(c) ACCEPT: • PIM: each input randomly accepts one of the granting outputs
            • iSLIP: accept in Round-Robin priority a granting output,
                     with top priority = next of previous accept

CS-534, Copyright Univ. of Crete   9

• if we were granting round-robin after the previous grant,
  without regard as to whether the grant was accepted or not,
  then the grant pointers may get synchronized in a "bad way"
  and stay that way for a long time, resulting in very poor perf.

• perform well even with
  a single iteration !
  (versus 2 to 4 iterations of PIM)
  reason: under heavy load, with
  ~all inputs requesting ~all outputs,
  pointers get and stay de-synchronized
  and scheduler degenerates into
  time-division multiplexing
  ⇒ can reach even 100% throughput under uniform load

Example:

in1 • ⟶ • out1
in2 • ⟶ • out2
requests (persistent)
grant t=1
grant t=2

• more iterations improve the delay
• has good fairness properties
• is relatively easy to implement
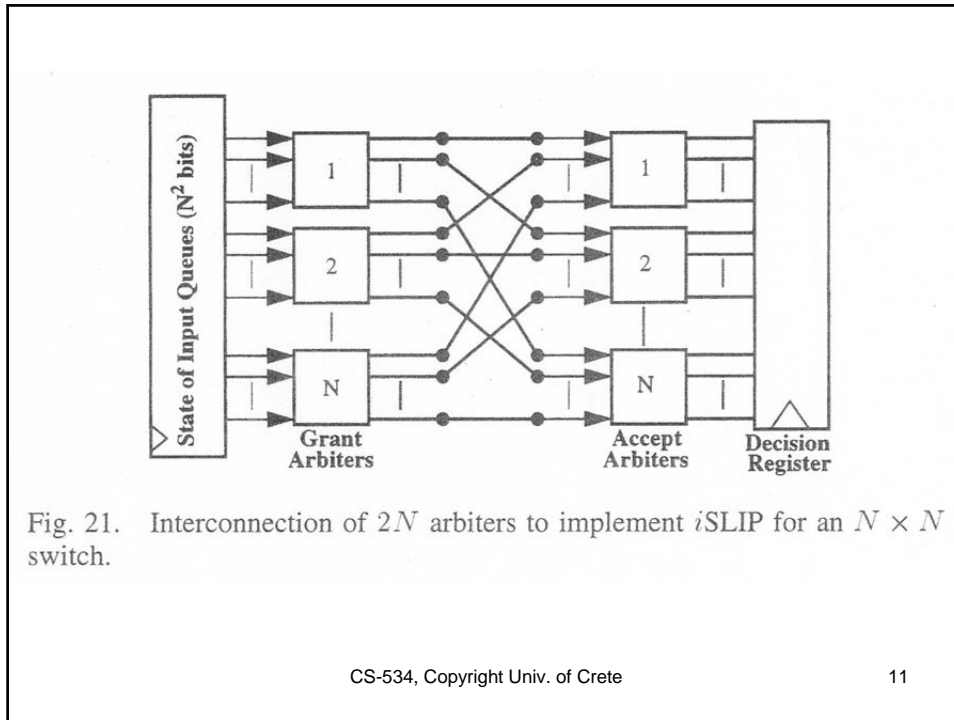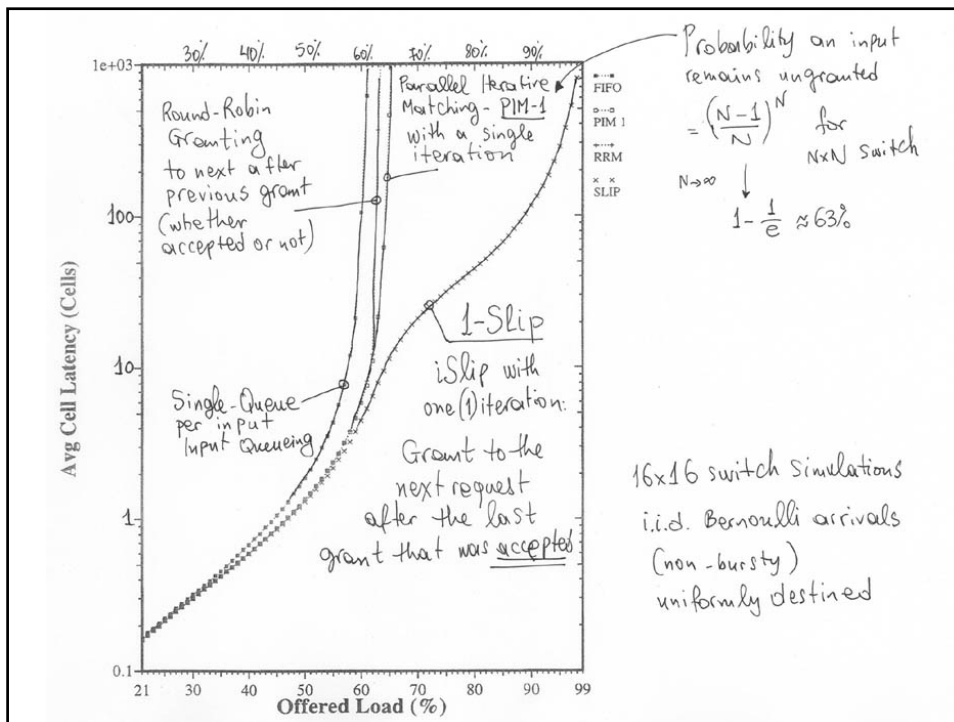
CS-534, Copyright Univ. of Crete   10

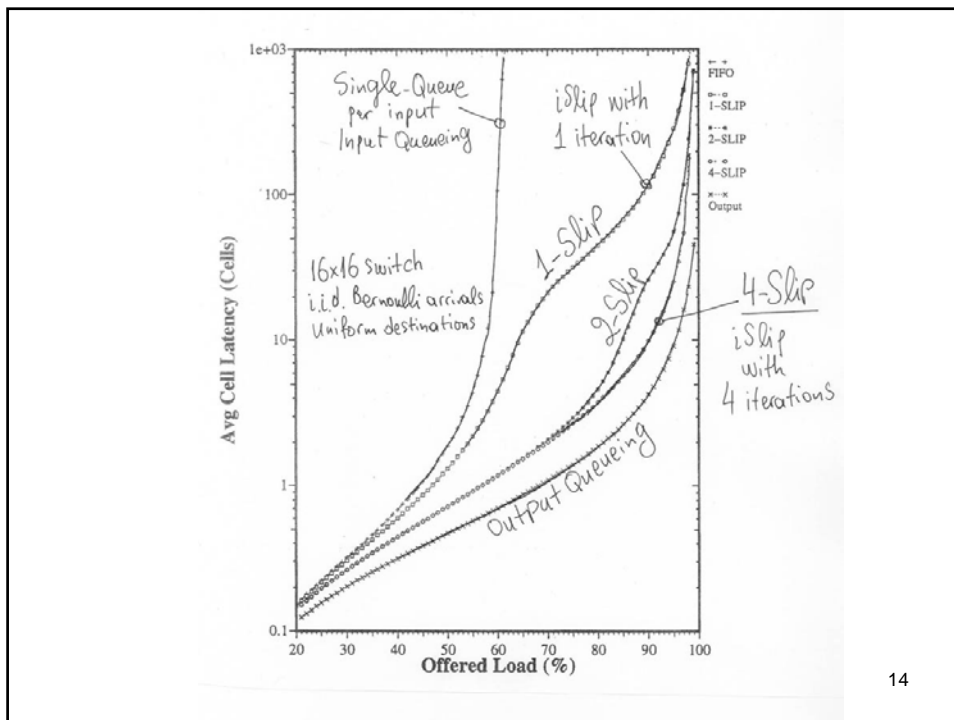4.3 VOQ, Crossbar Scheduling   5
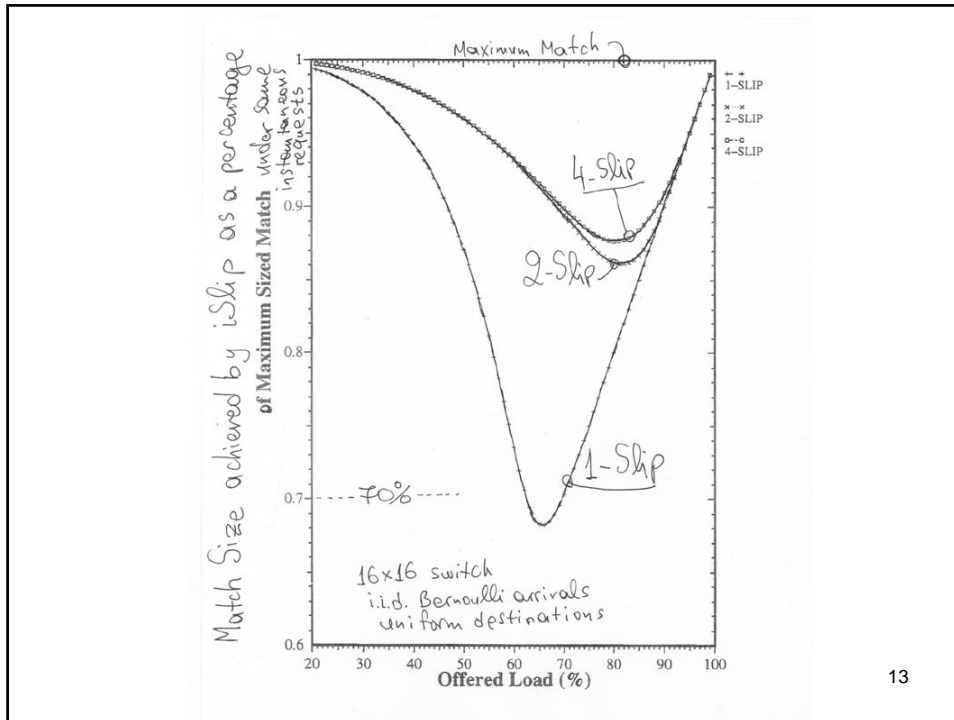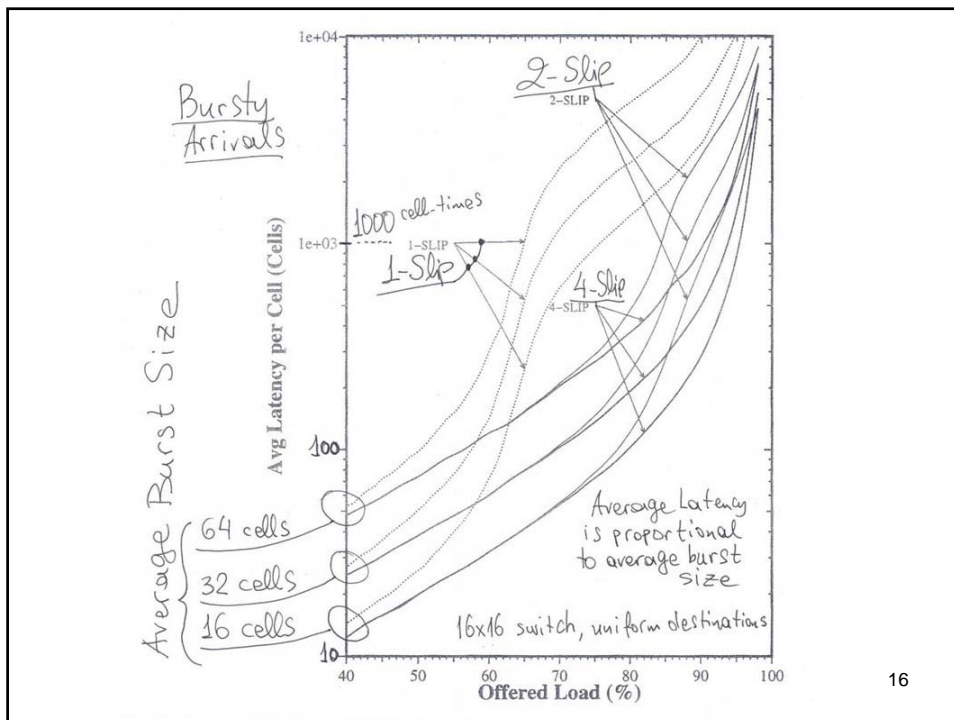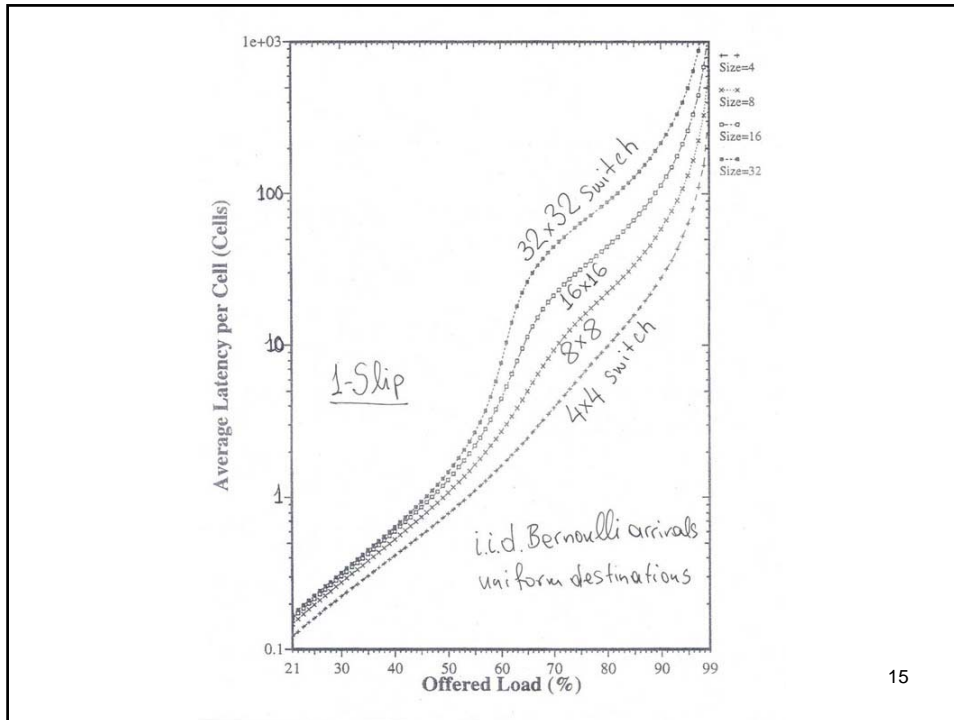
Fig. 21. Interconnection of $2N$ arbiters to implement iSLIP for an $N \times N$ switch.

CS-534, Copyright Univ. of Crete   11

13

14

4.3 VOQ, Crossbar Scheduling   8

Pipelined Scheduling for Cell-Based Crossbars

Q_{ij} = Occupancy (# of cells) of Queue i→j minus # of pending requests issued on behalf of that queue
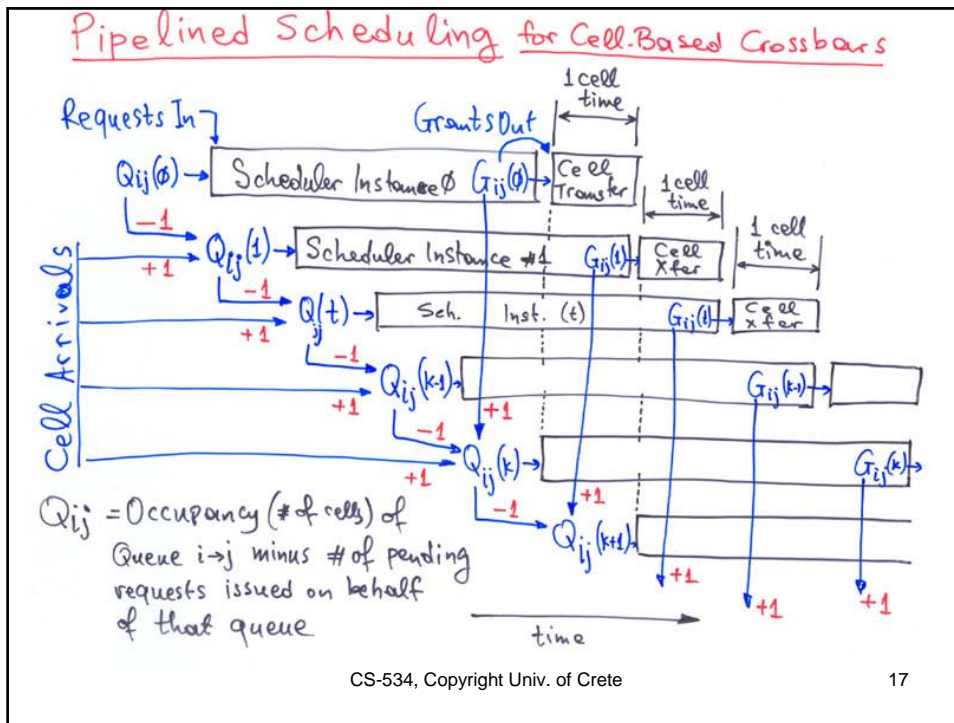
CS-534, Copyright Univ. of Crete   17

- Cell Arrivals increment Q_{ij}
- When $Q_{ij} \geq 1 \Rightarrow$ a Request R_{ij} is issued
- Requests issued decrement Q_{ij} (pending decision on cell transfer)
- When Pending Decision is resolved:
  - Successful Grants leave Q_{ij} as is (already decremented)
  - Failed Grants re-increment Q_{ij} (restore unaccepted request)

Assumption for this scheme to work: <u>fixed-size cells</u>
⇒ requests for cells and actual cell transfers are <u>interchangeable</u> with each other: if the "first" request, on behalf of the "first" cell in the queue is not granted, then the "second" request (issued on behalf of the "second" cell) will result in transfering the "first" cell, if granted.

Ref: Oki, Rojas-Cessa, Chao: "A Pipeline-Based Approach for Maximal-Sized Matching Scheduling in Input-Buffered Switches" IEEE Communications Letters, June 2001 — http://acts.poly.edu/~chao/publications.html

CS-534, Copyright Univ. of Crete   18