Asynchronous Operation of Bufferless Crossbars

Georgios Passas and Manolis Katevenis

Foundation for Research and Technology (FORTH),
Inst. Of Computer Science (ICS), Greece
&
Computer Science Dept., Univ. of Crete, Greece

May 31, 2007

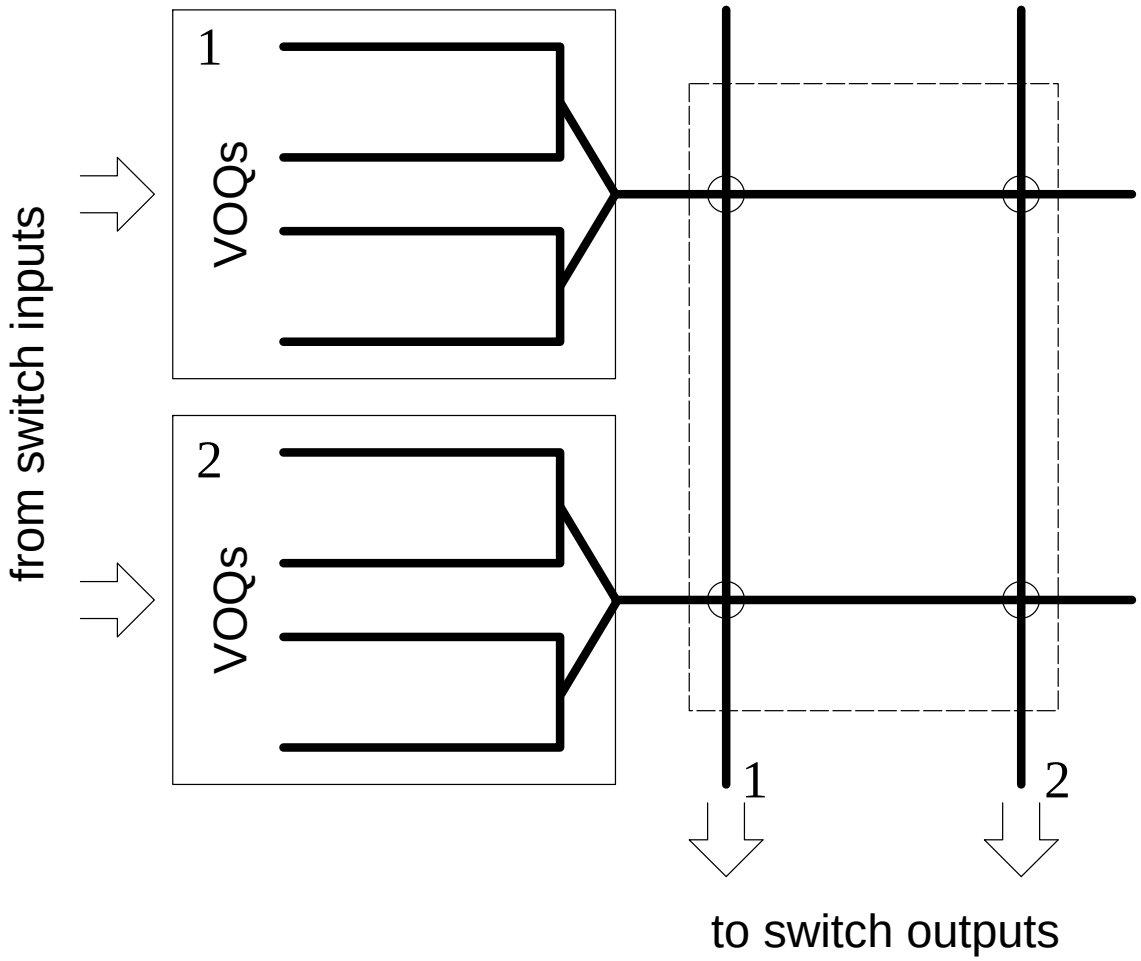
Topic

Switching of variable-size packets in crossbar switches

Outline

- Background
- Problem statement and Contribution
- Proposed operation and Results
- Summary

Background: switch architecture 1/2



Example of a 2×2 VOQ switch with a bufferless crossbar

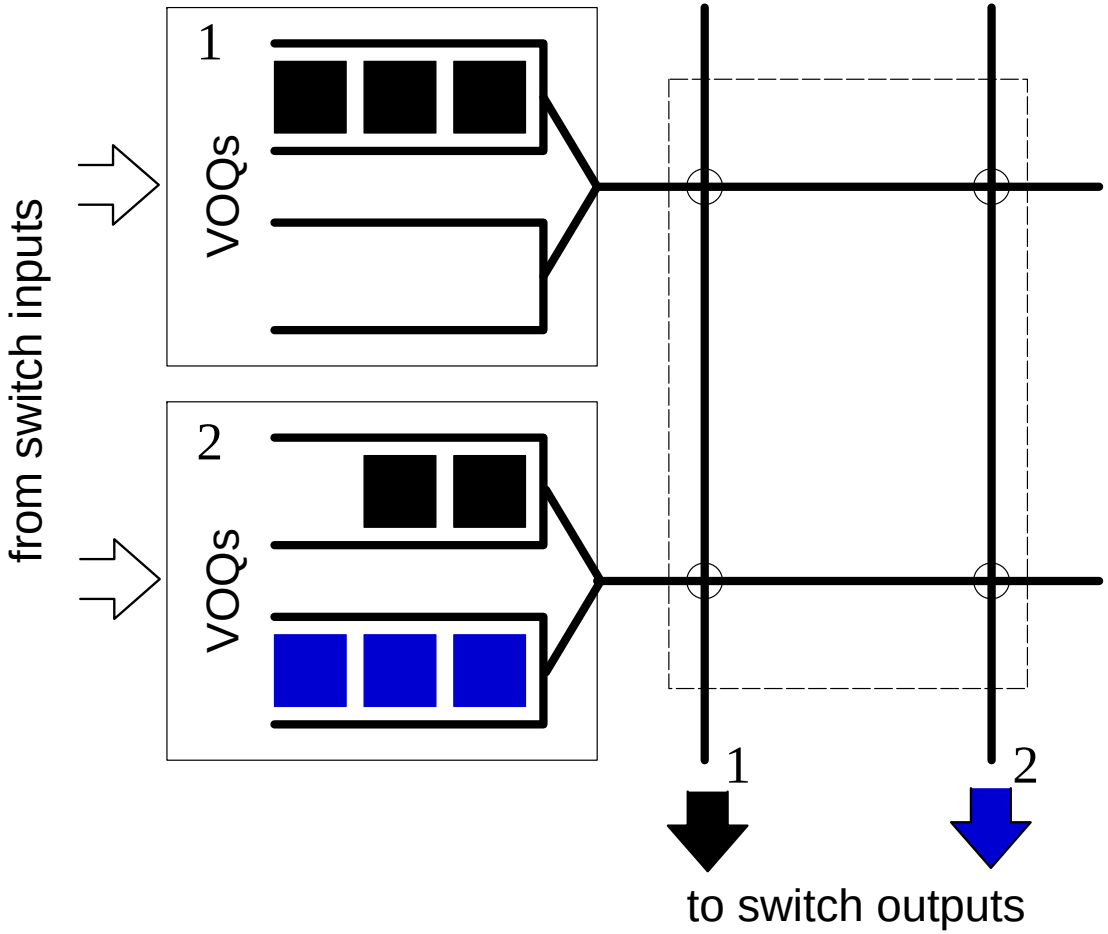
Background: switch architecture 2/2

- VOQ eliminates HOL blocking
  - HOL blocking may limit switch throughput to 59% of link capacity

- Scheduling is needed to configure the crossbar
  - each input can be connected to at most one output and vice versa
  - bipartite graph matching problem

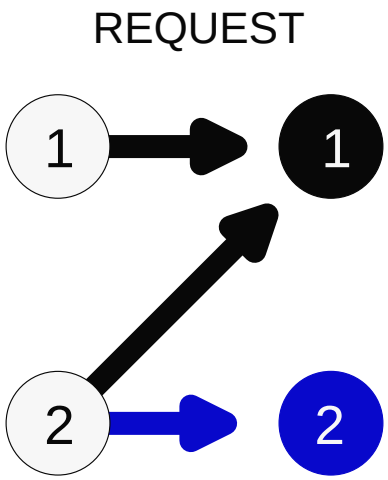
- Crossbar schedulers usually run a 3-phase matching algorithm

Background: switch scheduling 1/2



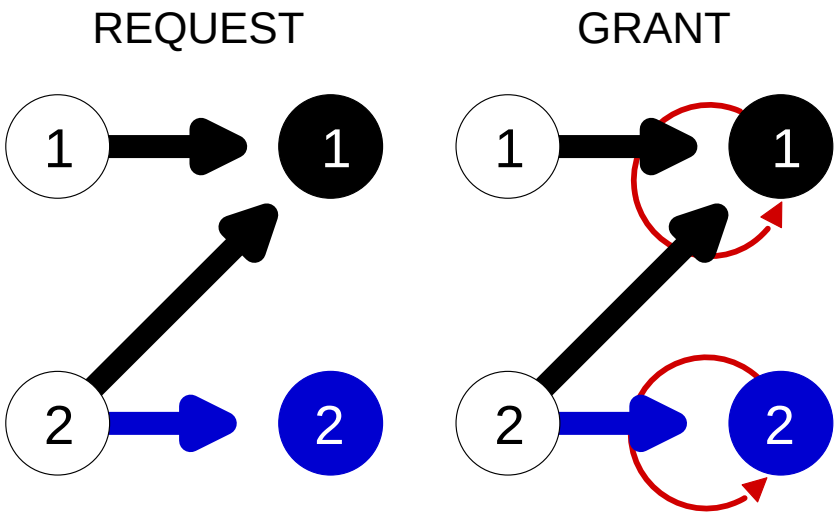
Traffic scenario

Background: switch scheduling 2/2



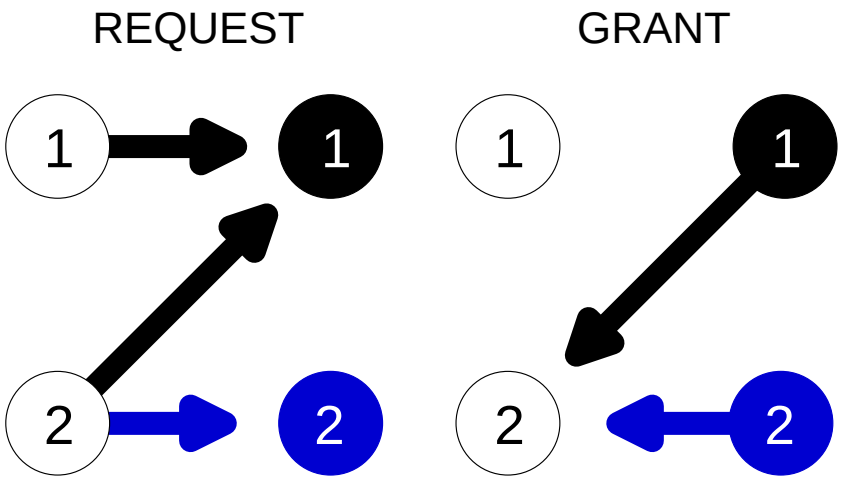
- Each input requests the outputs for which it has traffic
- Inputs submit requests in synchrony

Background: switch scheduling

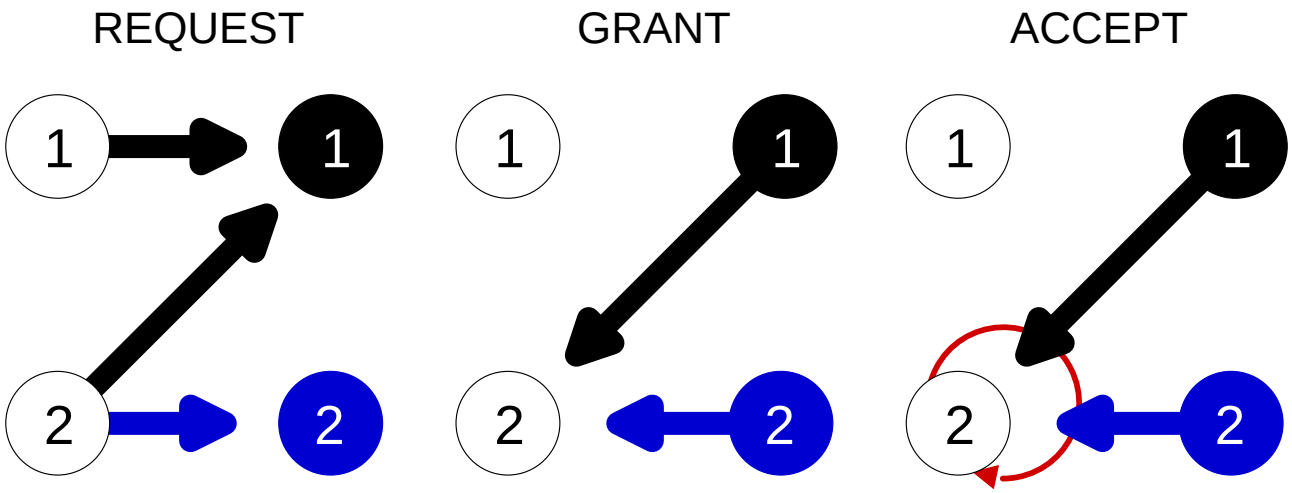


- Each output selects among requests the one to grant
- Independent but synchronized output decisions

Background: switch scheduling

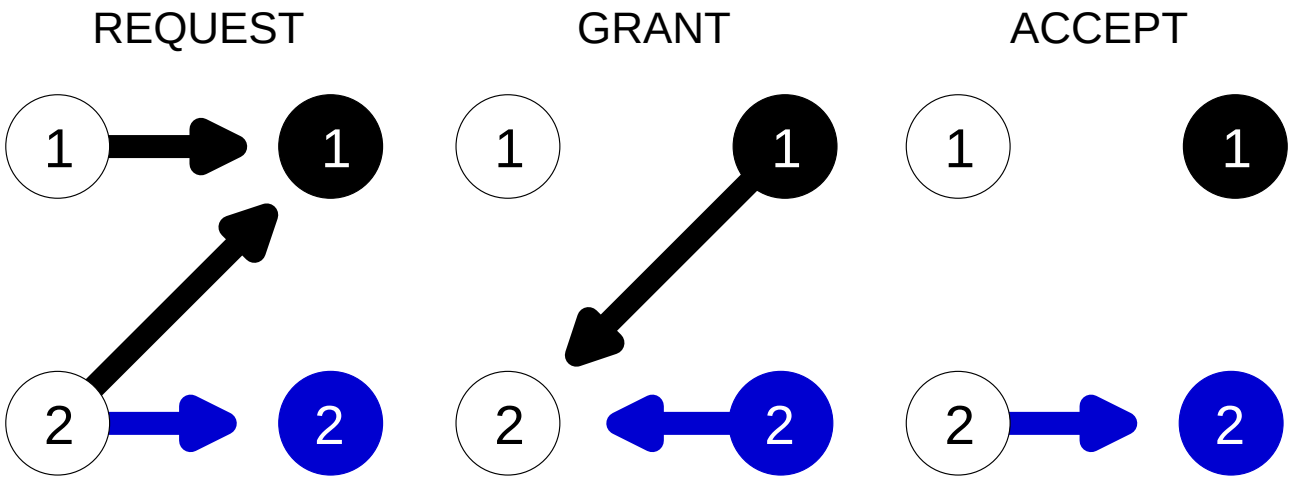


Background: switch scheduling 2/2

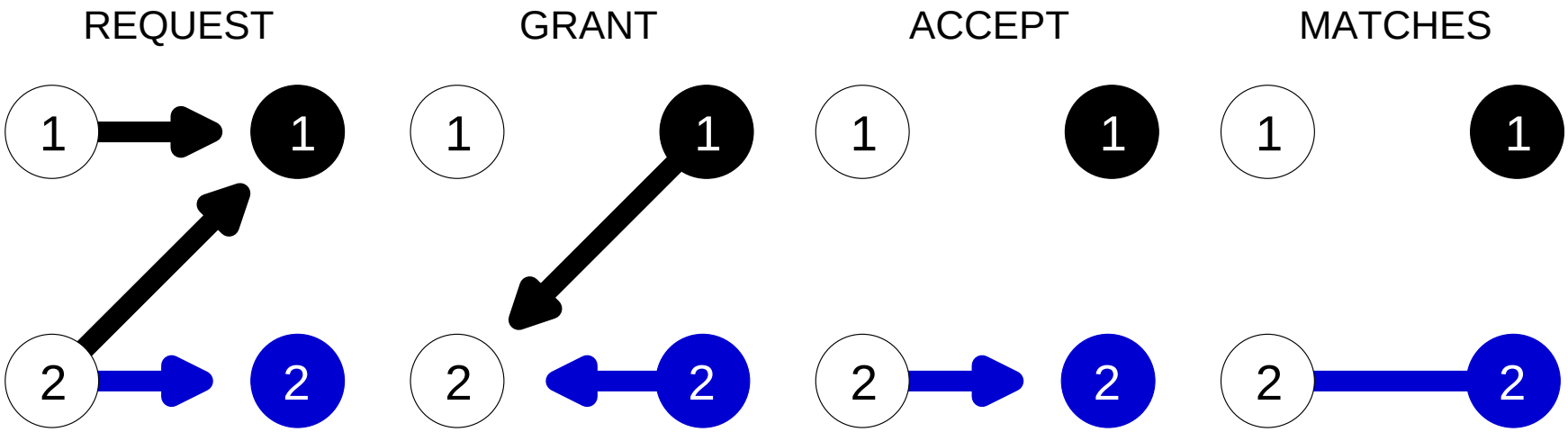


- Each input selects among grants the one to accept
- Independent but synchronized input decisions

Background: switch scheduling

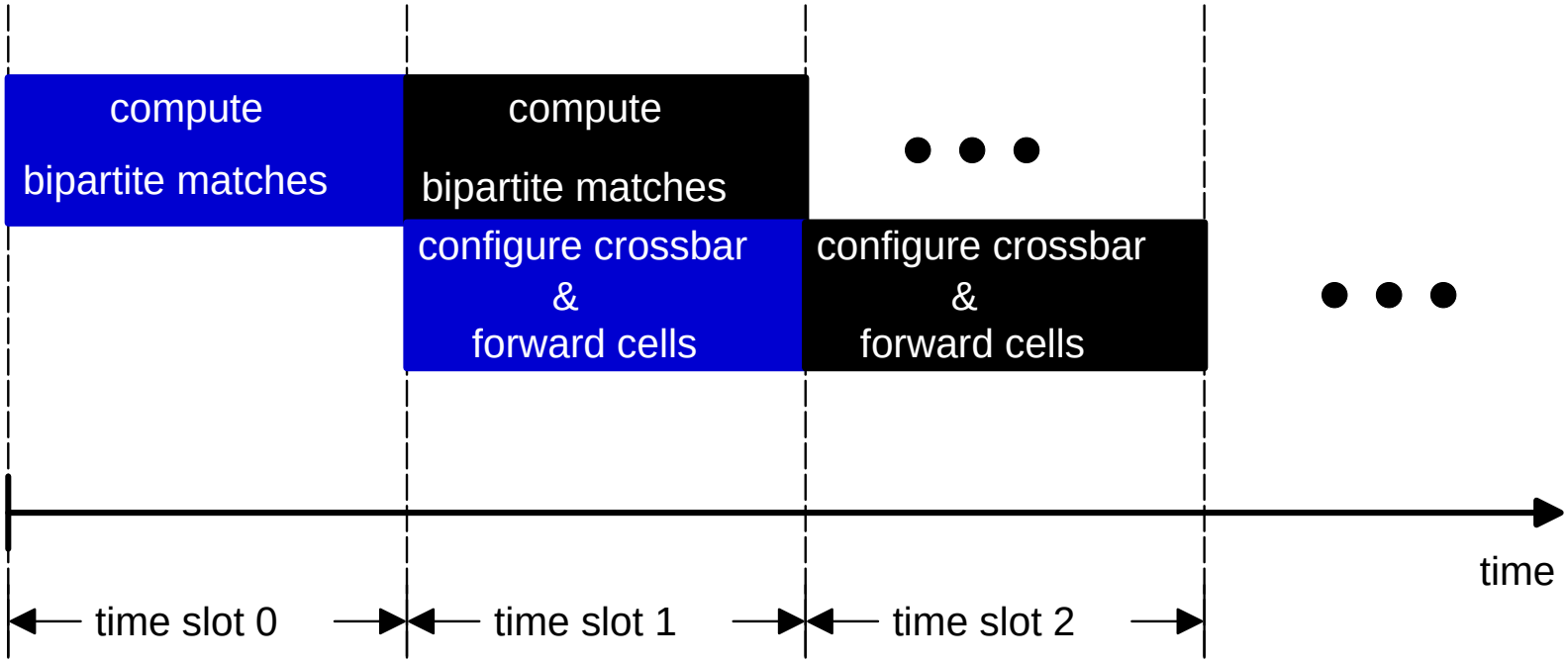


Background: switch scheduling 2/2



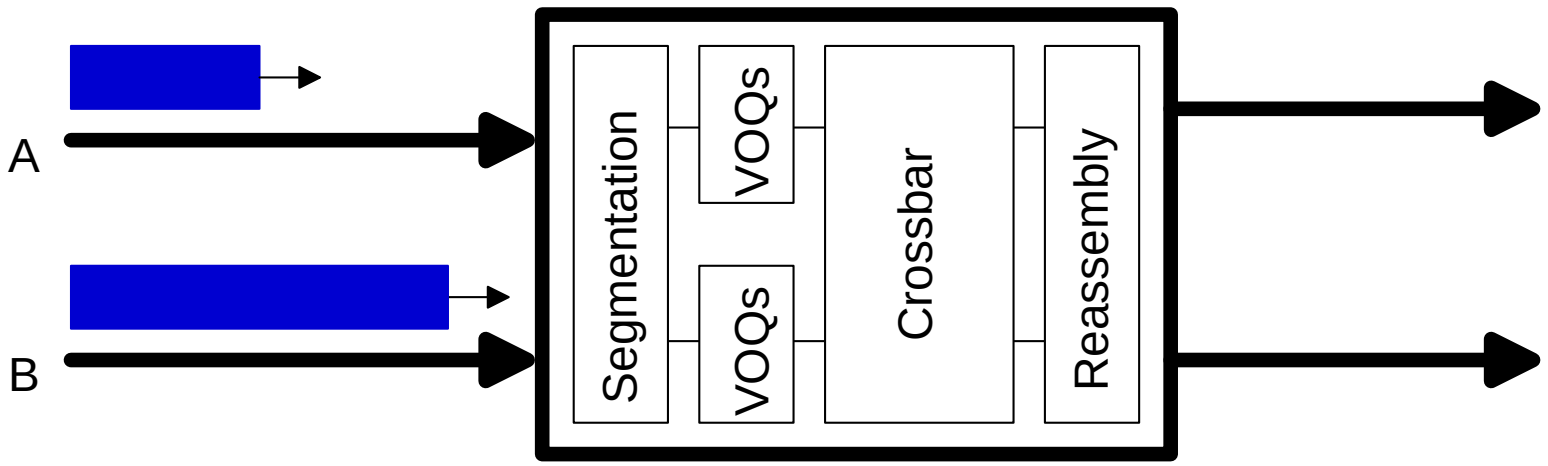
- Bipartite matches emerge after the accept phase is completed
- The crossbar is configured accordingly

Background: synchronous xbar reconfiguration



- Crossbar configuration changes synchronously
  - Configuration of time slot n+1 specified by matches computed in time slot n
- Operation on fixed-size internal cells
  - 1 time slot = 1 cell time

Background: variable-size packets 1/2



- External packets have variable size (e.g. IP packets)
- Segmentation to cells, switching, reassembly, transmission

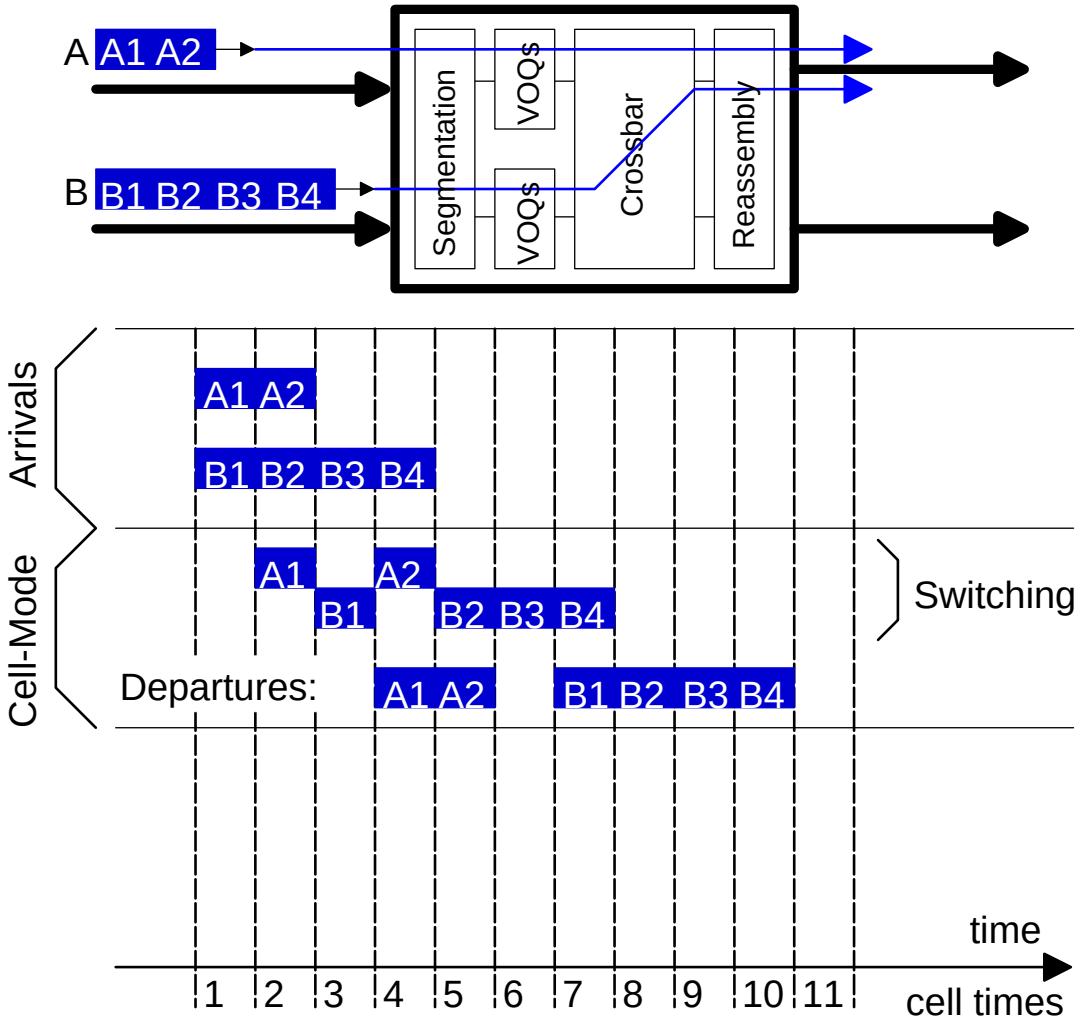
Background: variable-size packets 2/2

Two alternatives to switch the internal cells

- *cell mode*:
  ignore which cell belongs to which packet and
  reconfigure the crossbar from scratch every cell time

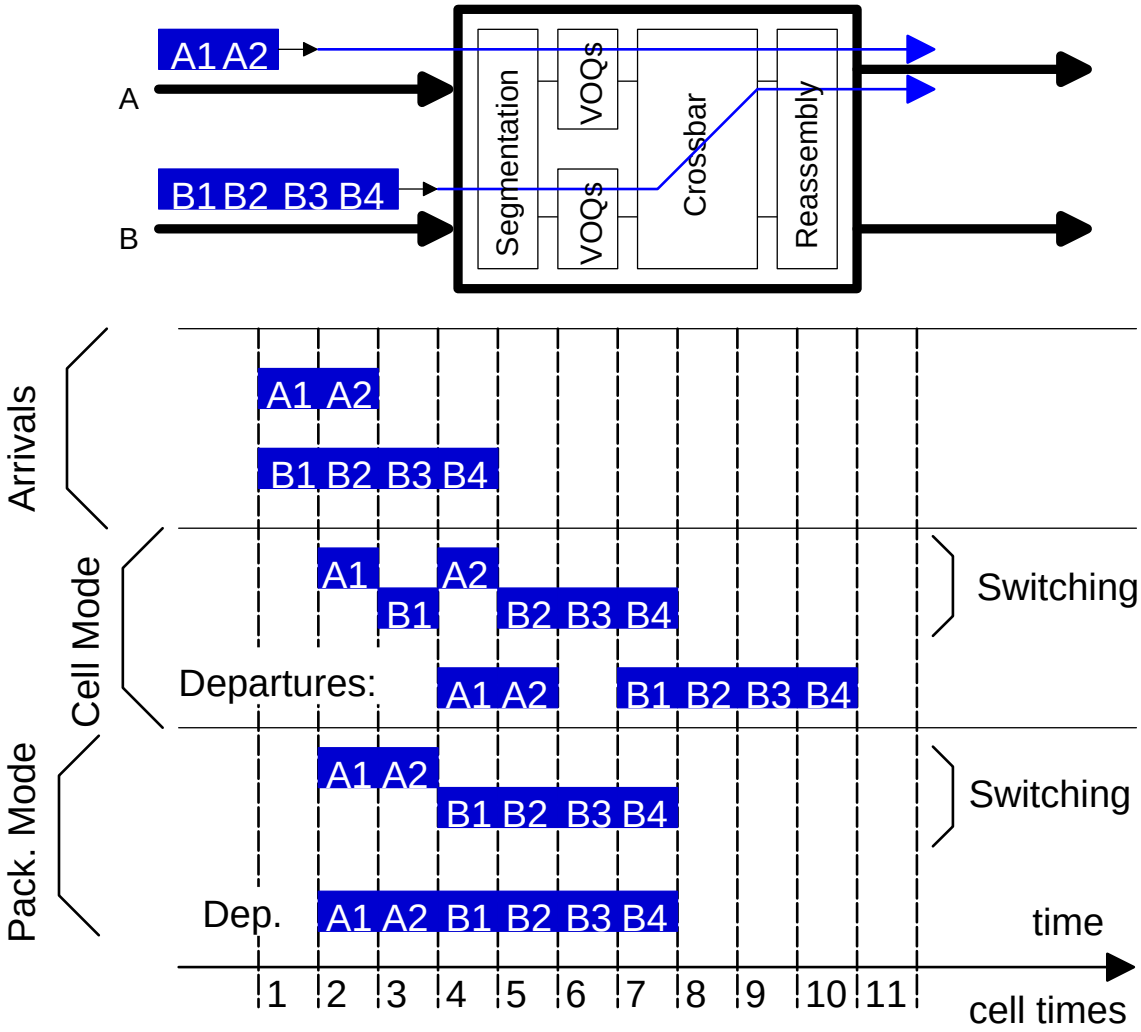
- *packet mode*:
  keep cells of the same packet consecutive by
  tearing down connections only when whole packets have been fwded

Background: cell-mode operation



Reassembly complexity is needed and store&fwd is enforced

Background: packet-mode operation



Packet reassembly is trivial and cut-through is allowed

Problem statement

- Synchronous operation presumes cell size for packet-size granularity
  - 64 Bytes is a typical cell size in research and production switches

- External packets are not always integral multiples of internal cells
  → the last cell of a packet is padded
  → internal speed-up is required for padding overheads

- Internal speed-up for padding overheads may be ~2 in the worst case
  - e.g. with 65-Byte packets in a 64-Byte cell switch

Contribution

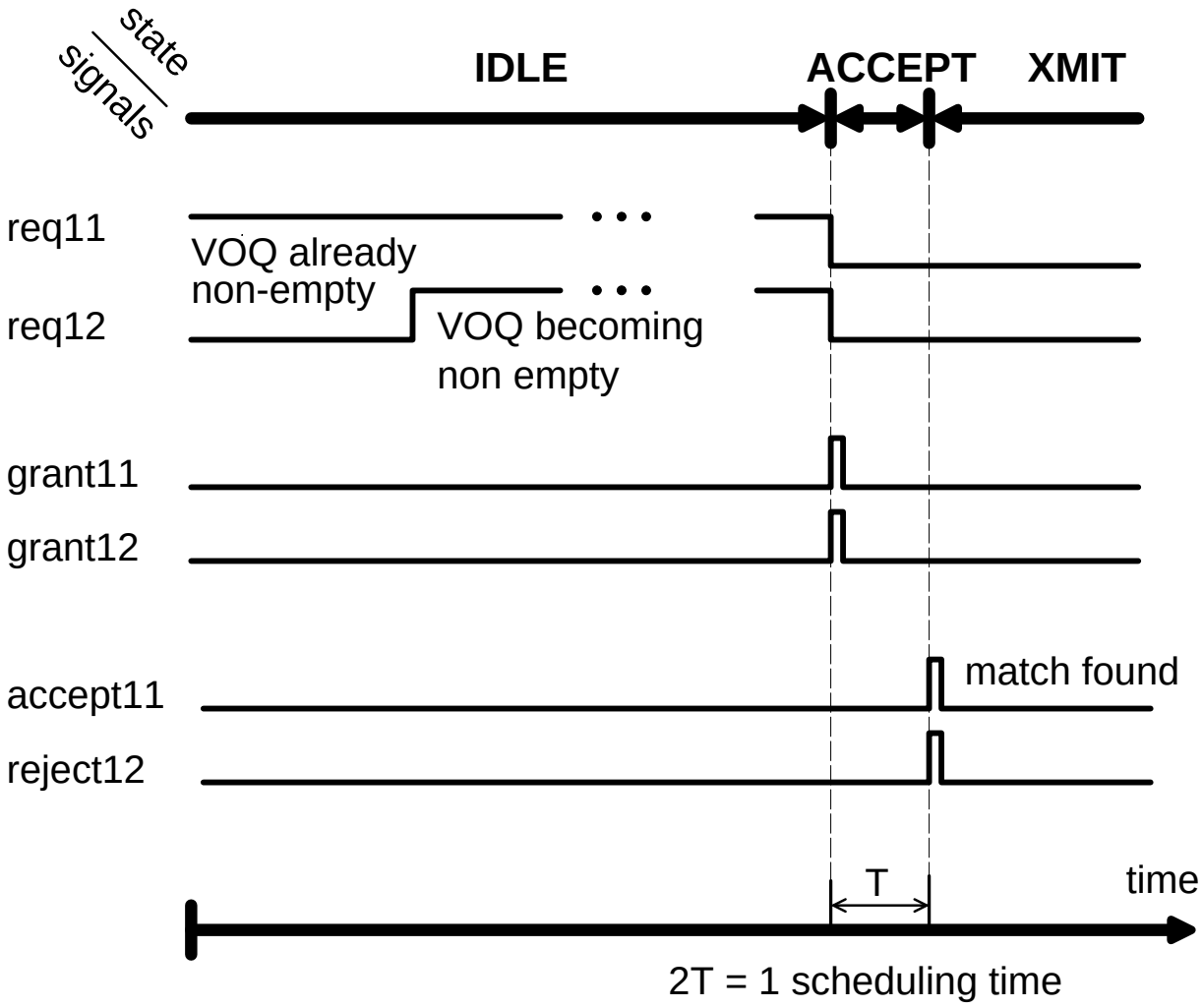
Bufferless crossbars operating directly on variable-size packets

- No segmentation & no reassembly
  → advantages of packet mode operation are inherited

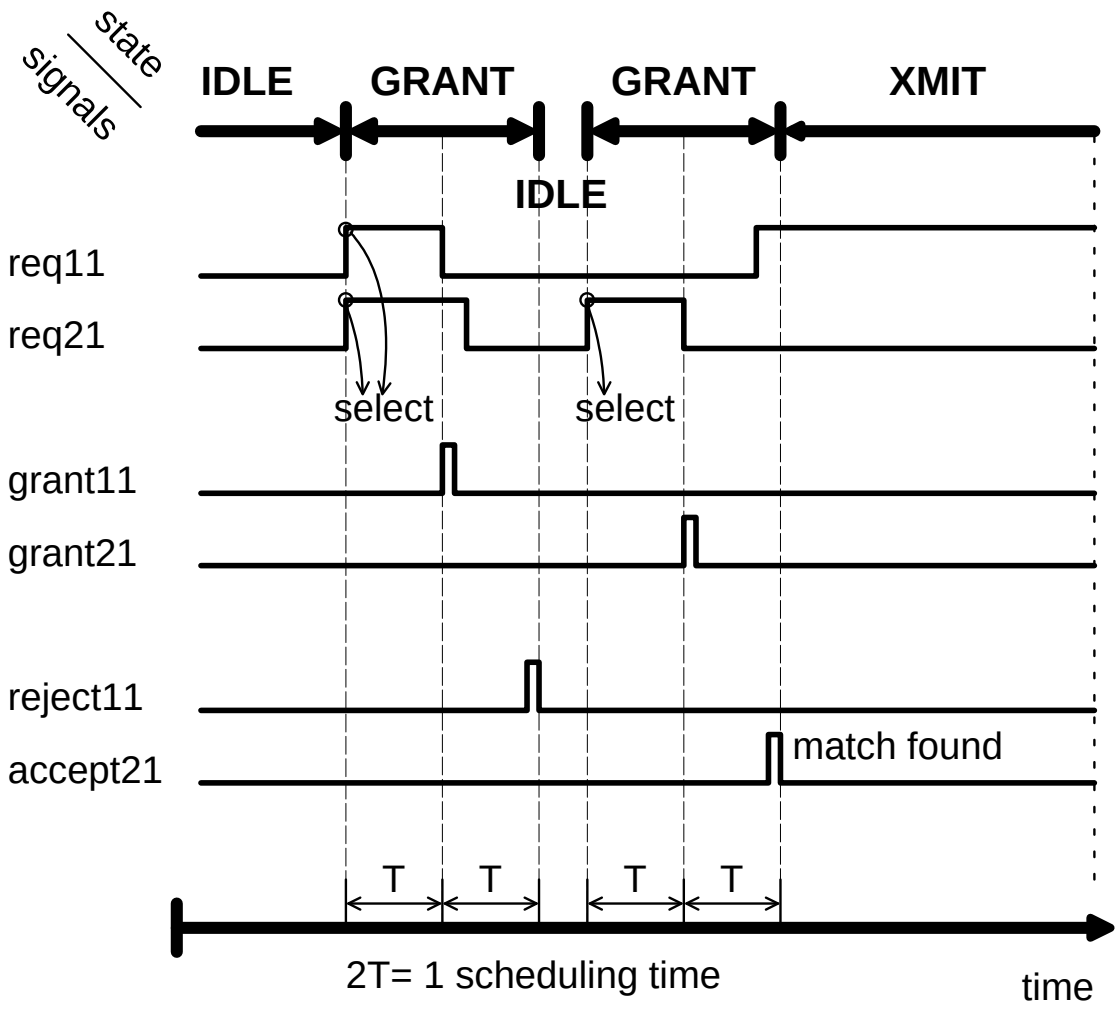
- No segmentation to fixed-size cells
  → no padding overheads, no internal speed-up for them

Proposed operation 1/5

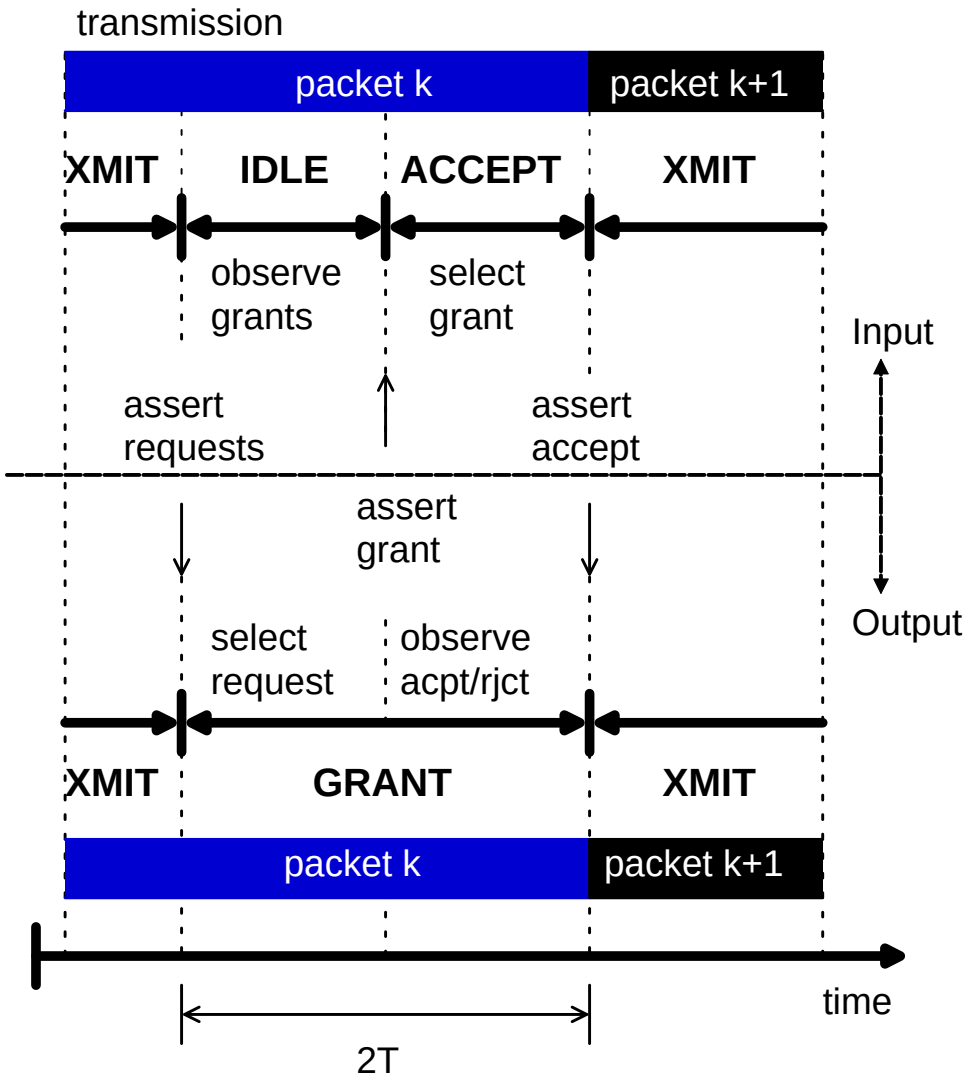
- Bipartite matches computed in three phases
- Inputs and outputs do not synchronize their scheduling decisions
- Crossbar configuration may change at any moment in time
  - infinitely small packet-size granularity



Example of input state transitions

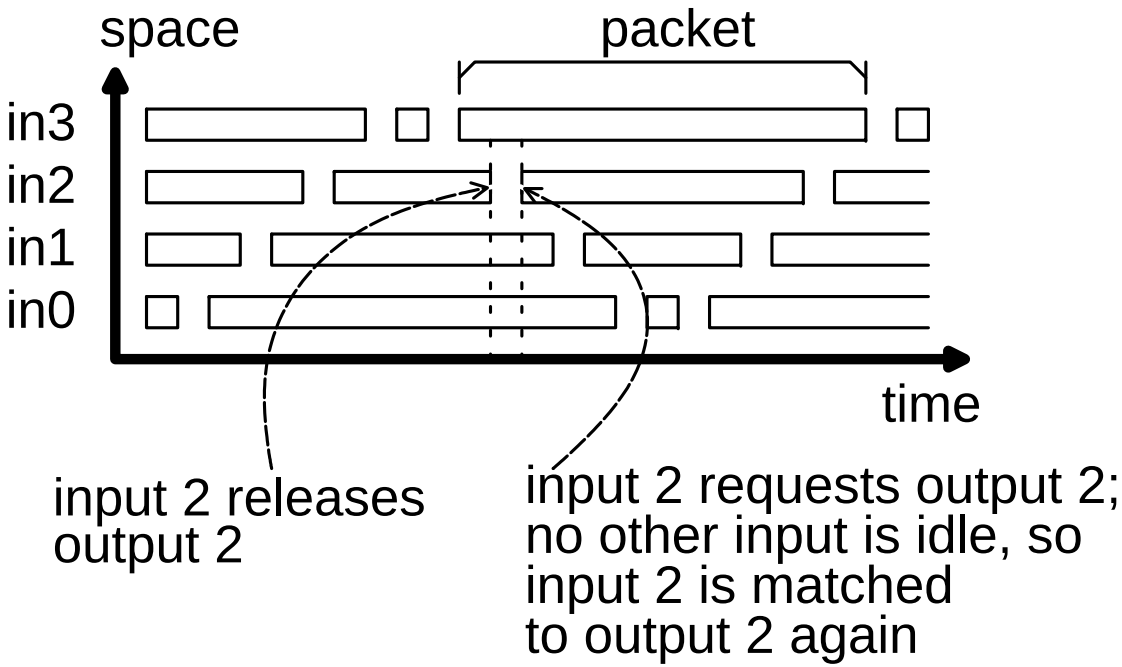


Example of output state transitions



Example of joint input-output state transitions

Proposed operation



- Scheduler does not consider candidacies for inputs or outputs which are in the middle of a packet forwarding
  → some flows may be locked-out
- Packet-mode scheduling similarly suffers from starvation

Simulations

- Event-driven simulation for asynchronous operation
- Compared against packet-mode operation
- Packet-size granularity
  - asynchronous: 40 time smaller than the scheduling time
  - synchronous: cell-size
- Synthetic traffic
  - constant, uniform, trimodal packet-size distributions
  - uniform, unbalanced (Zipf) destination distributions
- Result: *Asynchronous operation does not degrade performance compared to packet-mode operation*

Summary

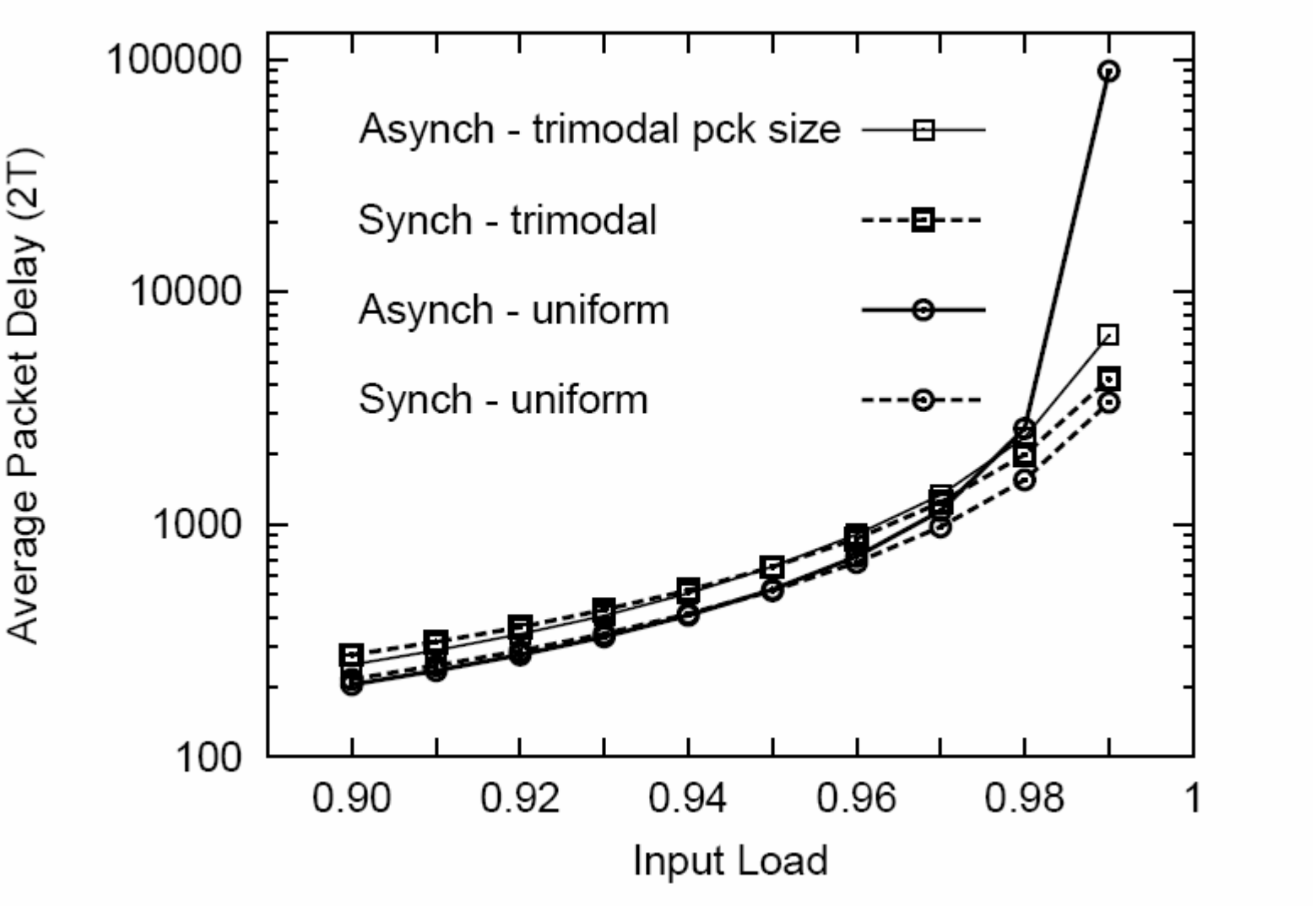
- We proposed asynchronous bufferless crossbar operation for direct switching of variable-size packets

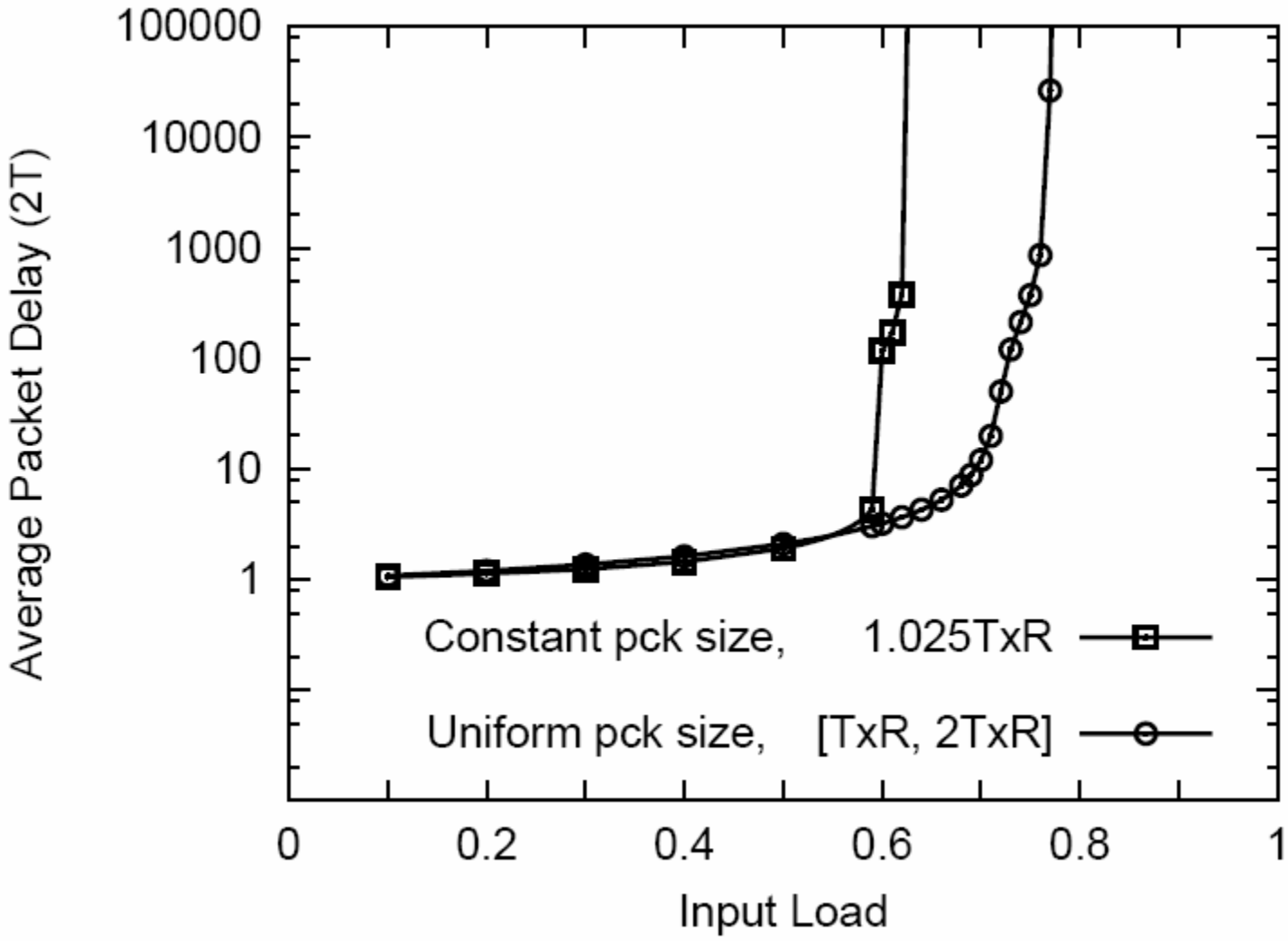
- Like packet-mode operation:
  no packet reassembly is needed and cut-through is allowed
  Unlike packet-mode operation:
  no internal speed up for cell padding overheads is needed

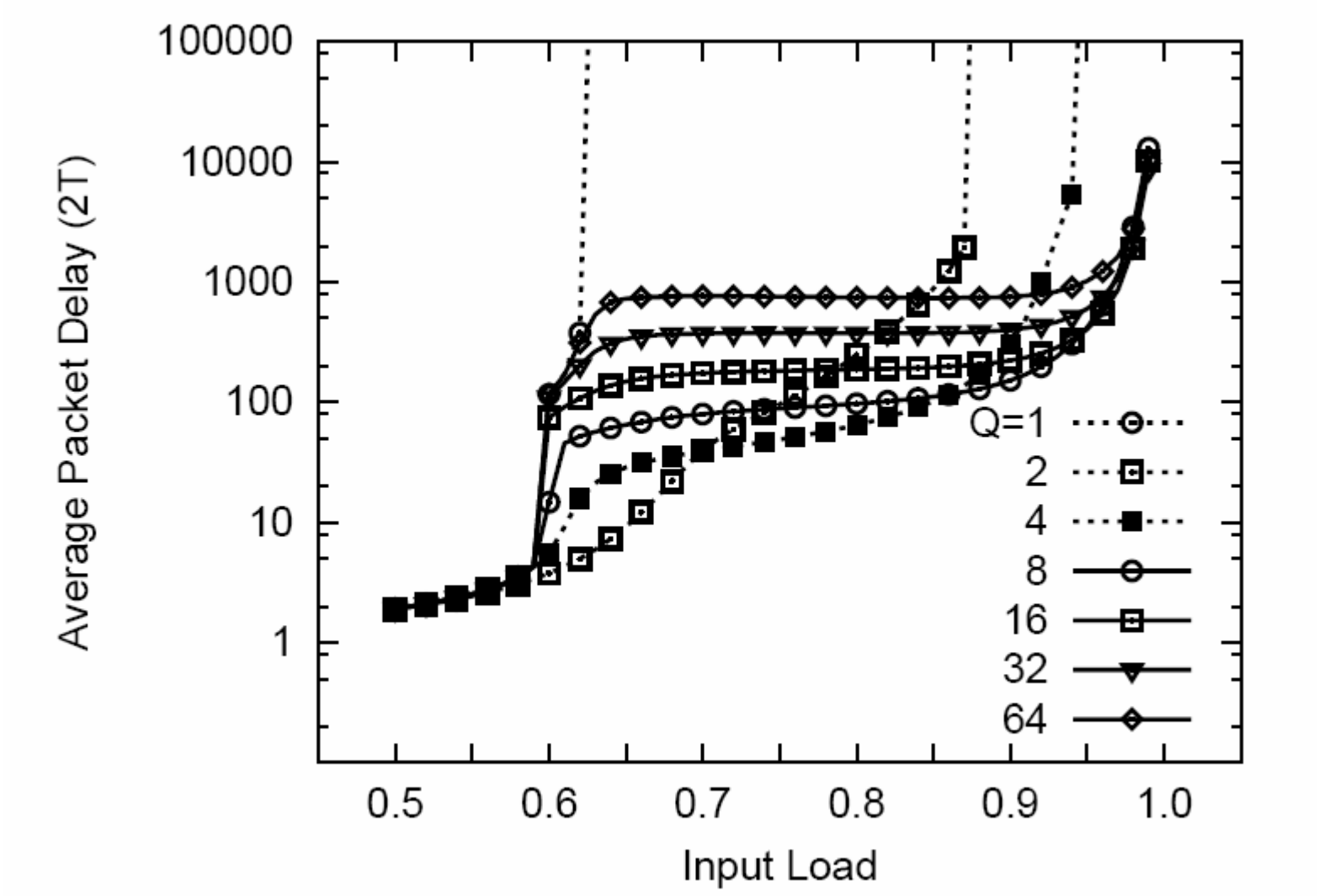
- Simulations showed almost no performance degradation compared to packet-mode scheduling

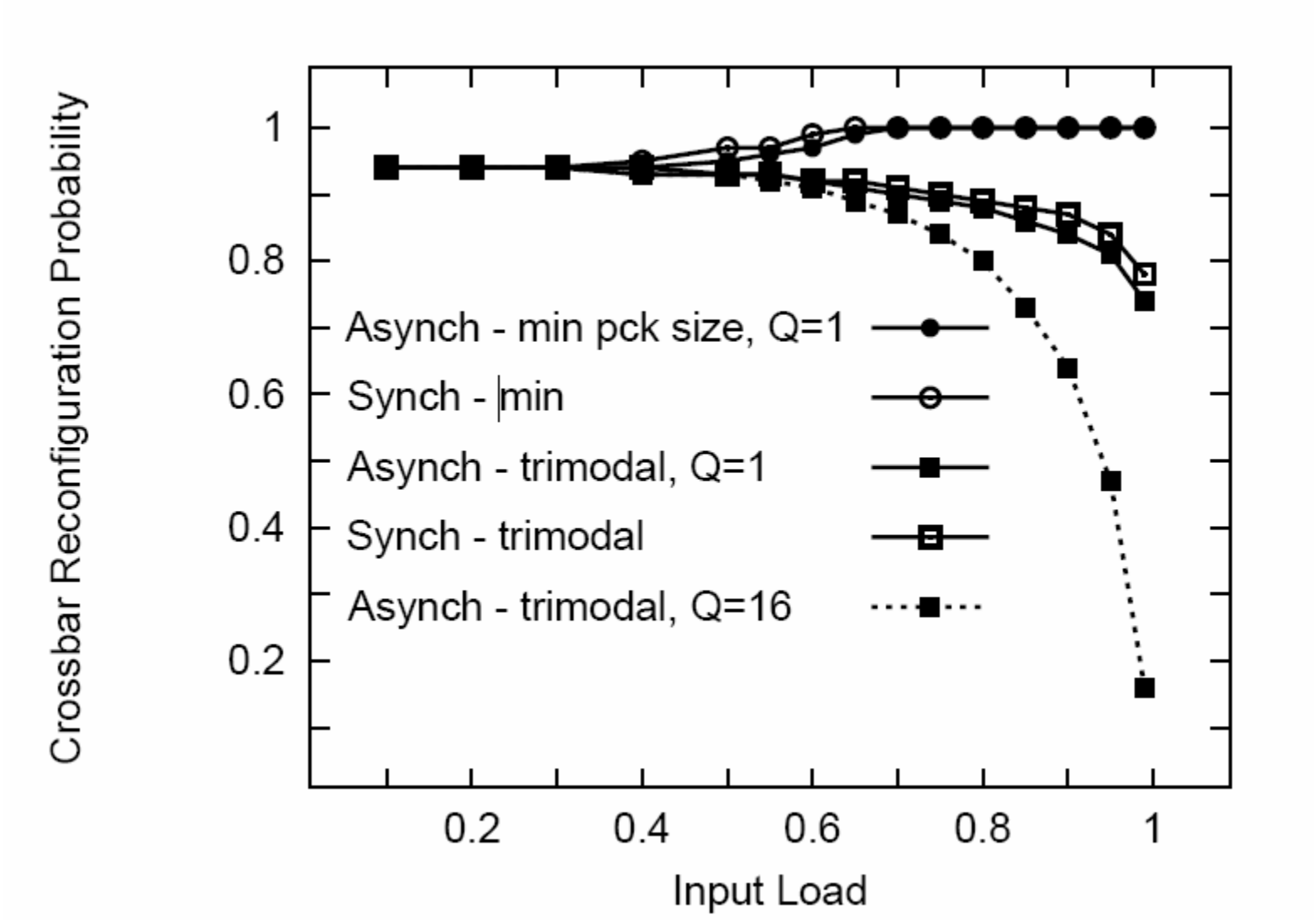
THANK YOU

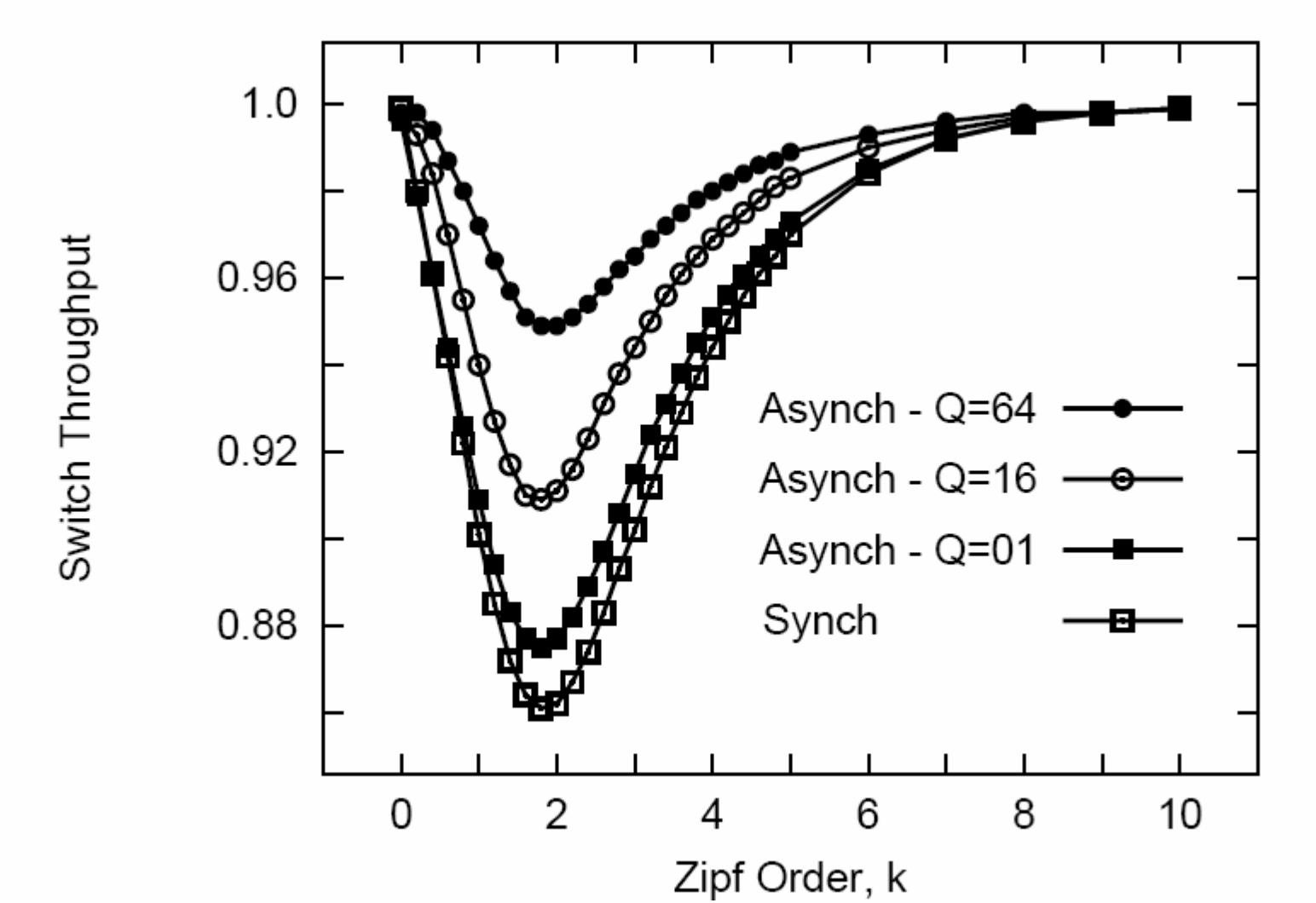
BACKUP SLIDES


Average Packet Delay (2T)
100000
10000
1000
100
0.90
0.92
0.94
0.96
0.98
1
Input Load
Asynch - trimodal pck size
Synch - trimodal
Asynch - uniform
Synch - uniform


Average Packet Delay (2T)
100000
10000
1000
100
10
1
0
0.2
0.4
0.6
0.8
1
Input Load
Constant pck size, 1.025TxR
Uniform pck size, [TxR, 2TxR]


100000
10000
1000
100
10
1
Average Packet Delay (2T)
0.5
0.6
0.7
0.8
0.9
1.0
Input Load
Q=1
2
4
8
16
32
64


Crossbar Reconfiguration Probability
1
0.8
0.6
0.4
0.2
0.2
0.4
0.6
0.8
1
Input Load
Asynch - min pck size, Q=1
Synch - min
Asynch - trimodal, Q=1
Synch - trimodal
Asynch - trimodal, Q=16


Switch Throughput
1.0
0.96
0.92
0.88
0
2
4
6
8
10
Zipf Order, k
Asynch - Q=64
Asynch - Q=16
Asynch - Q=01
Synch