Revisiting CIOQ Switching: From Exact OQ Emulation to Single-iteration Approximation

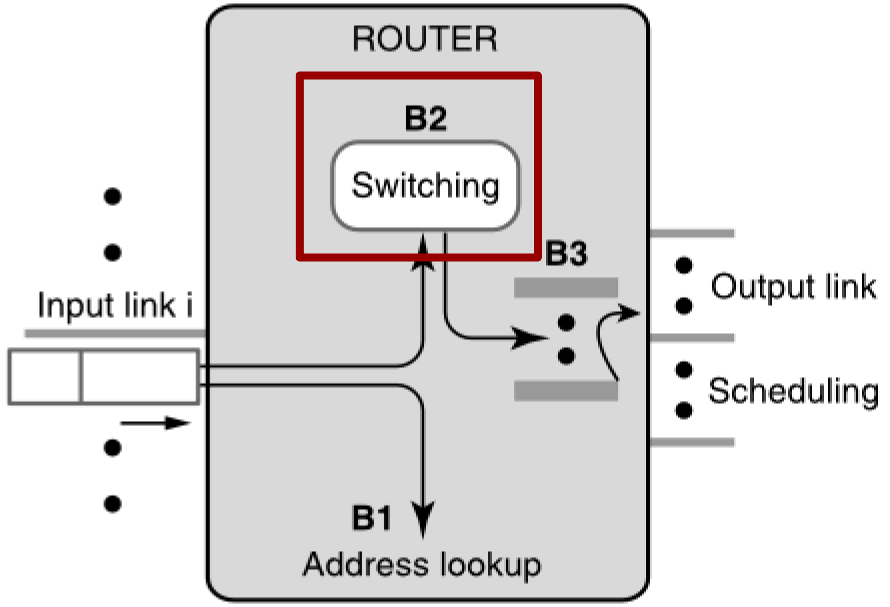
Georgia Tech

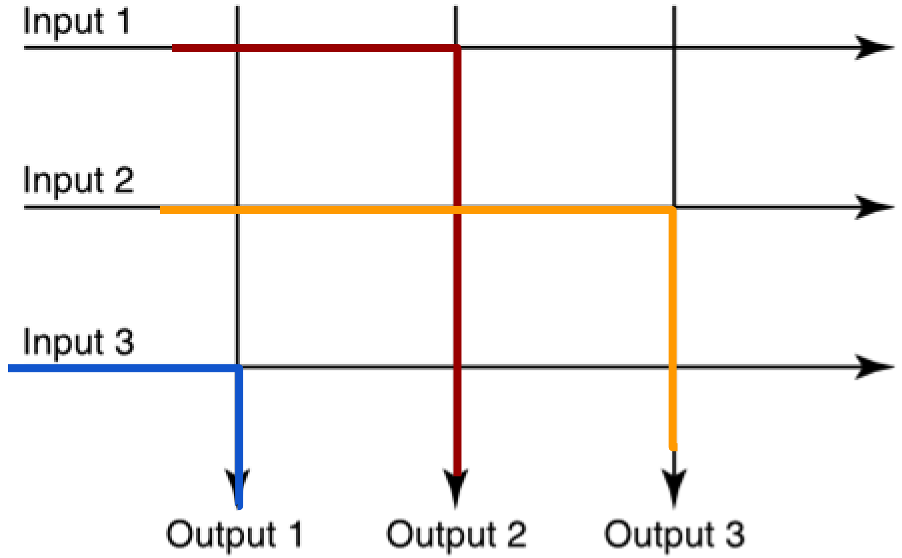
Outline

- ❑ *What* is CIOQ switching?
- ❑ *Why* CIOQ switch architecture matters?
- ❑ McKeown’s OQ-emulation theorem (1999)
- ❑ Single-Iteration CIOQ switching: SW-EDF (2025)

Recap: Switching

- **Router and Switches**
  - A general **interconnection** device
  - Switch packets from **input** to **output** links
  - Upon arrival of a packet, the output link it will be switched to is already determined by lookup
- **Switching fabric: Crossbar**
  - Find disjoint input-output pairs





Recap: Switching

- **Comparison of IQ, OQ, and CIOQ Switching**

Input-Queued (IQ)

Pros:

- Simple hardware
- Designed for improving throughput

Cons:

- Uncontrolled delay

Output-Queued (OQ)

Pros:

- 100% throughput
- Minimal delay

Cons:

- Requires N times speedup at crossbar and output buffer

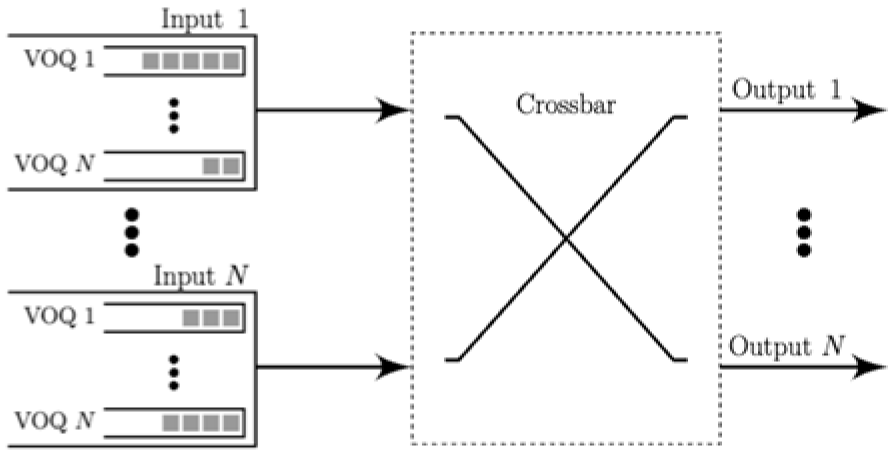


Fig. 1. Input-queued crossbar switch.

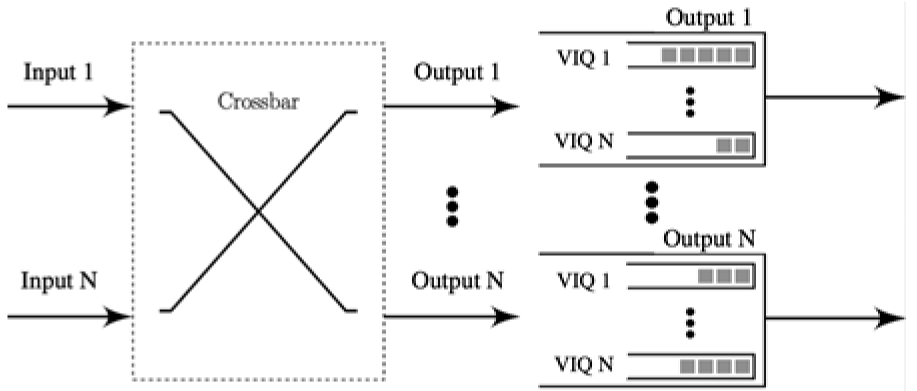


Fig. 2. Output-queued crossbar switch.

Recap: Switching

- Comparison of IQ, OQ, a CIOQ S

Provide Quality of Service (QoS)

Input-Queued (IQ)

Pros:
- Simple hardware
- Designed for improving throughput

Cons:
- Uncontrolled delay

utput-Queued (OQ)

Pros:
- 100% throughput
- Minimal delay

Cons:

The Problem researchers aim to solve ❗

mbined I t- & Output-Que l (CIOQ)

Pros:
- 100% throughput
- Lower delay
- Scalable: requires 2x speedup

Cons:
- Need complicated scheduling

Background on IQ switching

- **Challenge**: High quality matching for high-radix high-line-rate IQ switches
  - High throughputs (>90%) and low packet delays (in nanoseconds scale)
- A well-known IQ switching algo: iSLIP [McKeown, N.(1999)]
  - runs O(logN) iterations to compute each matching
  - each iteration includes 3 phase: request, grant, accept
  - >80% throughput
- **How good can single iteration do in previous work?**
  - None of existing single-iteration algos except the SB-QPS and SW-QPS can attain >60% throughputs

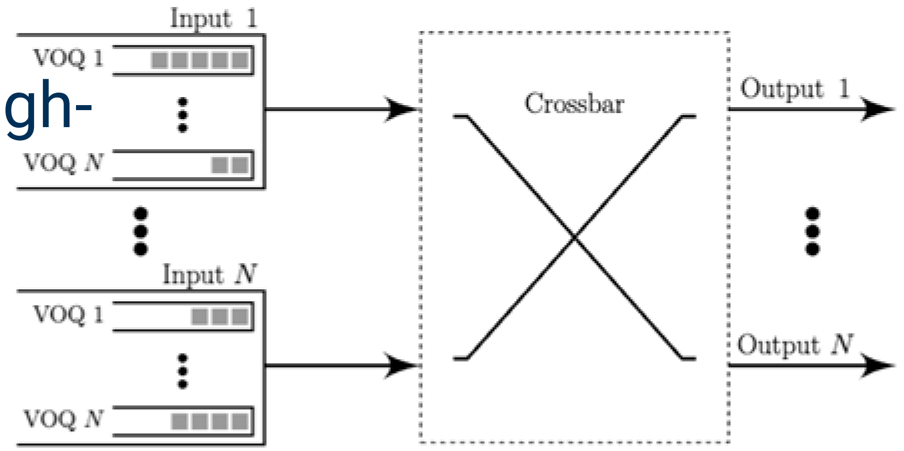
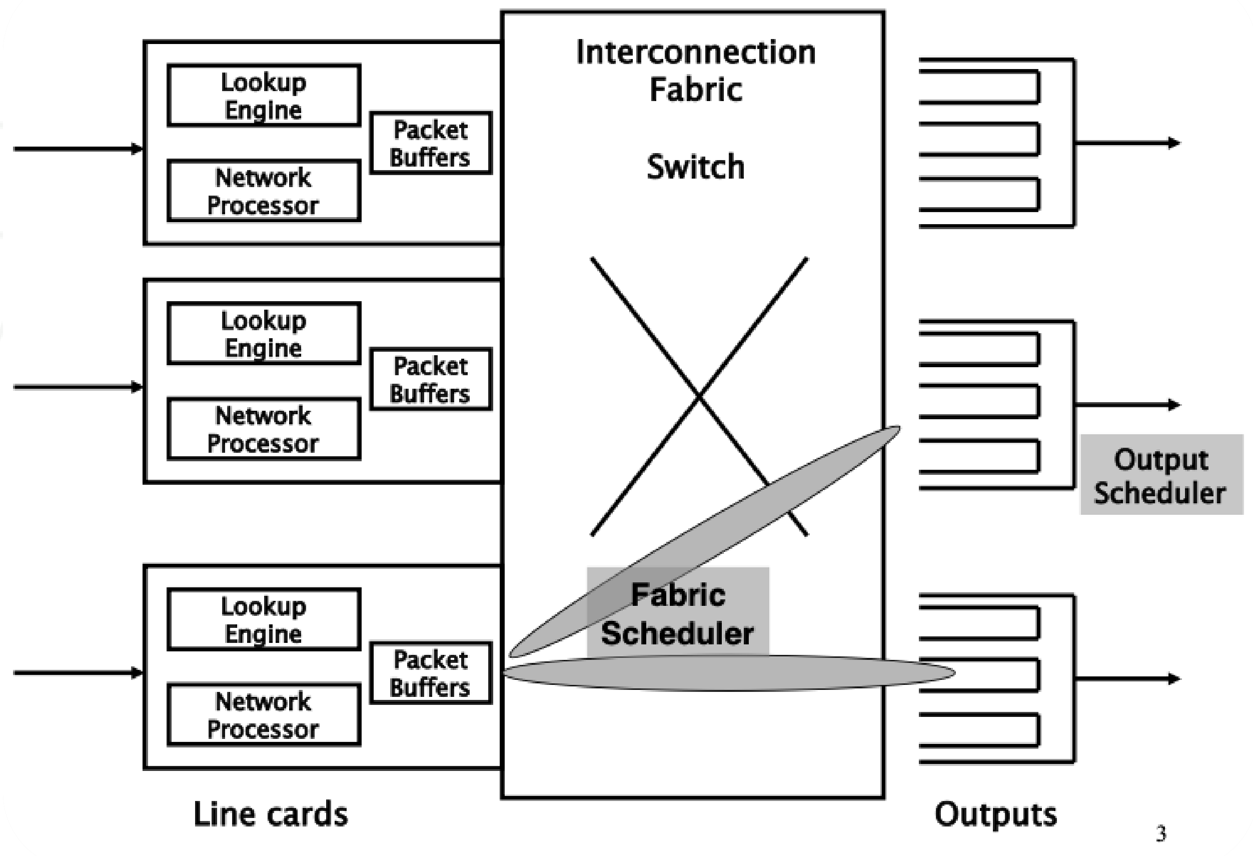


Fig. 1. Input-queued crossbar switch.

Belongs to Parallel Iterative Switching Algorithms (PISA)

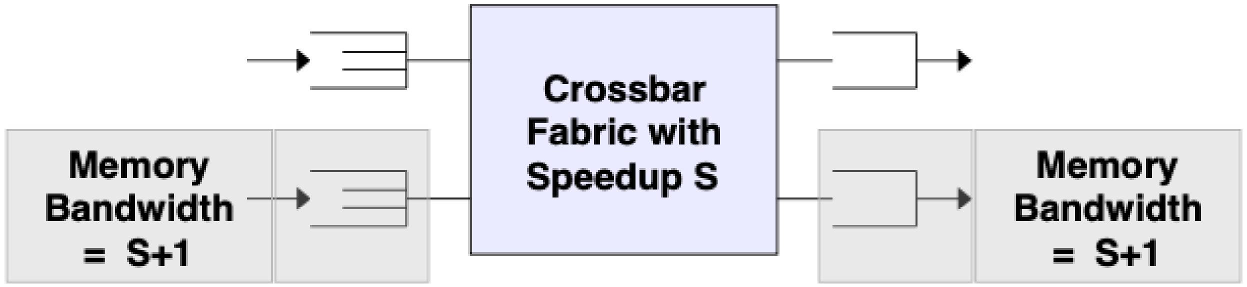
Background on CIOQ switching



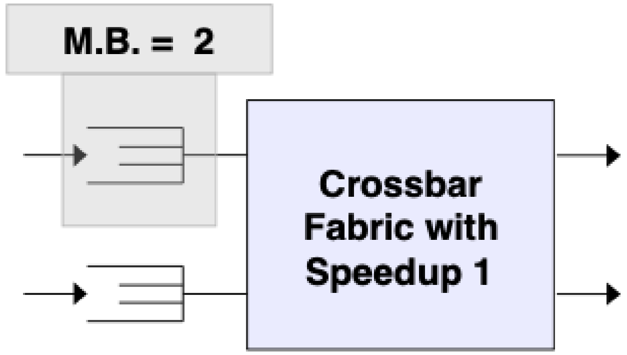
- Two interlocking schedulers
  - Fabric (crossbar) scheduler and the output scheduler
  - Speedup is the "grease" that smoothes their interaction or coordination
- Intuition for interlocking
  - When the output queue is long, the fabric scheduler’s load is lighter — it doesn’t need to rush new packets there.
  - When the output queue is short, the fabric scheduler works harder to fill it quickly.

B. Prabhakar, “Scheduling Algorithms for CIOQ Switches,” Telecom CenterWorkshop, 1997

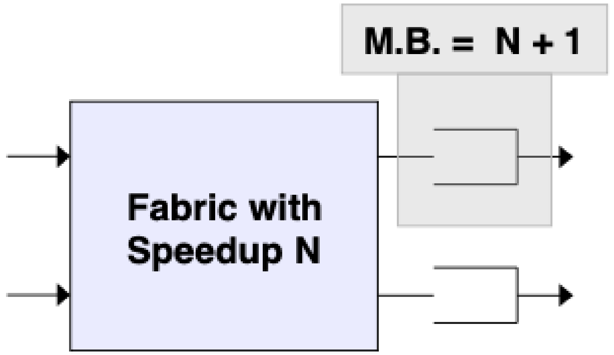
What is Speedup?



- Switch with speedup S: In each time slot
  - At most 1 packet arrives at (departs from) each input (output)
  - At most S packets switched from (to) each input (output)



Input-queued switch



Output-queued switch

B. Prabhakar, “Scheduling Algorithms for CIOQ Switches,” Telecom CenterWorkshop, 1997

Background on CIOQ switching

OQ is impractical in terms of hardware implementation

- **Output-Queued Switch**
  - Every arriving packet is assumed to “fly” to its output port
  - Better control over delay at output ports
  - **N times speedup** in crossbar and output buffer compared to input links
    - Expensive and infeasible in hardware

High time complexity for CIOQ switching

- **Combined Input- and Output-Queued Switch**
  - CIOQ can emulate OQ under **2 times speedup** [Chuang, S. (1999)]
  - One challenge remains: stable marriage problem requires Ω(N^2) iterations in the worst case

OQ Emulation under 2x Speedup

Problem Setup

- **Under arbitrary, but identical inputs** (arrivals, packet-by-packet)
- Is it possible to replace an OQ switch by a CIOQ switch and schedule it so that **the outputs (departures) are identical (per packet, per time slot)?**

Claims

- An algorithm, called Critical Cells First (CCF), **achieves OQ emulation at speedup of 2-1/N**; moreover, this is necessary and sufficient!
- The **output scheduling policy** can be any "monotone scheduling policy;" e.g., strict priority, WFQ, LIFO!

S.-T. Chuang, A. Goel, N. McKeown, and B. Prabhakar, "Matching Output Queueing with a Combined Input Output Queued Switch," IEEE INFOCOM, 1999.

OQ Emulation under 2x Speedup

- **Stable-Marriage algorithms**
  - **Each output ranks inputs waiting for it (how urgent a cell/packet is)**
  - **Proceed RG process until a stable matching is reached**
  - **Consider example input A(1,2,3), B(1,3,2), C(2,3,1) output 1(B,A,C), 2(A,C,B), 3(C,A,B)**
- **Why Stable Matching (Gale-Shapley) Works**
  - **Each output keeps a thread of which input's packet should depart next. Stable matching guarantees no thread conflicts.**
  - **Stability ensures no input-output pair would both prefer to be matched to each other over their current matching**

S.-T. Chuang, A. Goel, N. McKeown, and B. Prabhakar, "Matching Output Queueing with a Combined Input Output Queued Switch," IEEE INFOCOM, 1999.

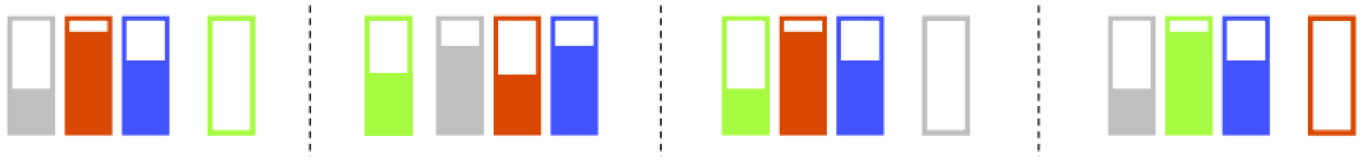
OQ Emulation under 2x Speedup

- **Critical Cells First (CCF) Algorithm (1999)**
  - **A cell must cross the switch when it becomes critical, i.e., when its slackness reaches zero: L(c) = OC(c) - IT(c) = 0**
  - **Output Cushion (OC): # of packets at the same output that depart before cell c**
  - **Input Thread (IT): # of packets ahead of c in its input queue**
  - **Critical Cell: one with L(c) = 0. It must be switched now to preserve OQ order.**
  - **Request-Grant Phases need switch's global information**
  - **Converge to stable matching**

S.-T. Chuang, A. Goel, N. McKeown, and B. Prabhakar, “Matching Output Queueing with a Combined Input Output Queued Switch,” IEEE INFOCOM, 1999.

OQ Emulation under 2x Speedup

- **Fully Local Gale-Shapley (FLGS) Algorithm (2006)**
  - **Goal: Achieve the same stable matching behavior as CCF, without global slackness.**
  - **Output ranking: port-level fair policy urgency / OQ order**
  - **Input ranking: simple fixed local rule rather than slackness**
  - **The VOQs are divided into two groups: empty and non-empty**
  - **All empty VOQs have rank "infinity", and when an empty VOQ has an arrival, it becomes ranked 1. It will remain in this rank until another empty VOQ receives an arrival, or it becomes empty, whichever is first.**



A. Firoozshahian, V. Manshadi, A. Goel and B. Prabhakar, "Efficient, Fully Local Algorithms for CIOQ Switches," IEEE INFOCOM 2007

Algorithm: SW-EDF

Solving the last challenge for practical CIOQ switching

Earliest Deadline First (EDF) for each scheduling phase

Near Perfect OQ emulation

Objective → **Key Mechanisms** → **Outcome**

Effectively reduce time complexity for CIOQ switching

Built on the Sliding-Window (SW) mechanism

Computes high quality matching in a single iteration

Georgia Tech

What is Earliest Deadline First (EDF)?

- What is deadline for every cell?
  - Defined by its departure time in the shadow OQ switch with the same traffic arrivals
  - Smaller the deadline, more urgent the cell is
- Assumption
  - The corresponding output knows cell’s deadline upon arrival
  - Orthogonal to the computation problem, providing how urgent the cell is
- Output ports propose / request
- Earliest Deadline First
  - During each scheduling time, the most urgent cell is selected by the output

What is Sliding Window (SW) mechanism?

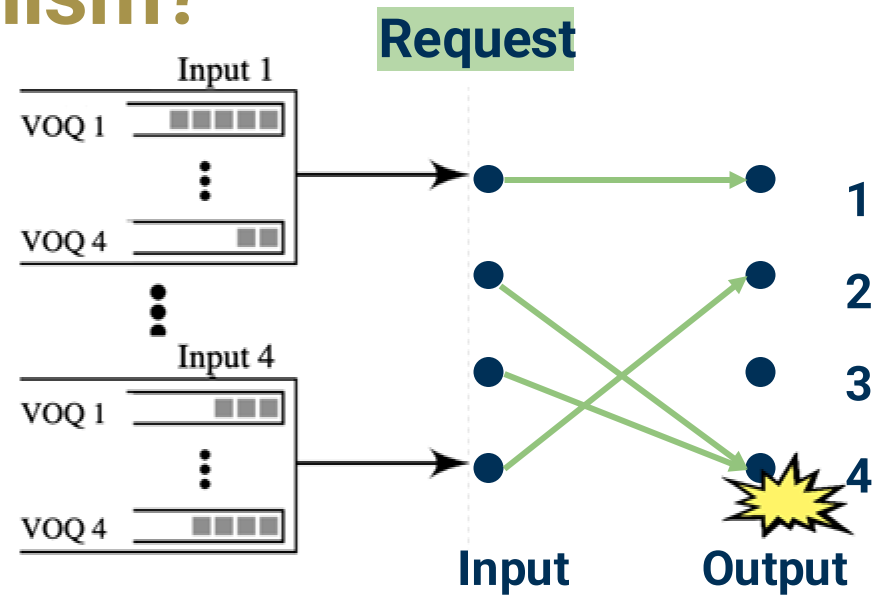
Small-Batch QPS Queue Proportional Sampling

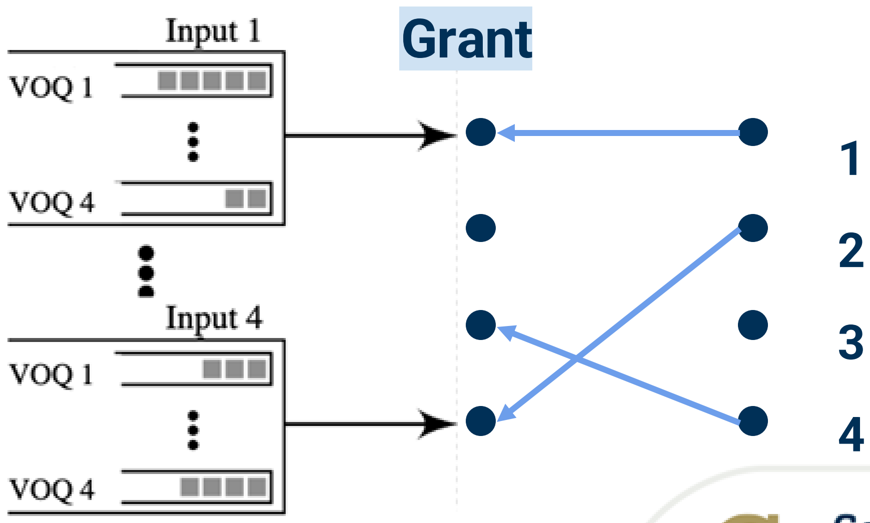
[Meng, J., (2021)]

- QPS: **Request** + **Grant**
- Batching strategy: batch size T
- Attains comparable throughput as iSLIP

T-length Time Schedules (Output View)

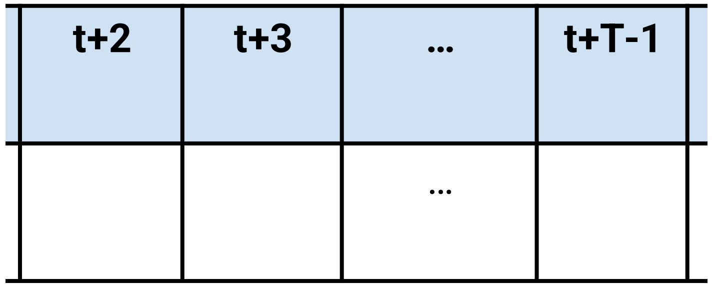
Time	1	2	3	4	...	T
O1	I1				...	
O2	I4				...	
O3					...	
O4	I3	I2			...	





What is Sliding Window (SW) mechanism?

- **Sliding-Window QPS** [Meng, J., (2021)]
  - Inherits all nice properties from SB-QPS while eliminating batching delay
  - Using T-bit-long bitmap that encodes input port i's availabilities during T time slots
  - Matching (schedule) for time t "graduates"



IQ switching is good enough on throughput performance, but we need further lower the packet delays to provide QoS

Algorithm 1: Request phase at output port j.

```
1 Procedure Request()
2     Select input port $i^*$ whose HOL cell has the
      earliest deadline;
3     Send output port $j$'s availability bitmap to input
      port $i^*$;
```

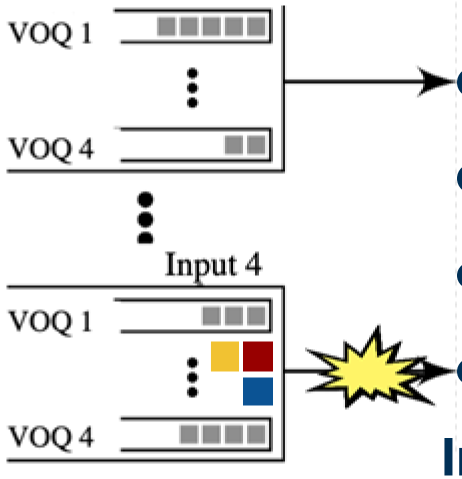
Algorithm 2: Grant phase at input port i.

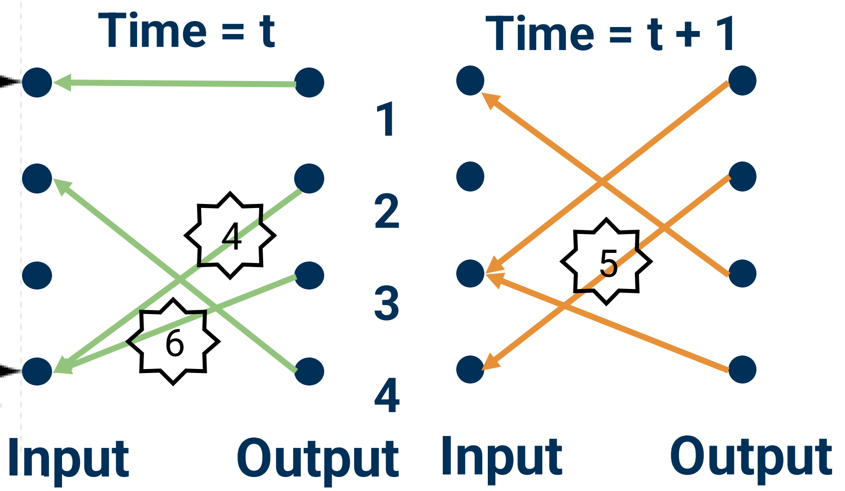
```
1 Procedure Grant()
2     if one or more proposals are received then
3         foreach such proposal (say from $j^*$) do
4             grant in the earliest time slot in $JAB(i, j^*)$;
```

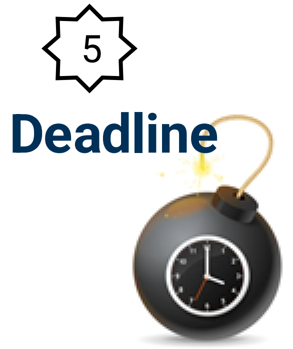
Within each SW-EDF iteration

Request Phase

Grant Phase







T-length Time Schedules (Input View)

Time	t+2	t+3	...	t+T-1	t+T	t+T+1
I1	O3		...			
I2			...			
I3	O4		...			
I4	O2		...			

4 5

Schedules (matching) generated by SW-EDF

Time	t	t+1	t+2	t+3	…	t+T-1
I1	O1		O3		…	
I2	O4	O4			…	
I3	O3	O1	O4		…	
I4	O2 🟥	O3 🟦	O2 🟨		…	
	4	6	5			

Schedules (matching) generated by Stable Marriage

Time	t	t+1	t+2
I1	O1	O3	
I2	O4		
I3		O1	
I4	O2 🟥	O2 🟨	O3 🟦
	4	5	6

Differences

- SW-EDF do not allow preemption
- Stable marriage can change the temporary connections (pairings).

Similarity

- In each scheduling phase, the most urgent cell of each output port (top preference in stable marriage) will be requested by SW-EDF.
- The schedule evolves dynamically within the sliding window

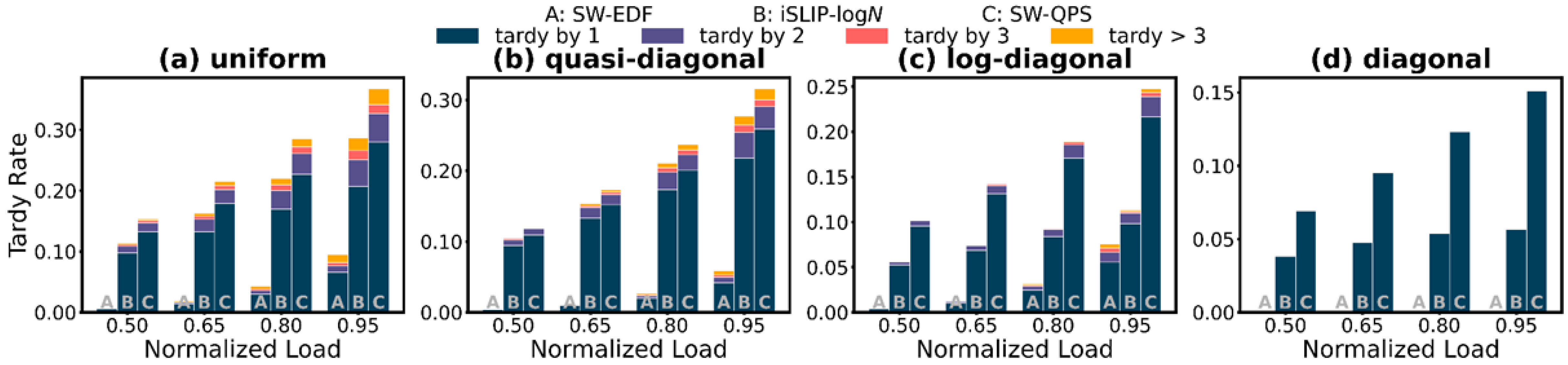
Simulation Setup

- **Switch Setting**
  - N=64, T=16, 2x speedup, scheduling policy: Weighted Fair Queueing
- **Evaluation Metrics**
  - Tardy rates: the percentage of cells that miss their deadlines
  - Tardy distribution: the distribution of tardiness amounts among all the tardy cells
- **Algorithms to compare with**
  - iSLIP: most widely used IQ switching algorithm
  - SW-QPS: the state-of-the-art IQ switching algorithm
- **Cell arrival process: a. Bernoulli arrivals**
  - (least skewed) Uniform, Quasi-diagonal, Log-diagonal, Diagonal (most skewed)

b. Bursty Arrivals

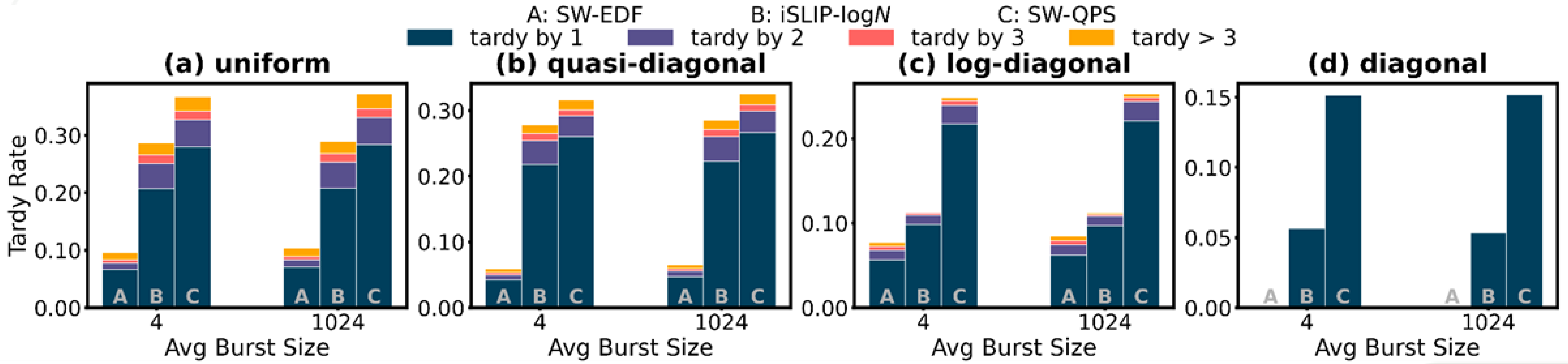
Evaluation

- Results: Near Perfect OQ emulation to provide Quality of Service (QoS)
- Tardy rates under Bernoulli arrivals
  - “missing columns” for SW-EDF: near-zero tardy rates
  - consistently below 3% under four traffic patterns even under load = 0.8



Evaluation

- Results: Near Perfect OQ emulation to provide Quality of Service (QoS)
- Tardy rates under bursty arrivals
  - run experiments with average burst sizes ranging **from 4 to 1024** cells & load factor of **0.95**
  - SW-EDF consistently outperforms iSLIP and SW-QPS by a significant margin



Conclusion

- IQ switch has simple hardware but no guarantee for delay.
- Ideal OQ switch gives the best delay and throughput but is impossible to build at scale.
- CIOQ switching bridges that gap by using small buffers and smart scheduling.
    - The Critical-Cells-First algorithm showed exact emulation with just 2× speedup — though it needed global coordination.
    - The Fully Local Gale–Shapley algorithm made that feasible using only local logic and simple input rankings while preserving stability.
    - SW-EDF solve the last high complexity challenge, balancing the performance and the computing effort.

Q&A

Thank you for listening!

Feel free to ask any questions

Georgia Tech