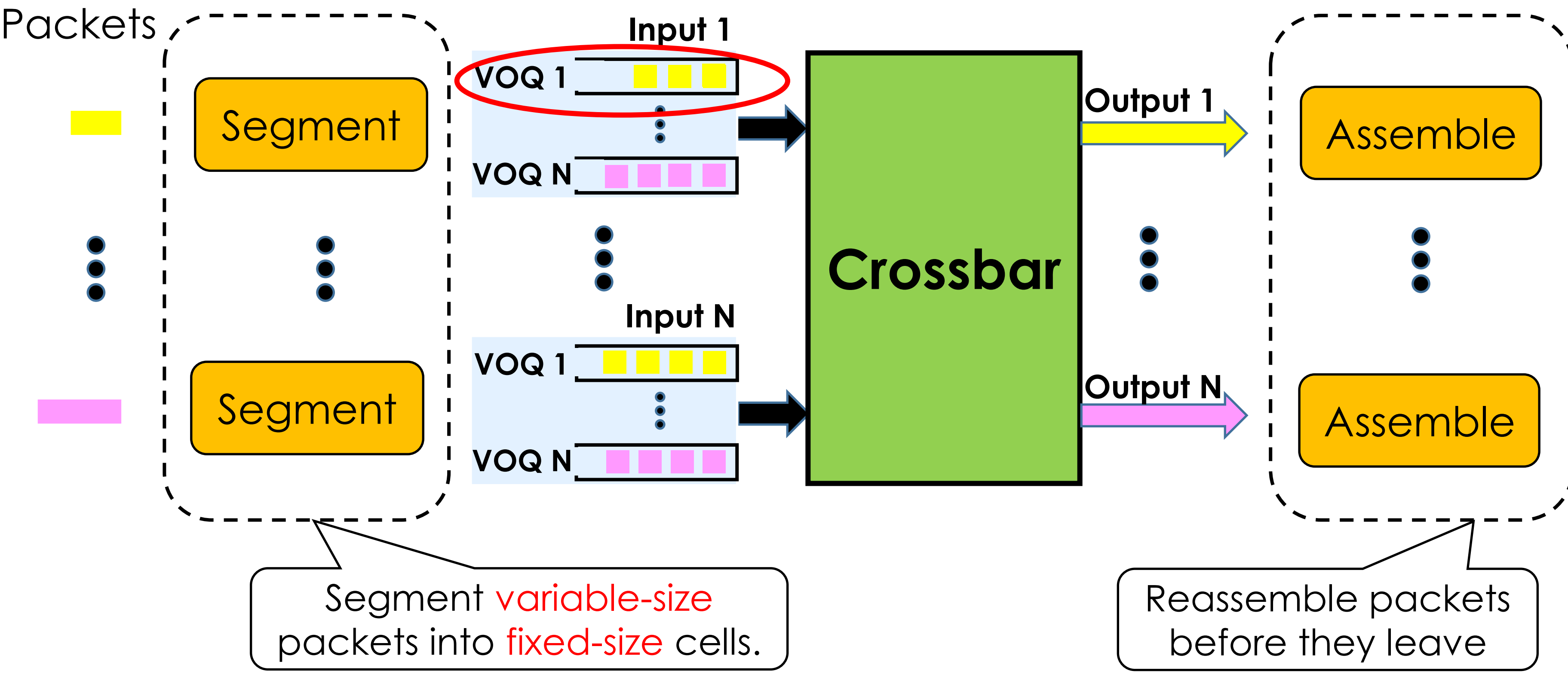
Sliding-Window QPS (SW-QPS): A Perfect Parallel Iterative Switching Algorithm for Input-Queued Switches

Jingfan Meng

Long Gong, Jun (Jim) Xu

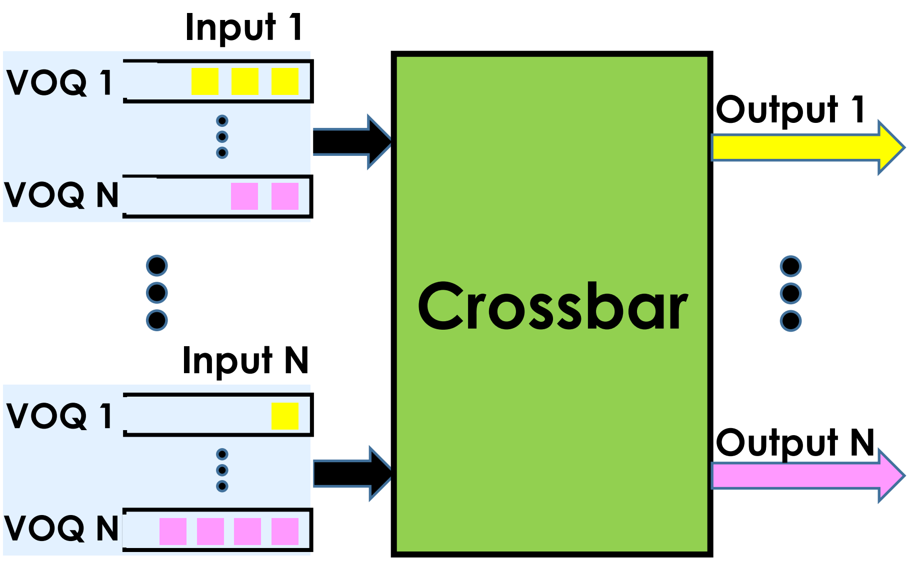
Georgia Tech

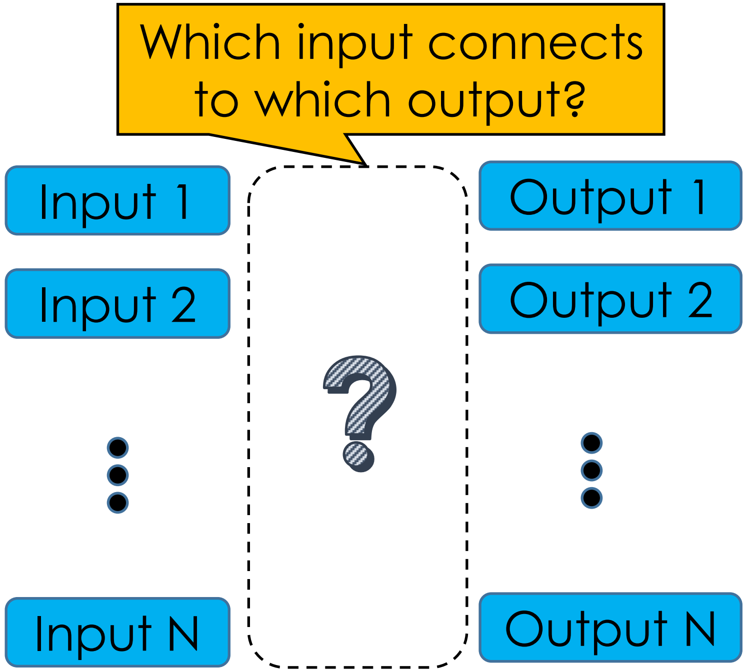
Input-Queued Crossbar Switches



- All the (input/output) ports and the crossbar operate at the same speed;
- This speed is normalized at 1.

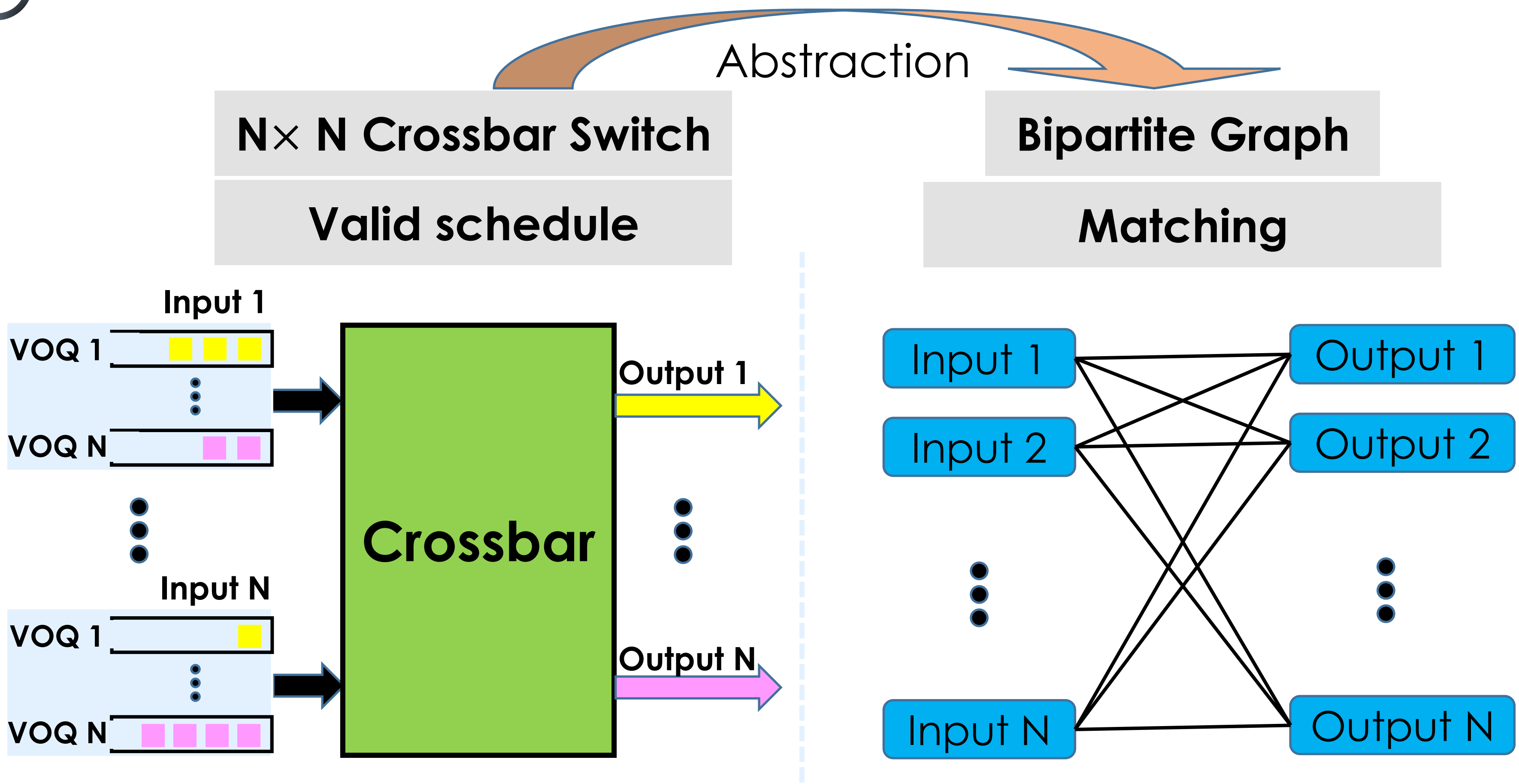
Crossbar Scheduling: Constraint





During each switching cycle, or time slot
- Each input can only connect to a single output
- Each output can only be connected by a single input

Crossbar Scheduling: Model



Crossbar Scheduling: Formulation

objective: matching

- ***maximize throughput***
- ***minimize (mean) delay***

strict timing constraint

StrataXGS Tomahawk 4 has **256** ports with **100Gbps** line rate. Supposing cell sizes are **128 bytes**, one (256x256) matching is required every **10ns**.

Implementation Constraint:

The algorithm should be simple to implement in hardware.

Crossbar Scheduling: Tradeoff

Quality of the matching

Time to compute the matching

Existing Research Work: Maximum Weighted Matching

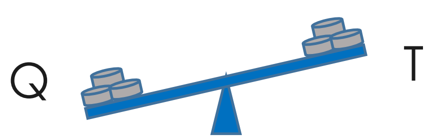
objective ***maximize throughput and minimize delay***

no timing constraint

Maximum Weighted Matching (MWM) [McKeown99a]

100% throughput and near-optimal empirical delay

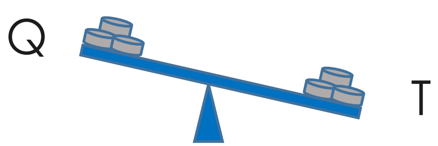
centralized $O(N^{2.5} \log W)$ time



Existing Research Work : Parallel Iterative Schedulers

objective ***maximize throughput and minimize delay***

subject to the timing constraint



iSLIP [McKeown99b] — widely used

QPS-1 [Gong20] — O(1) complexity

Their throughput and (high load) delay performances are much worse than MWM.

Roadmap

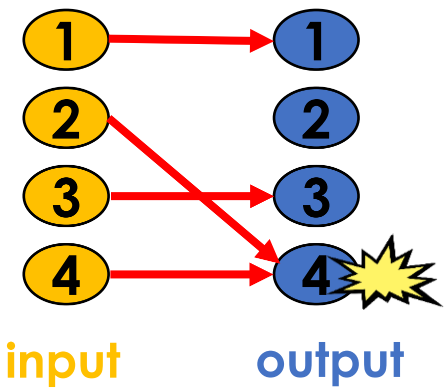
- **QPS-1 [Gong2020]**
- Basic framework (proposing and accept)

- **SB-QPS** (Small batch QPS)
- High throughput with a small batch size

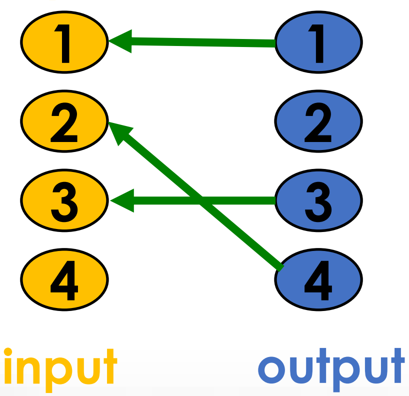
- **SW-QPS** (Sliding Window QPS)
- No batching delay

QPS-1 : Propose and Accept

QPS-1 [Gong20] computes each matching in the following two stages:



1. Proposing Stage (at input ports)
Each input port samples exactly one output port and proposes to it with the VOQ length. It uses a **O(1)**-time sampling algorithm called QPS [Gong17], in which the probability for each output to be sampled is **proportional to the corresponding VOQ length**.



2. Accepting Stage (at output ports)
Each output port accepts exactly one proposal with the longest VOQ length, if there is any.

SB-QPS: High Throughput in Small Batch

- SB-QPS schedules in **batches** (whose size T is small).
- Each batch consists of **T** matchings/time-slots, which is computed in multiple rounds of proposing and accepting stages. In this work, each batch is computed in **T** rounds: one round per time slot.
- The proposing stage of SB-QPS is almost the **same** as in QPS-1. The only difference is each proposal also includes the information concerning the **availability** of the corresponding input during each of the T time slots in the batch.

SB-QPS: High Throughput in Small Batch

- The accepting stage of SB-QPS attempts to accept all proposals if possible.
- When multiple proposals are received, the output port first sorts them in **descending** order of VOQ lengths and then attempts to accept them one by one on the **first commonly available** time slot (the first time slot in the batch for which both the proposer and the proposee are unmatched).
- For small batch size T = 16, the availability field fits in one machine word, and the first commonly available time slot can be found in **one** instruction.
- Therefore, the time complexity of both stages of SB-QPS is **O(1)**.

SW-QPS: Avoiding the Batching Delay

- SB-QPS pays a nontrivial batching delay of T time slots since it generates a batch of T matchings every T time slots.
- SW-QPS avoids the batching delay by generating one matching during each time slot.
- The only difference between SB-QPS and SW-QPS is when the matchings are generated. The two stages (proposing and accepting) are exactly the same.

SW-QPS: Sliding Window (Animation)

Time slot	t	t+1	t+2	…	t+T-1	t+T	t+T+1
Output 1				…			
Output 2				…			
Output 3				…			
Output 4				…			

scheduled at time slot t
scheduled at time slot t+1

Each matching (column) can possibly be filled in T time slots. (the same as in SB-QPS)

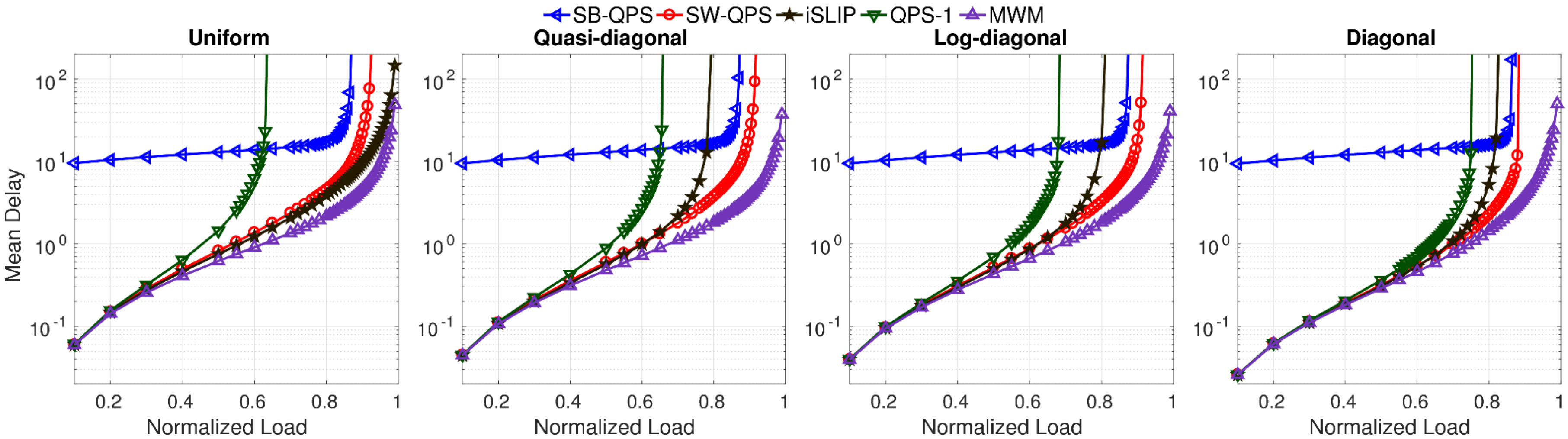
SW-QPS: Empirical Performance

Table 1: Maximum achievable throughput.

Traffic	Uniform	Quasi-diag	Log-diag	Diag	rounds / matching
SB-QPS	86.88%	87.10%	87.31%	86.47%	1
SW-QPS	92.56%	91.71%	91.40%	87.74%	1
iSLIP	99.56%	80.43%	83.16%	82.96%	O(log N)
QPS-1	63.54%	66.60%	68.78%	75.16%	1

Assumptions:
Independent **Bernoulli** arrival process
N=64 input and output ports, batch size **T=16**

SW-QPS: Empirical Performance



Assumptions:
Independent **Bernoulli** arrival process
64 input and output ports, batch size **T=16**

Conclusion

- We propose **SB-QPS**, a parallel $O(1)$ time crossbar scheduler that achieves good performance with a **small batch** size.
- We propose **SW-QPS**, which is based our new **sliding window** switching framework. SW-QPS inherits all the benefits of SB-QPS and reduces the batching delay to zero.
- We show, through simulations, that the throughput and delay performance of SW-QPS are much better than QPS-1, the state-of-the-art bipartite matching algorithm with parallel $O(1)$ running time.

References

- [Gong17] L. Gong, P. Tune, L. Liu, S. Yang, and J. Xu. Queue-proportional sampling: A better approach to crossbar scheduling for input-queued switches. Proceedings of the ACM SIGMETRICS, 1(1):3:1–3:33, June 2017.
- [Gong20] L. Gong, J. Xu, L. Liu, and S. T. Maguluri. QPS-r: A cost-effective crossbar scheduling algorithm and its stability and delay analysis. In Proceedings of the EAI VALUETOOLS, 2020.
- [McKeown99a] N. McKeown, A. Mekkittikul, V. Anantharam, and J. Walrand. Achieving 100% Throughput in an Input-Queued Switch. IEEE Trans. Commun., 47(8):1260–1267, Aug. 1999.
- [McKeown99b] N. McKeown. The iSLIP Scheduling Algorithm for Input-queued Switches. IEEE/ACM Trans. Netw., 7(2):188–201, Apr. 1999.