NYU·poly
POLYTECHNIC INSTITUTE OF NYU

CENTER FOR ADVANCED TECHNOLOGY IN TELECOMMUNICATIONS

Efficient Buffering and Scheduling for a Single-Chip Crosspoint-Queued Switch

Presenter: Zizhong CAO

Co-author: Prof. Shivendra Panwar

NYU WIRELESS

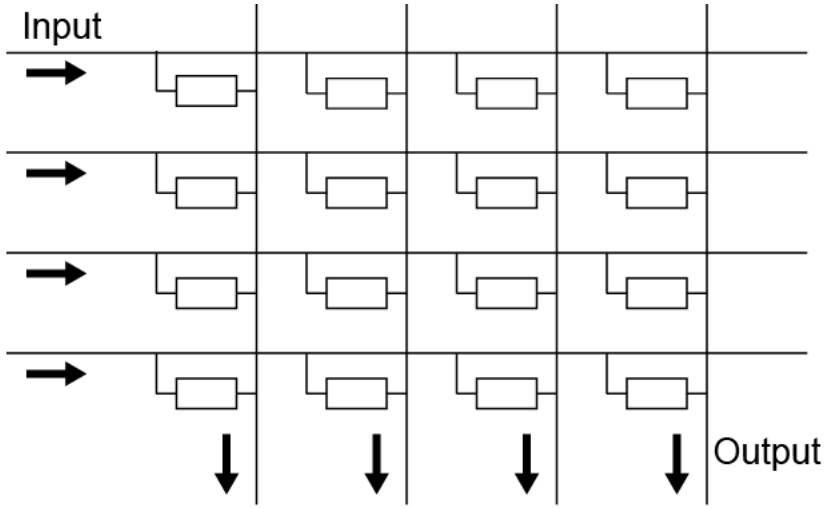
NYU

Introduction

- Background
  - Ubiquitous Internet services, e.g., video, social, cloud, etc.
  - Boom of portable electronic devices, e.g., smartphones, tablets, etc.
- Impact
  - Continuous, exponential growth in Internet traffic
  - Capability of Internet core switches must grow commensurately
- High performance switches and scheduling algorithms
  - High throughput, minimal delay
  - Low complexity, minimal speedup, small buffer size

Crosspoint-Queued Switch

- The single-chip crosspoint-queued (CQ) switch is a self-sufficient switching architecture.
- All buffering and scheduling are performed inside the switching core.



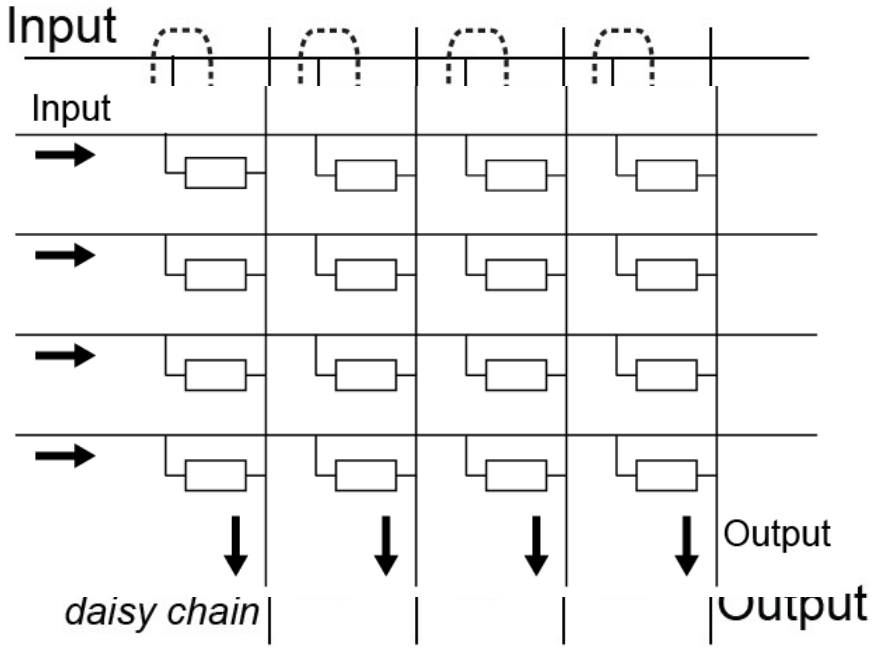
	OQ	IQ (w/o VOQ)	IQ (w. VOQ)	CICQ	CQ
Pros	high throughput minimal delay simple sched.	no speedup simple sched.	high throughput low speedup	high throughput low speedup separated I/O	high throughput minimal delay no speedup simple sched.
Cons	high speedup	low throughput high delay instant comm.	complex sched. instant comm.	complex sched. instant comm.	low buffer util. small chip

Contributions

- Architecture design
  - Chained crosspoint-queued (CCQ) switch
- Buffer sharing to mimic an output-queued (OQ) switch
  - Load balancing
  - Deflection routing
- Scheduling design for correct packet ordering
  - Oldest-Cell-First
  - Round-Robin with wait-counters
- Simulation and verification
  - Various synthetic and real Internet traces (bursty and non-uniform)
  - Meet practical needs

Chained CQ Switch

- Input/output: *1 ≤i,j ≤N*
- Crosspoint buffer: *0≤B(i,j) ≤B*
- Basic CQ switch
  - Small, segregated buffers
  - High drop rate
  - Longest-Queue-First (LQF) is almost the best
- Chained CQ switch
  - Connect crosspoints associated with a common output into a daisy chain in the order of input indices, e.g., *(i-1,j)--(i,j)--(i+1,j)*
  - Enable buffer sharing along the daisy chains
  - Support message passing to facilitate scheduling

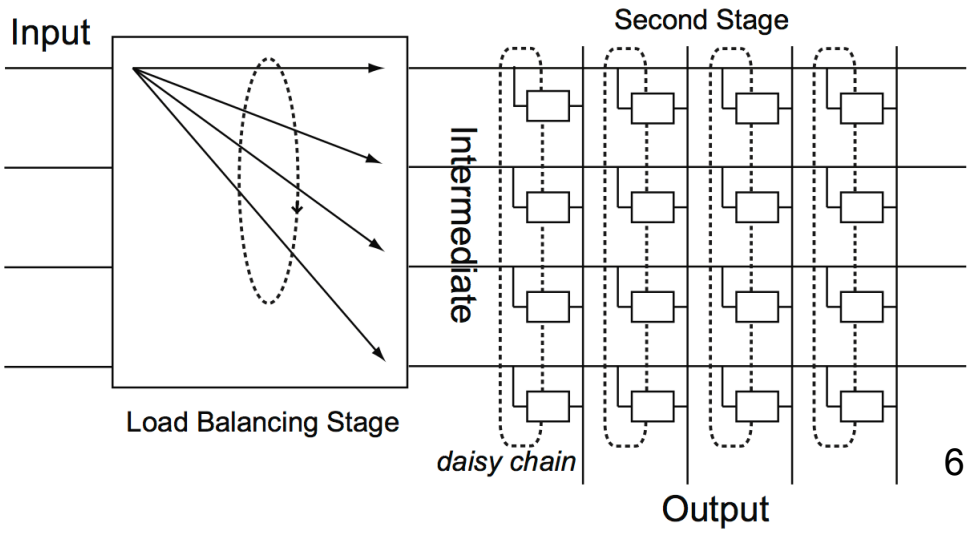


Efficient Buffer Sharing Techniques

- Load balancing
  - Extra first stage which walks through a fixed sequence of configurations
  - Same order as in daisy chains, i.e., at time *t*, input *i* is connected to intermediate port *i+t*
  - Passive mechanism which reduces burstiness and non-uniformity

- Deflection routing
  - Cell deflection between adjacent cells along the daisy chains, i.e., $(i,j) \rightarrow (i+1,j)$
  - From highly-utilized buffers to under-utilized buffers, i.e., $B(i,j) > B(i+1,j)$
  - Reactive strategy which balances the buffer utilizations

Mis-sequencing!



Packet Ordering

- Mis-sequencing
  - Cause: multi-path routing
  - Result: jitters in packet delay & incorrect packet order
  - Impact: TCP fast retransmission, fragmentation and reassembly, etc.

- Approaches to restore packet ordering
  - Re-sequencing buffers at the outputs, e.g., Fully-Ordered-Frame-First
  - Strict frame-based scheduling, e.g., Padded-Frame
  - Feedback-based scheduling, e.g., Mailbox switch

Extra buffering & scheduling delays.

Scheduling Design

- Oldest-Cell-First (OCF)
  - Straightforward
  - Per-output packet ordering
  - Timestamps can be expensive
  - Repeated comparison of head-of-line (HOL) cells

- Round-Robin (RR)
  - Lower cost and complexity using priority encoders
  - Serve crosspoints according to a pre-determined order
  - Need additional scheduling design to partly mimic OCF
  - Per-flow <input, output> packet ordering through cell alignment

CCQ-OCF

-Scheduling Design & Analysis

- ***Arrival Phase***
  - At time *t*, each new cell arriving at input *i* and destined to output *j* is sent to crosspoint *(i+t,j)* after load balancing, and is accepted at the tail of line (TOL) if that buffer is not full. A timestamp is also tagged to this new cell.

- ***Departure Phase***
  - Each output *j* serves the oldest cell after comparing the timestamps of the all HOL cells buffered at its associated crosspoints *(i,j), i=1,2,…,N*.

- ***Deflection Phase***
  - Each crosspoint *(i,j)* deflects the TOL cell to its successor *(i+1,j)* if *B(i,j)>B(i+1,j)*. The deflected cell is inserted into the ordered queue at crosspoint *(i+1,j)* according to its timestamp.

- Work conserving? *Yes.*
- Correct cell order? *Per output.*
- Complexity? *Quite high.*

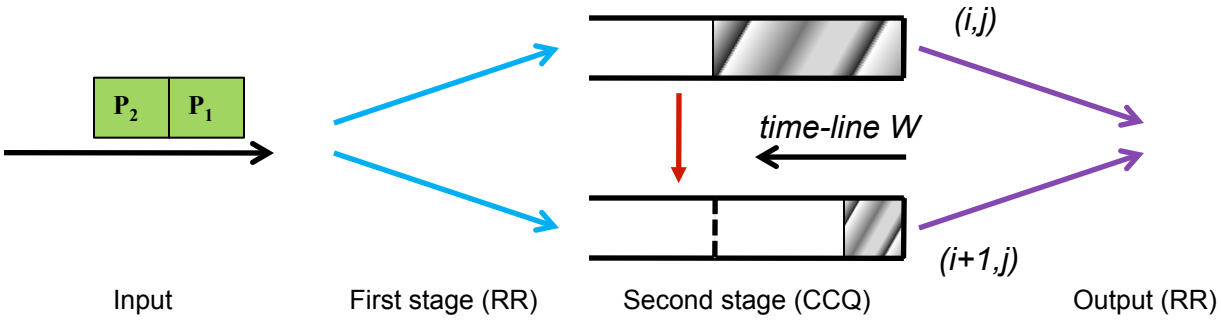
CCQ-RR

-Preserving packet order

- Wait-counter *W(i,j,k)* and round-counter *R(j)*
  - Wait-counters are maintained locally at each crosspoint and assigned to new cells upon acceptance.
  - Round-counters are maintained by each output which record the number of passed RR cycles.
  - Loop invariant: $R(j) \le W(i,j,1) \le \ldots \le W(i,j,B(i,j)) \le W(i,j,B(i,j)+1)$
  - Eligibility for departure: $W(i,j,1) = R(j)$

anticipatory

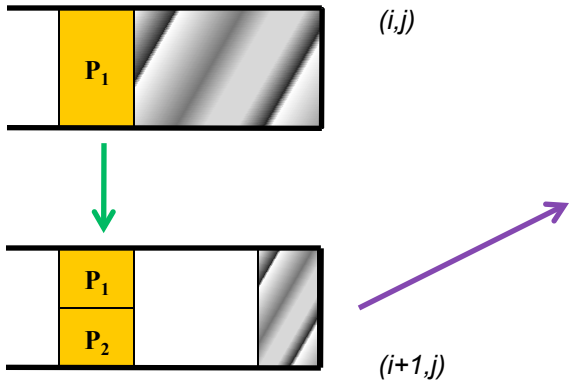
- Counter notification & cell alignment
  - Counter notification $CA(i,j)=W(i,j,B(i,j))$ is generated for each new cell P_1 upon acceptance by crosspoint (i,j) at time t, and passed down to successor crosspoint $(i+1,j)$ along the daisy chain.
  - Crosspoint $(i+1,j)$ should compare $CA(i,j)$ with $W(i+1,j,B(i+1,j)+1)$, and determine whether to increase $W(i+1,j,B(i+1,j)+1)$ and relay $CA(i,j)$ along the daisy chain, or to drop $CA(i,j)$.
  - The next cell P_2 belonging to the same flow will arrive at crosspoint $(i+1,j)$ at time $t+1$. Since its wait-counter $W(i+1,j,B(i+1,j)+1)$ has already been aligned with $W(i,j,B(i,j))$, it will be served later then P_1.



CCQ-RR

-Preserving packet order (cont'd)

- Special design for deflection routing
  - The direction of deflection can only be from a highly-utilized crosspoint to its under-utilized successor.
  - Only TOL cells can be deflected, and their wait-counters should be carried.
  - The wait-counter of a deflected cell may overlap with that of an existing cell already buffered at the receiver crosspoint, but it should be inserted in front of it in the ordered queue.
  - Outputs should employ an exhaustive RR method to serve all cells with *W(i,j,1)=R(j)* at crosspoint *(i,j)* before proceeding to the next crosspoint *(i+1,j)*.



Second stage (CCQ)

Output (RR)

CCQ-RR

-Scheduling Design & Analysis

- ***Arrival Phase***
  - Same as in CCQ-OCF, except that the existing anticipatory wait-counter is assigned to this new cell, and a new anticipatory wait-counter is generated $W(i,j,B(i,j)+1)=W(i,j,B(i,j))+1$.
- ***Notification Phase***
  - Each crosspoint may initiate or relay a counter-notification message to its successor. Then it receives another message from its predecessor and make necessary updates to its own wait-counters.
- ***Departure Phase***
  - Each output *j* polls its associated crosspoints in an exhaustive RR fashion, until it finds an eligible crosspoint with $W(i,j,1)=R(j)$ and serves the HOL cell, or it stops after a certain number of polls.
- ***Deflection Phase***
  - Each crosspoint (i,j) deflects the TOL cell to its successor $(i+1,j)$ if $B(i,j)>B(i+1,j)$. The deflected packet is inserted into the ordered queue at crosspoint $(i+1,j)$ according to its wait-counter.

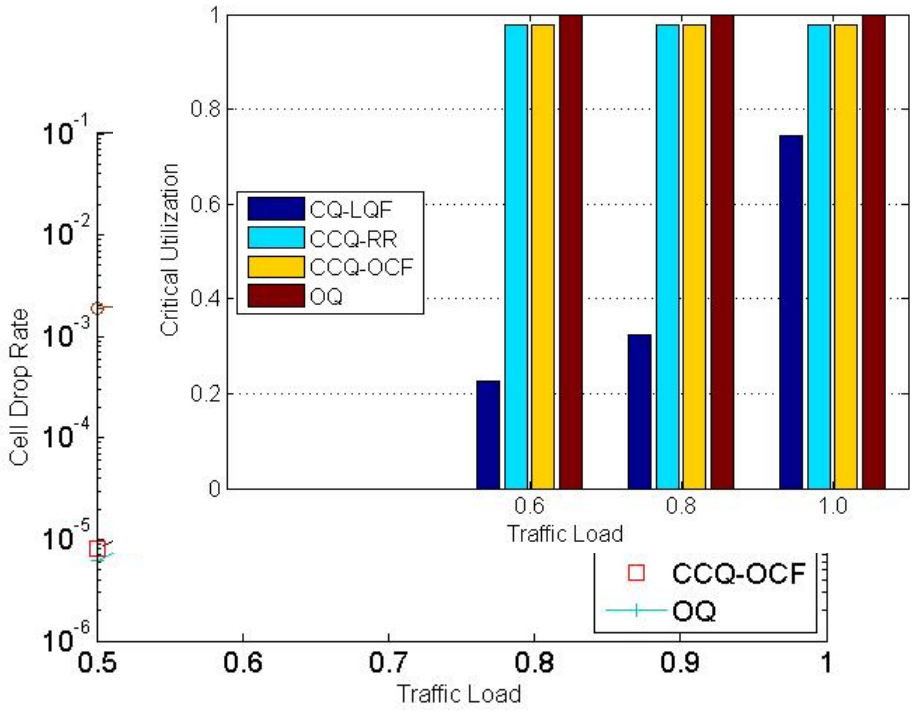
- Work conserving?
  Yes, if the maximum number of deflections is restricted to K, and the output can poll N+K+1 crosspoints in each time slot.
- Correct cell order? *Per-flow <input, output>.*
- Complexity? *Much lower.*

Simulation

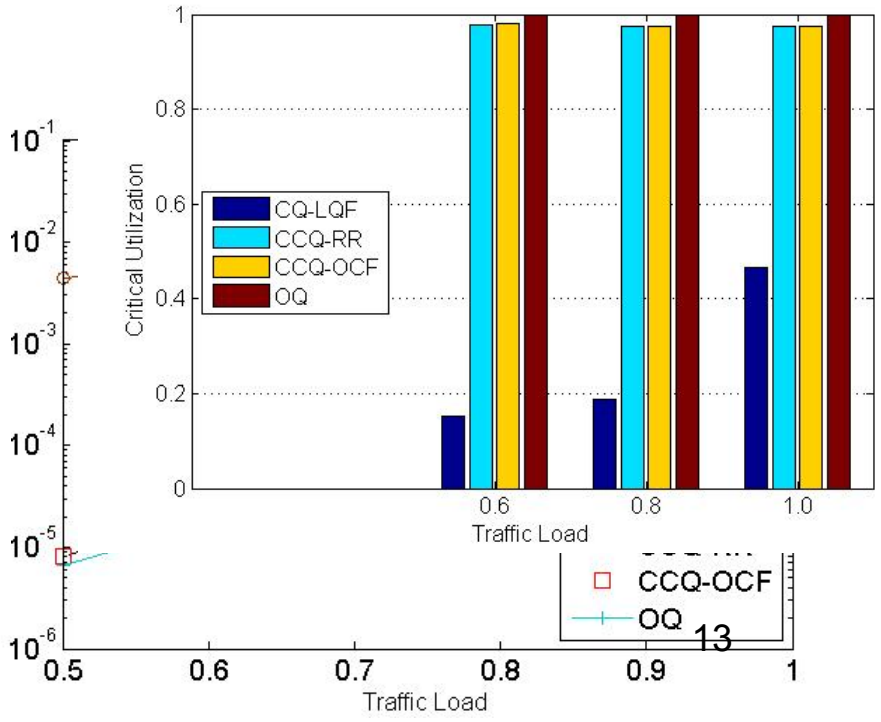
-Uniform and Non-uniform Traffic

- MATLAB, 10^7 time-slots
- Uniform vs. Hot-spot long-range-dependent (LRD) traffic with H=0.75
- Non-uniform → larger gain

32x32x40 switches under ***uniform*** LRD traffic



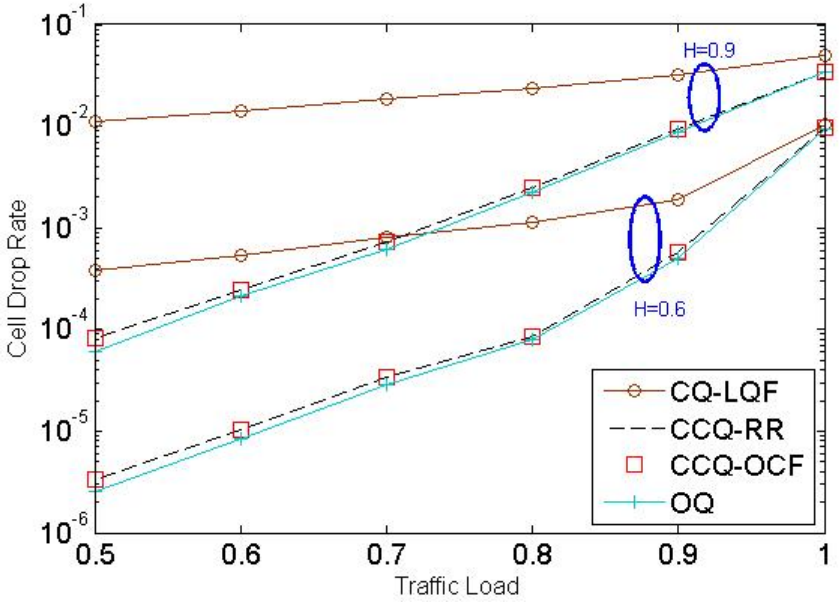
32x32x40 switches under ***hot-spot*** LRD traffic

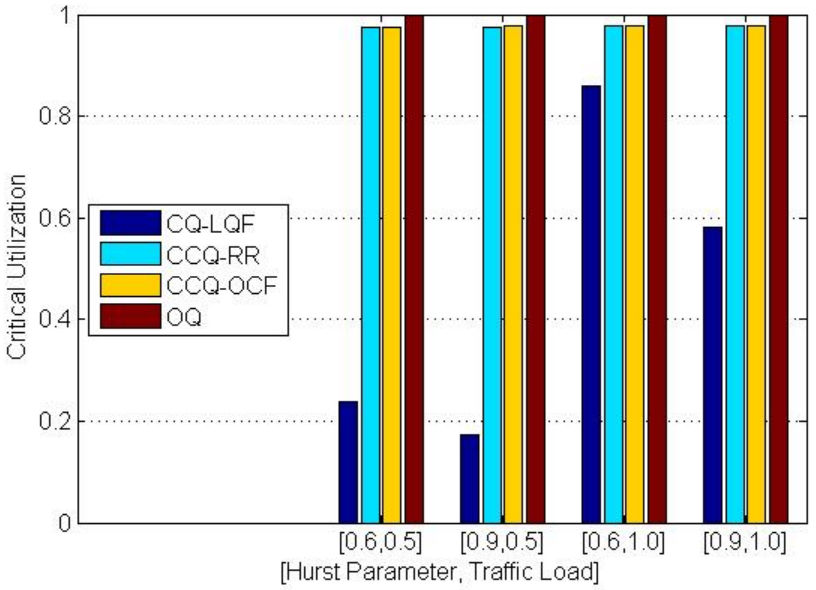


Simulation

-Burstiness

- More bursty, lower load → larger gain





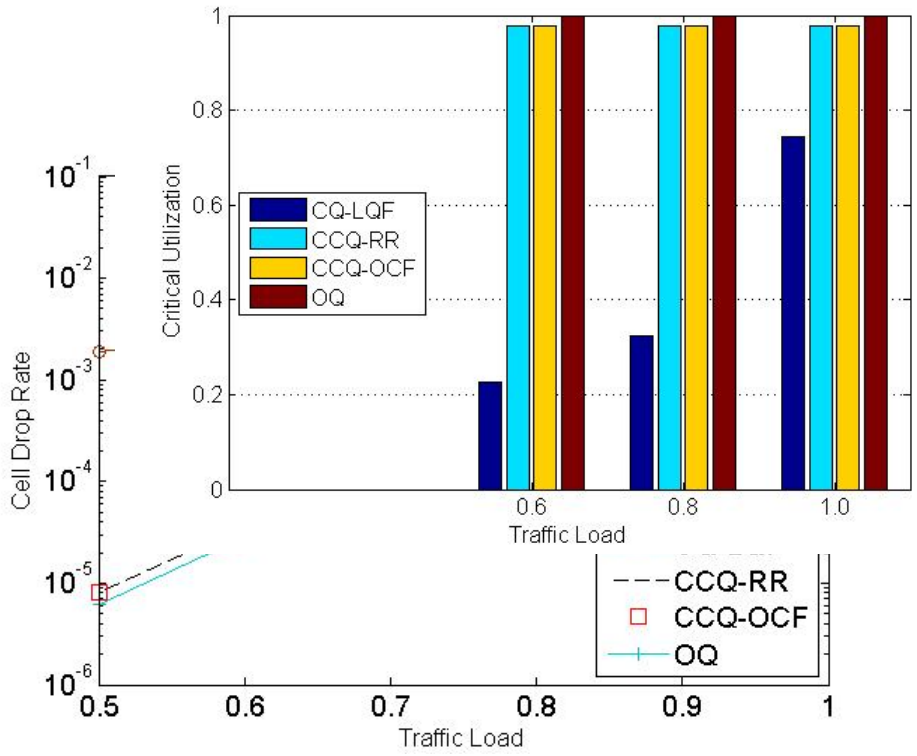
32x32x40 switches under uniform LRD traffic with ***0.6 ≤ H ≤ 0.9*** and ***0.5 ≤ λ ≤ 1.0***

Simulation

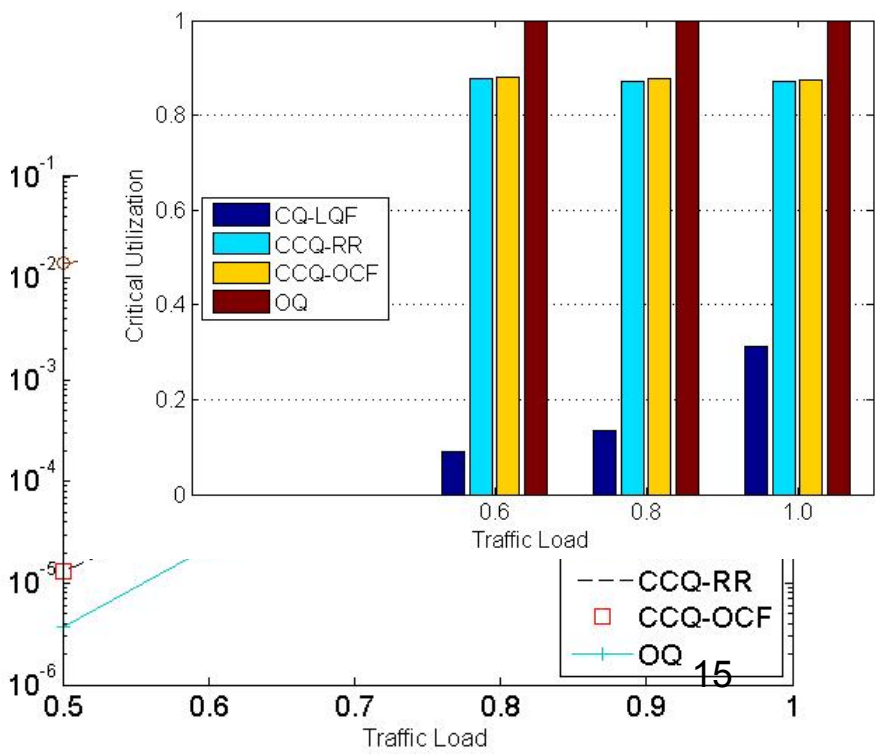
-Switch Size

- Larger switch, smaller crosspoint buffers → larger gain

32x32x40 switches under uniform LRD traffic



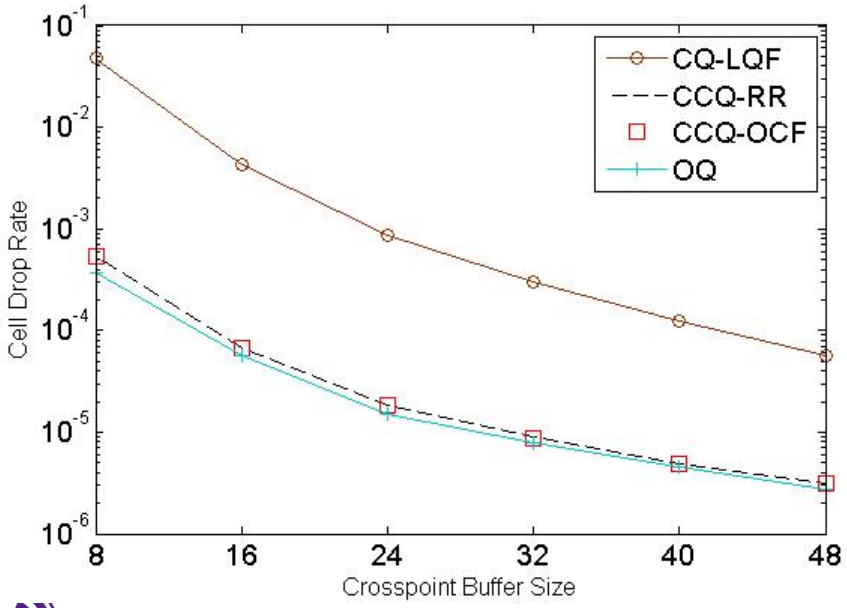
128x128x10 switches under uniform LRD traffic



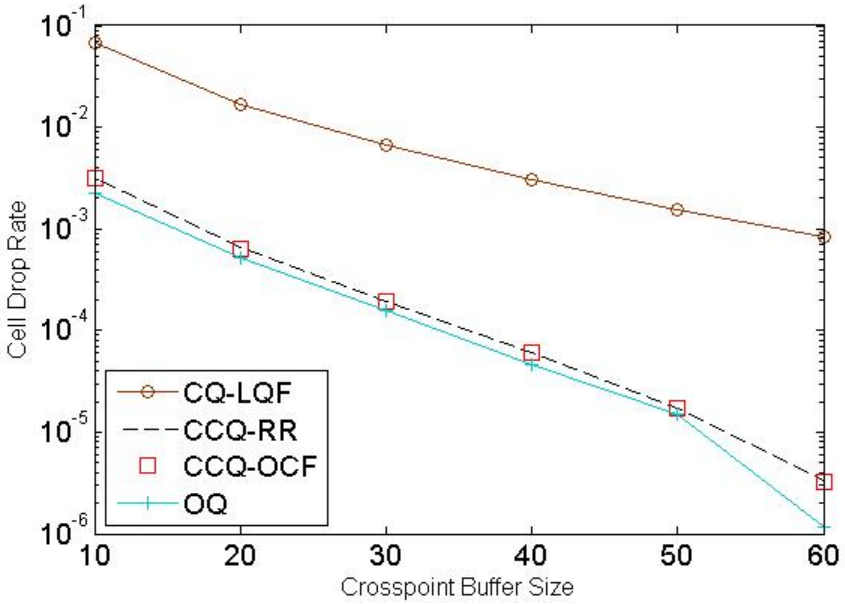
Simulation

-Real Internet Traces

- CAIDA OC-192 (10Gbps) Internet traces
- Total buffer size: 32 input × 32 output × 32 cells × 64byte = 2Mbyte,
  128 input × 128 output × 60 cells× 64byte = 60Mbyte



32x32 switches under ***CAIDA*** trace with ***λ=0.45***



128x128 switches under ***CAIDA*** trace with ***λ=0.7***

NYU WIRELESS

NYU

Conclusion

- We have raised efficient buffer sharing techniques and scheduling algorithms to improve the buffer utilization of a single-chip CQ switch and reduce the overall packet drop rate, while still preserving the correct packet order. Effectively, we let the CCQ switch mimic an OQ switch without the factor-of-N speedup.

- Simulation results show that the CCQ switch may meet practical needs using state-of-art ASIC technology. Our design gains more advantages in large switches with smaller crosspoint buffers, and when the incoming traffic is bursty and non-uniform.

- Future work:
  - Improve the algorithm and mathematically bound the performance.
  - Push the limits of buffer sharing across different outputs.
  - Investigate chip area and power consumption though hardware simulation.