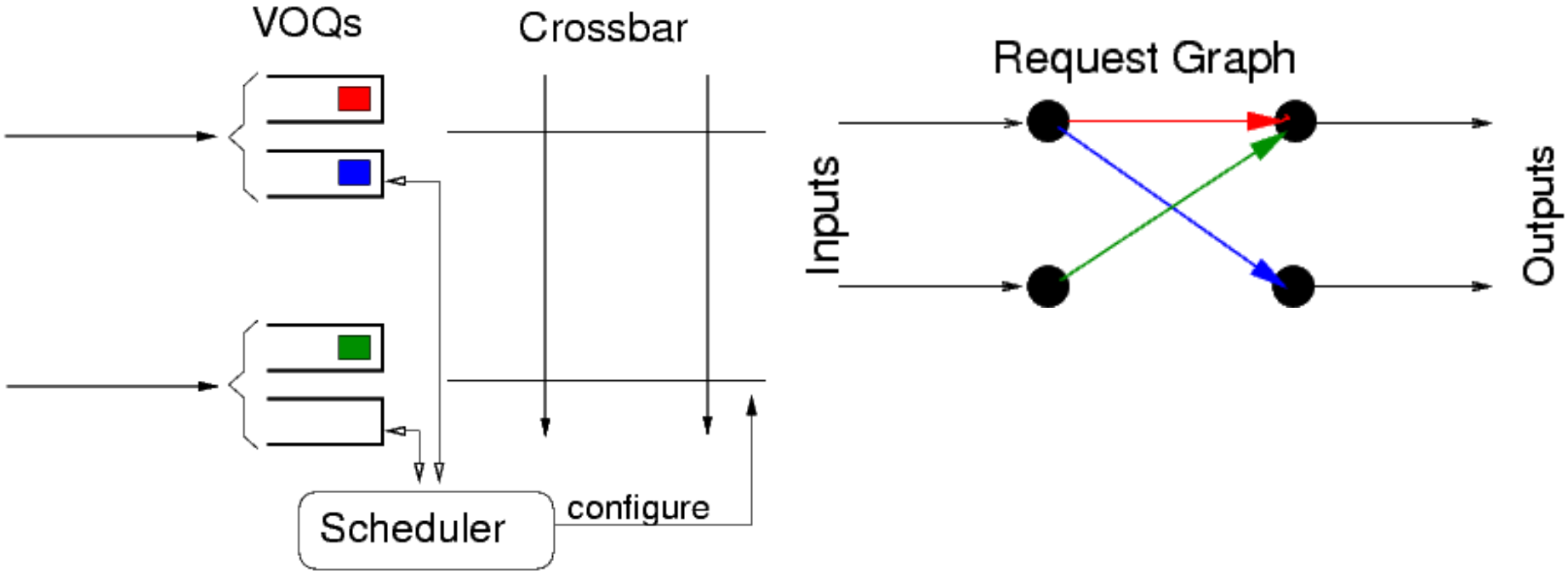
Weighted Fairness in Buffered Crossbar Scheduling

Nikolaos Chrysos and Manolis Katevenis

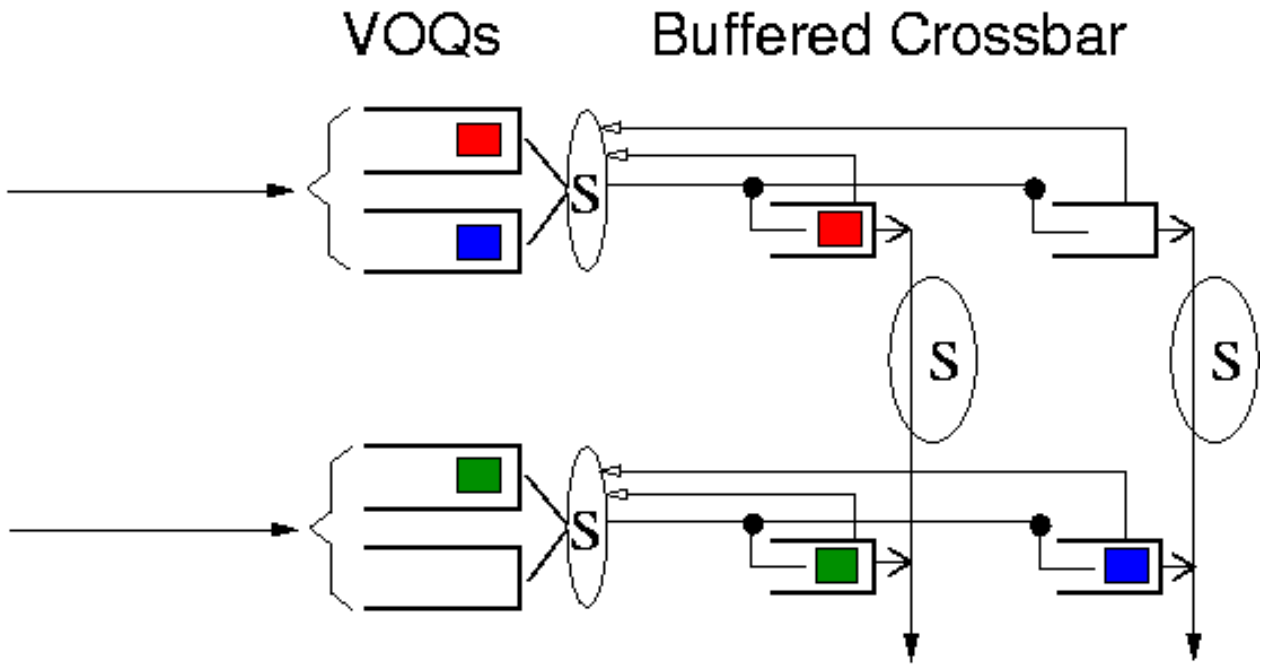
FORTH & Univ. of Crete, Greece

Crossbar Scheduling



- Multi-Resource Scheduler: Complexity Cost & Scalability issue
  - 1 cell per input => per-output decisions depend on each other
  - 1 cell per output => per-input decisions depend on each other
- Need multiple iterations and/or internal speedup (CIOQ)
- Fairness is an issue – weighted fairness is very hard
- Works only with fixed-size cells (segments)

Buffered Crossbar



- Independent schedulers, loosely coordinated via backpressure
- No global time frame needed
  ➔Variable-size packets acceptable
- Advanced QoS via sophisticated local disciplines
  – RR-RR; LQF-RR; WFQ-WFQ
- This paper: WFQ-WFQ ➔Weighted Max-Min Fairness

Challenges, Issues

- Size of the Crossbar Chip Memory
  - Modern CMOS: several Mbits per chip
    => few Kbytes per crosspoint feasible
- Backpressure Overhead
  - Peak Rate: N per Input per Time-Slot
    (when all outputs select the same input at the same time-slot)
  - Average Rate: 1 per Input per Time-Slot

minor compared to the advantages

This Work

- Buffered Crossbar with WFQ/SFQ Schedulers & fixed-size cells
  - Arbitrary weights: Input Fair Share $\neq$ Output Fair Share
- We find that bandwidth allocation approximates
  Weighted Max-Min (WMM) Fairness
- We study the impact of:
  - Crosspoint Buffer Size
  - Weight Factors
- Results:
  - 2-6 cells/crosspoint ➔1-5% accuracy (32x32 switch)
  - 2-4 cells/crosspoint ➔output utilization > 99%

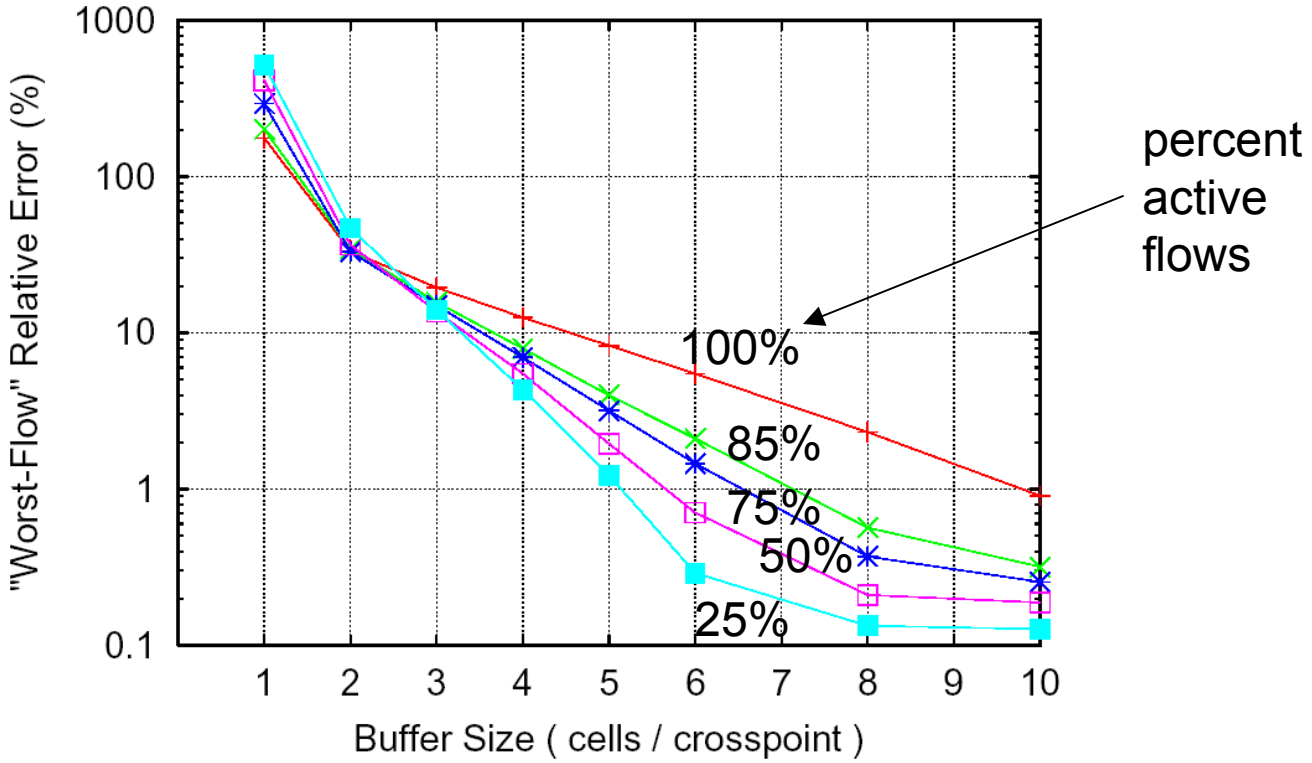
Simulation Environment

- No speedup
- Persistent Sources (X% of them Active)
  - Models the short-term Switch Operation
  - Output utilization approx. 100% for most outputs, owing to excess Bandwidth redistributions
- Metric: Relative Error % (RE %)

$$100 \times \frac{|\ FairService(f) - SimulatedService(f)\ |}{FairService(f)}$$

- RTT = 1 cell time
- Each plot point = average of many (>20) runs with different/random weights

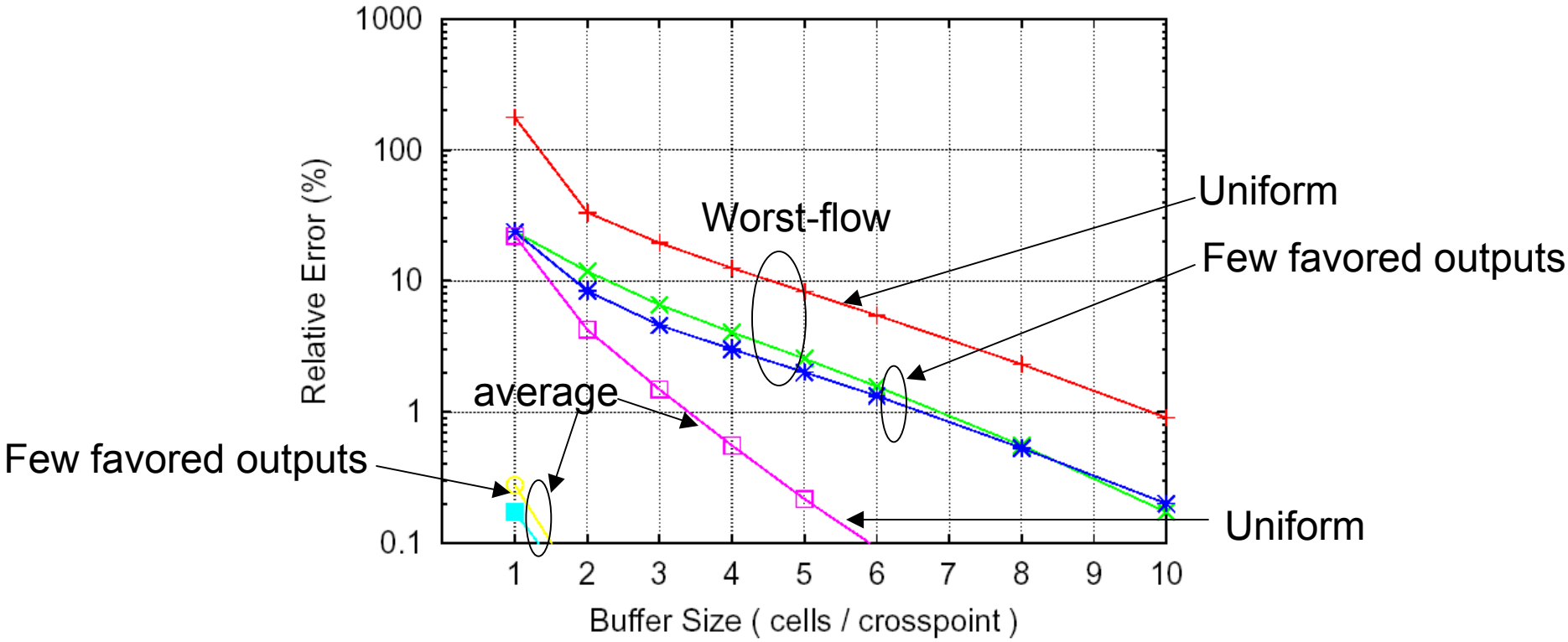
Crosspoint Buffer Size needed to achieve Low WMMF Error



- Worst-Flow RE < 5% with 6 cells/crosspoint
- Need more buffers to reduce RE
- But small buffers => less time to re-converge
  - ~ time for xpoint buffers to empty or to fill

*(32x32 switch , w_{IJ}= **uniform_random(10,1010)**)*

Effect of Weight Distribution



- Average RE < 1% (buffer size >= 4)

  => High output utilization

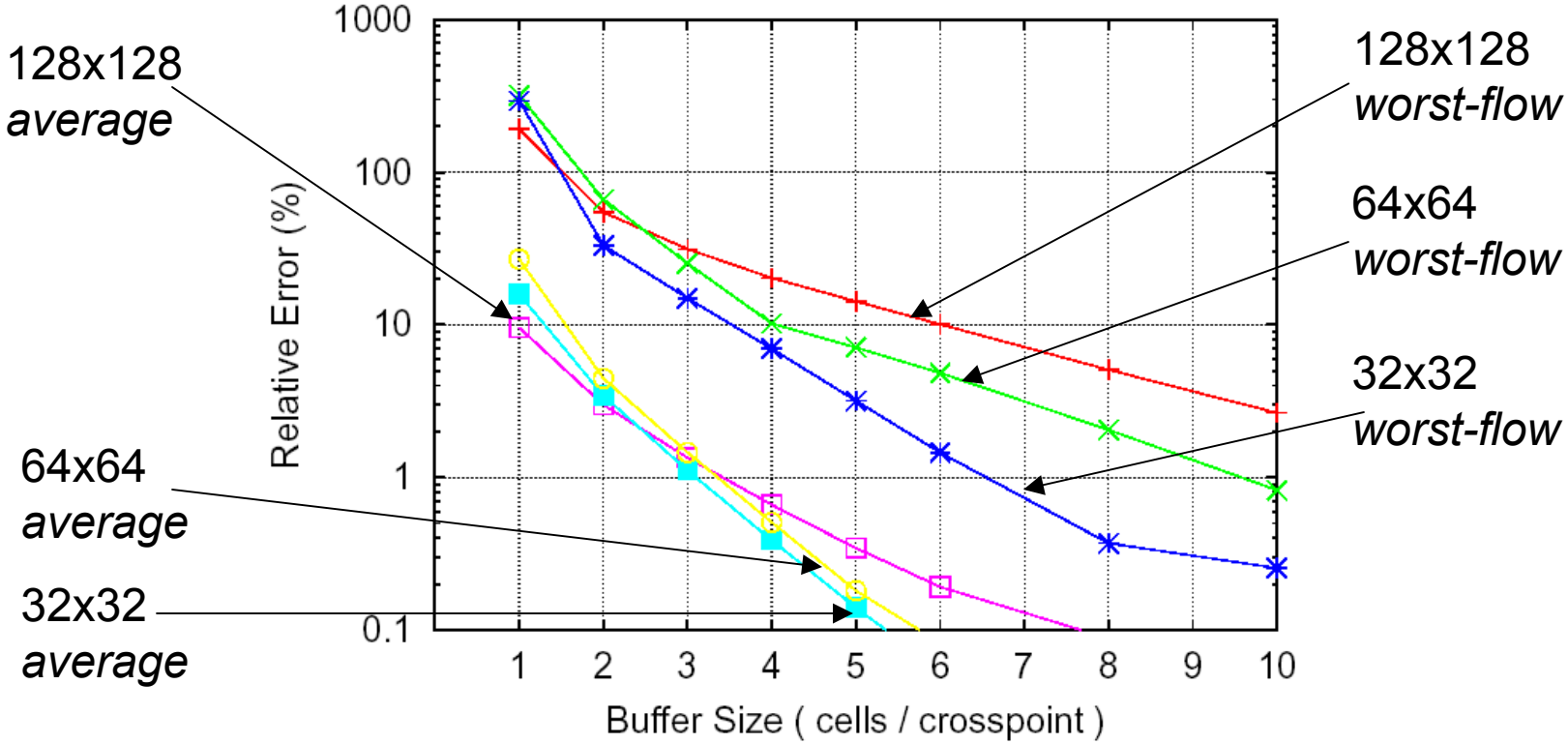
- Uniform distribution ➔ worst results

(32x32 switch , 75% active flows,

Uniform-weights: w_{IJ} = **uniform_random(10,1010)**

Few-favored-output: w_{IJ} = **$10J^2$ + uniform_random(0,$10J^2$)**)

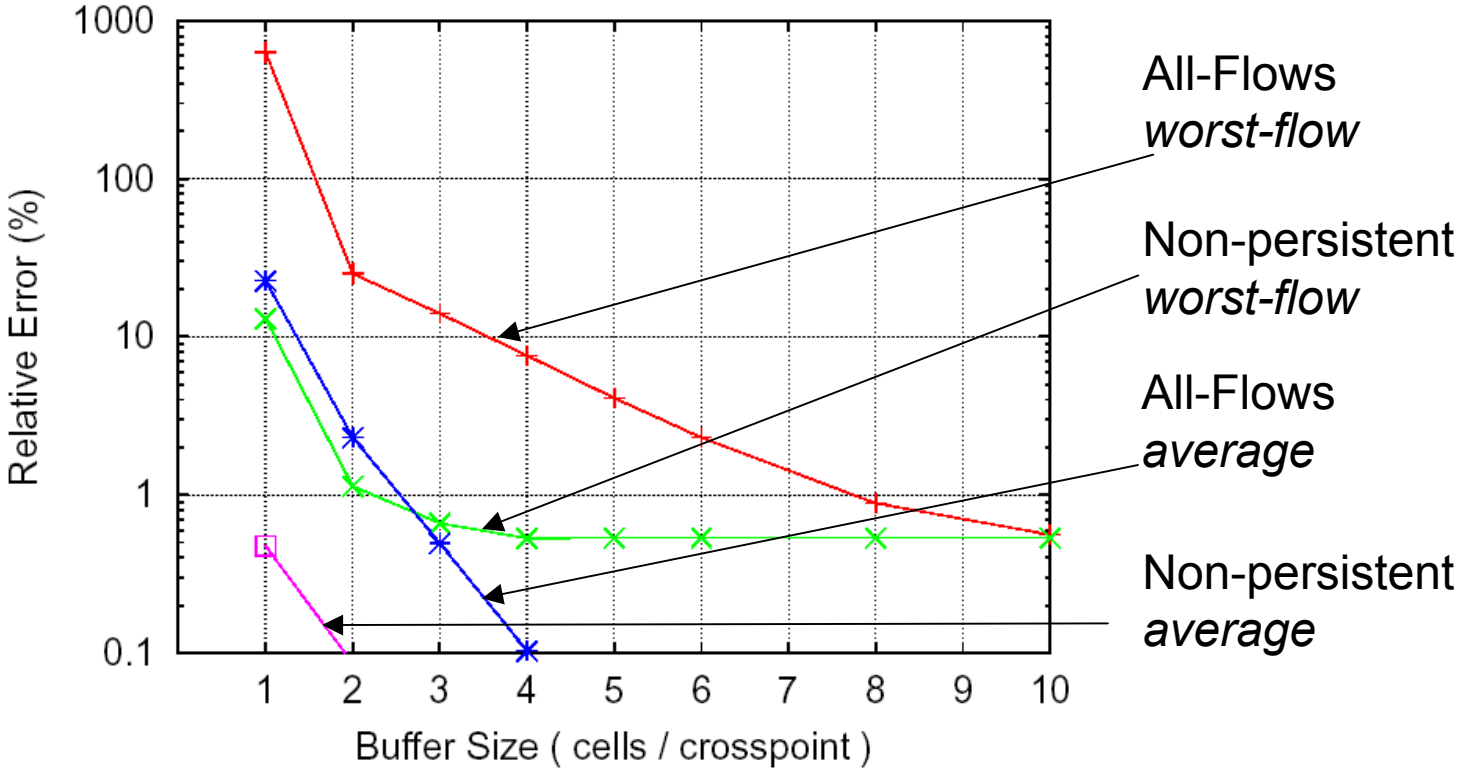
Effect of Switch Size



- Worst Flow RE increases with Switch Size
  - More flows → more service jitter
- Average error…
  - Many flows with good accuracy
- WF^2Q+ smaller jitter ➔ better convergence accuracy

(75% active flows, uniform weights distribution)

Not-Persistent Flows



- One flow at each input with arrival-rate:
  - MIN(InputFairShare, OutputFairShare)
  - Less than WMM Fair rate
- Excess Bandwidth Redistributed Fairly

(75% active flows, uniform weights distribution)

Conclusions

- Buffered Crossbars
  - Scheduler Simplicity/ Sophistication
  - High throughput & Delay Guarantees
  - WMM Fairness
- Our Results:
  - High accuracy to Weighted Max-Min Fairness with 3-8 cells/xpoint
  - Service jitter and small crosspoint buffers reduces accuracy, but
  - Small xpoint buffers improve speed of convergence
- Current Research: directly switching variable Size Packets
  - Eliminating Speed-Up, Output Buffers, Fragmentation & Reassembly

Operation & WMM Fairness

- When crosspoint buffers empty
  - *Input Service = Input Fair Share*
  - *Some Flows Ineligible at Outputs*
- As crosspoint buffers fill-up
  - *Output Service ~ Output Fair Share*
  - *Some Flows Ineligible at Inputs*
- Eventually WMM Fairness -- maximizes MIN (Bandwidth / weight)
  - *Input Fair Share(f) > Output Fair Share(f)*
    - ➔ *crosspoint buffer(f) usually FULL*
  - *Output Fair Share(f) > Input Fair Share(f)*
    - ➔ *crosspoint buffer(f) usually EMPTY*