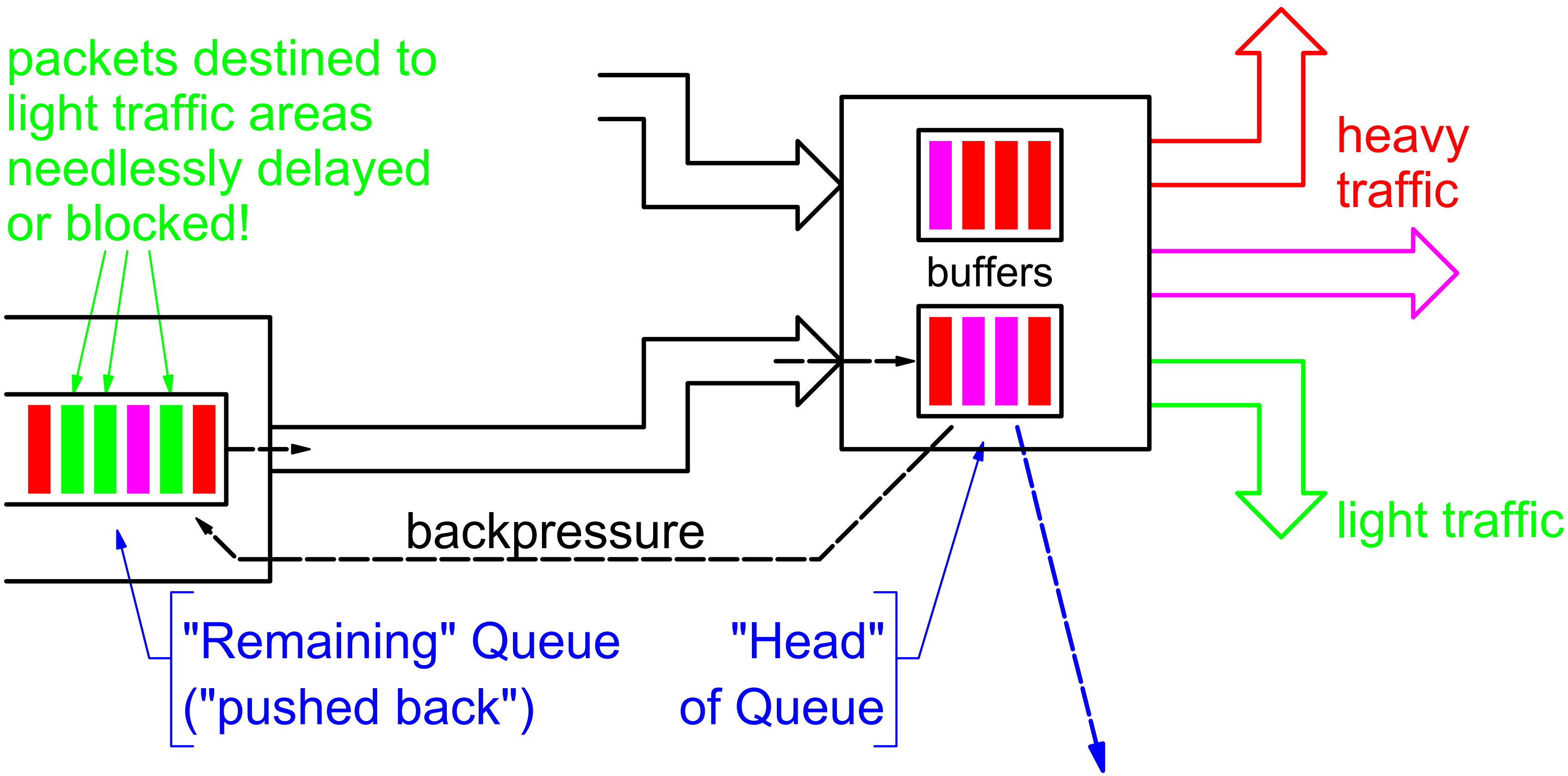
6.2 Per-Flow Queueing and Flow Control

- Indiscriminate flow control causes local congestion (output contention) to adversely affect other, unrelated flows
  - indiscriminate flow control causes phenomena analogous to HOL blocking, independent of how many queues there are, when the equivalent of a “queue” spreads across multiple switches in a fabric
- Shared queues cause fairness problems between the flows that share them
  - service rates determined by the original sources, rather than by a scheduler at the contention point
  - even with FC feedback, shared queues delay policy enforcement
- The solution: Per-Flow Queueing and Flow Control
  - keep the “head” of each flow’s queue near its intended output
  - keep the “bulk” of each flow’s queue in its input buffer(s)
- Application: Buffered Crossbars (CICQ – Comb. Input-Crosspt. Q.)

Indiscriminate Lossless FC => Head-of-Line Blocking



Solution 1:

lossy flow control...

Solution 2:

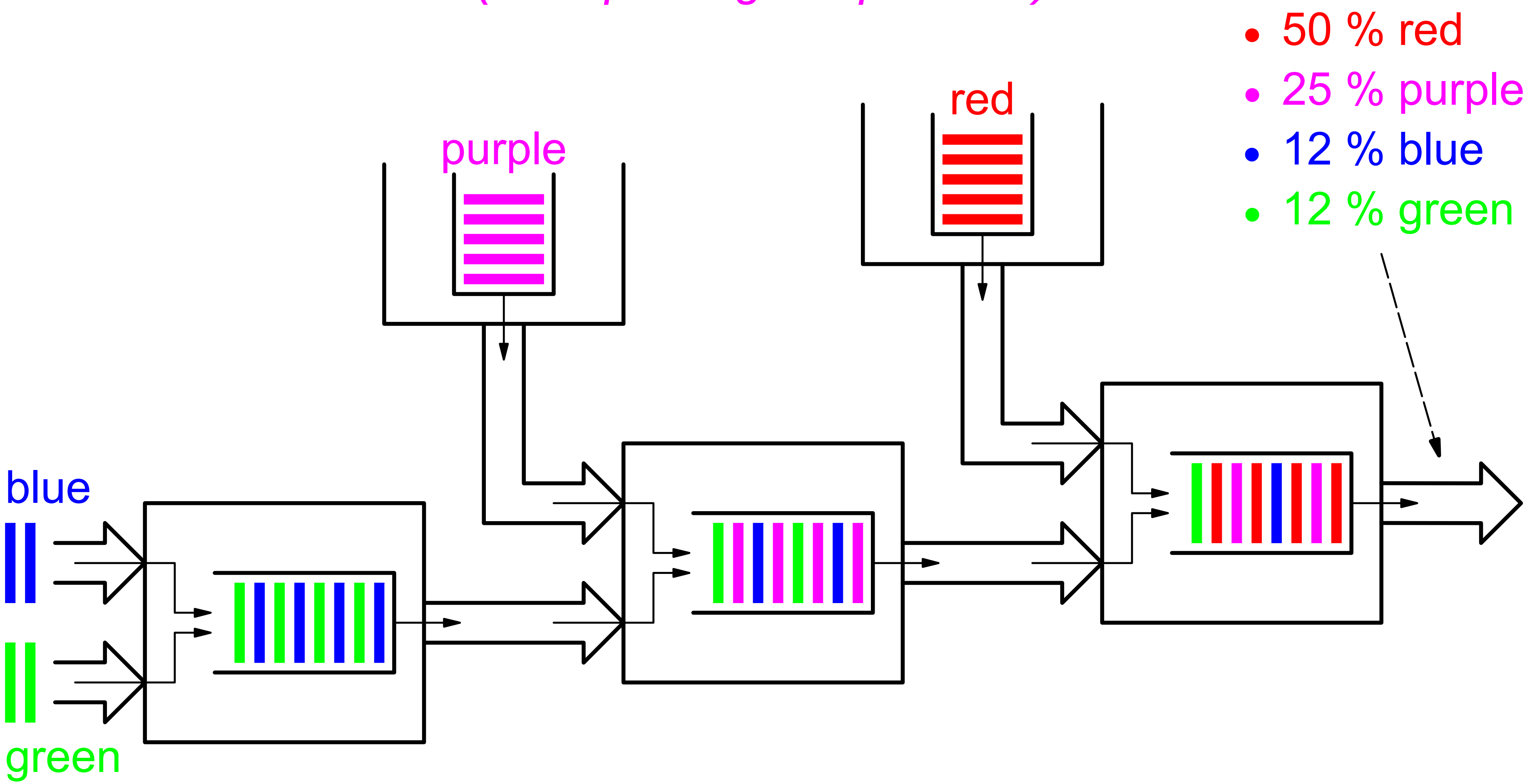
Per-Flow
queueing & flow control

With any queueing discipline (FIFO or not) this switch has only access to and can only schedule packets in this limited buffer space

⇒ similar to Head-of-Line (HOL) Blocking!

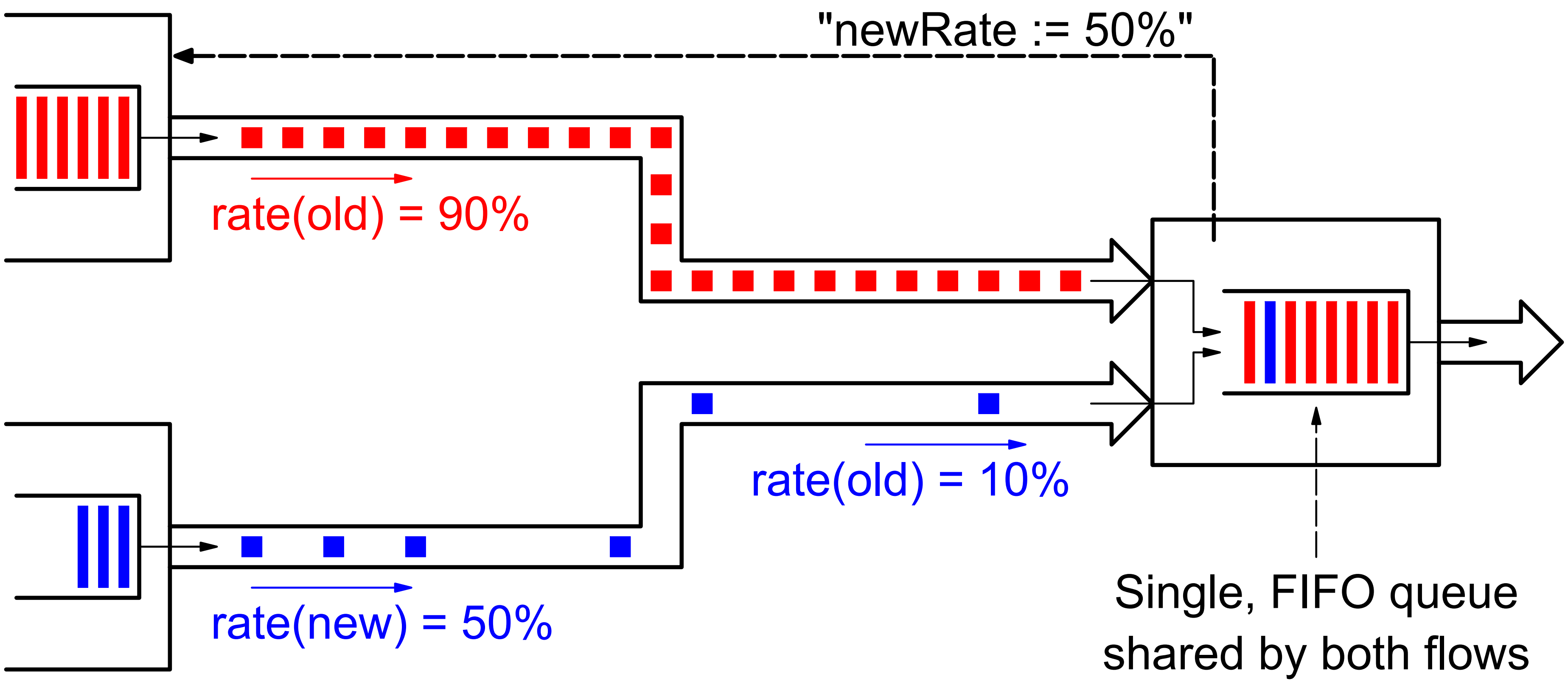
Indiscriminate (FIFO) Queueing is Unfair

(the ``parking lot'' problem)



Solution: ***Per-Flow*** queueing & (weighted) round-robin scheduling

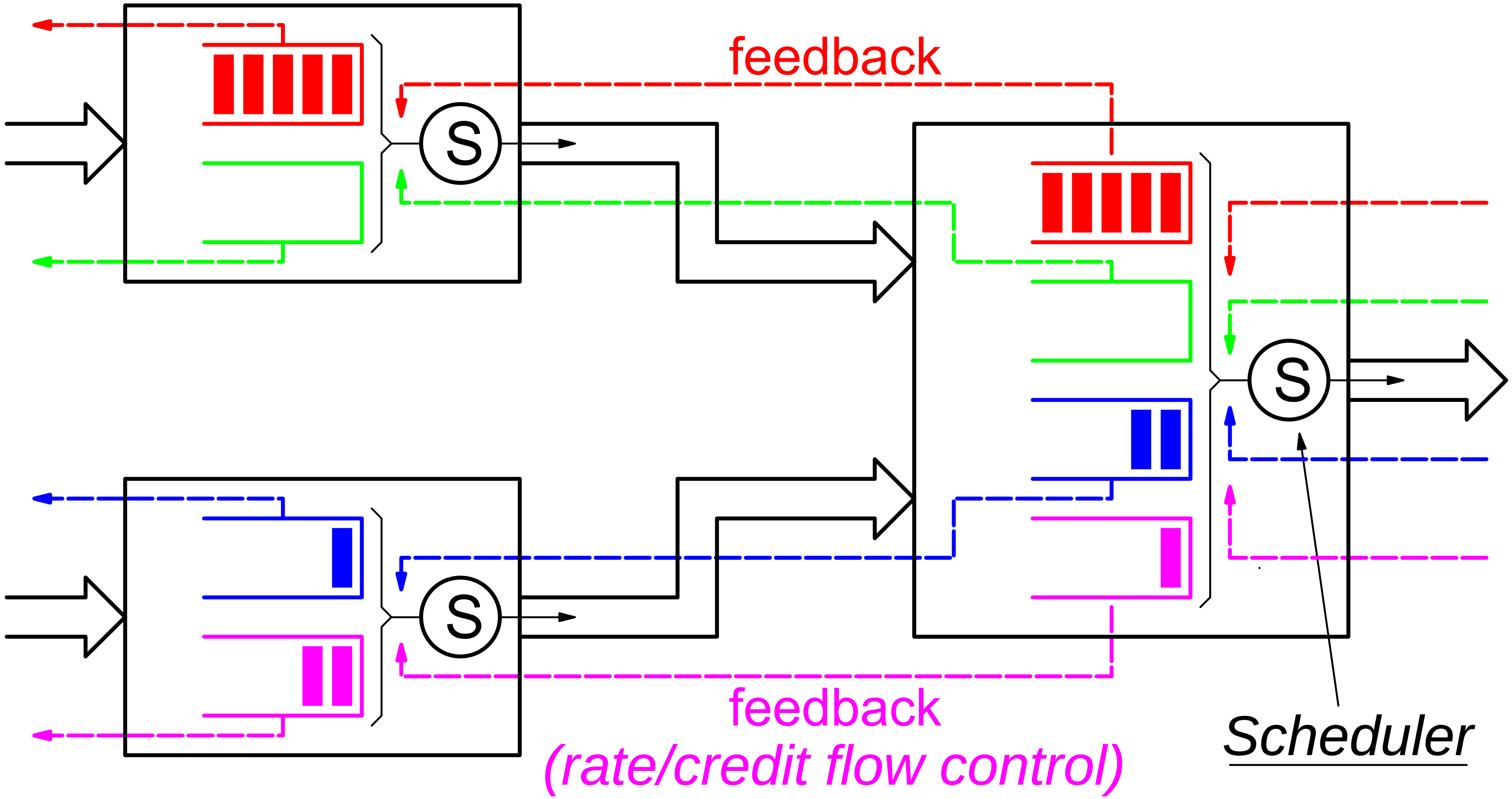
Shared (FIFO) Queueing with Fair (per-flow) Rate Control is slow in enforcing fairness



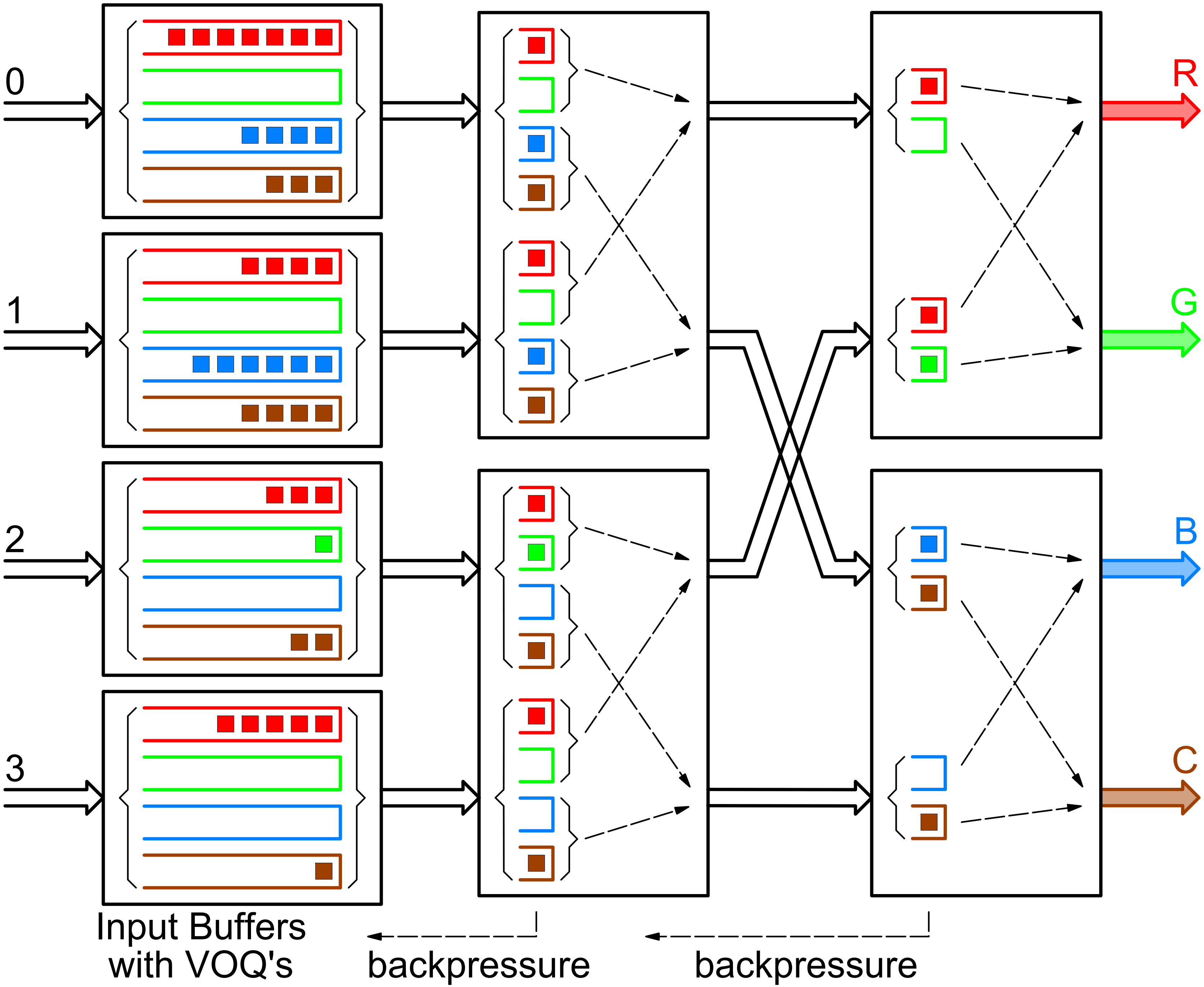
Solution: ***Per-Flow*** queueing

Solution:

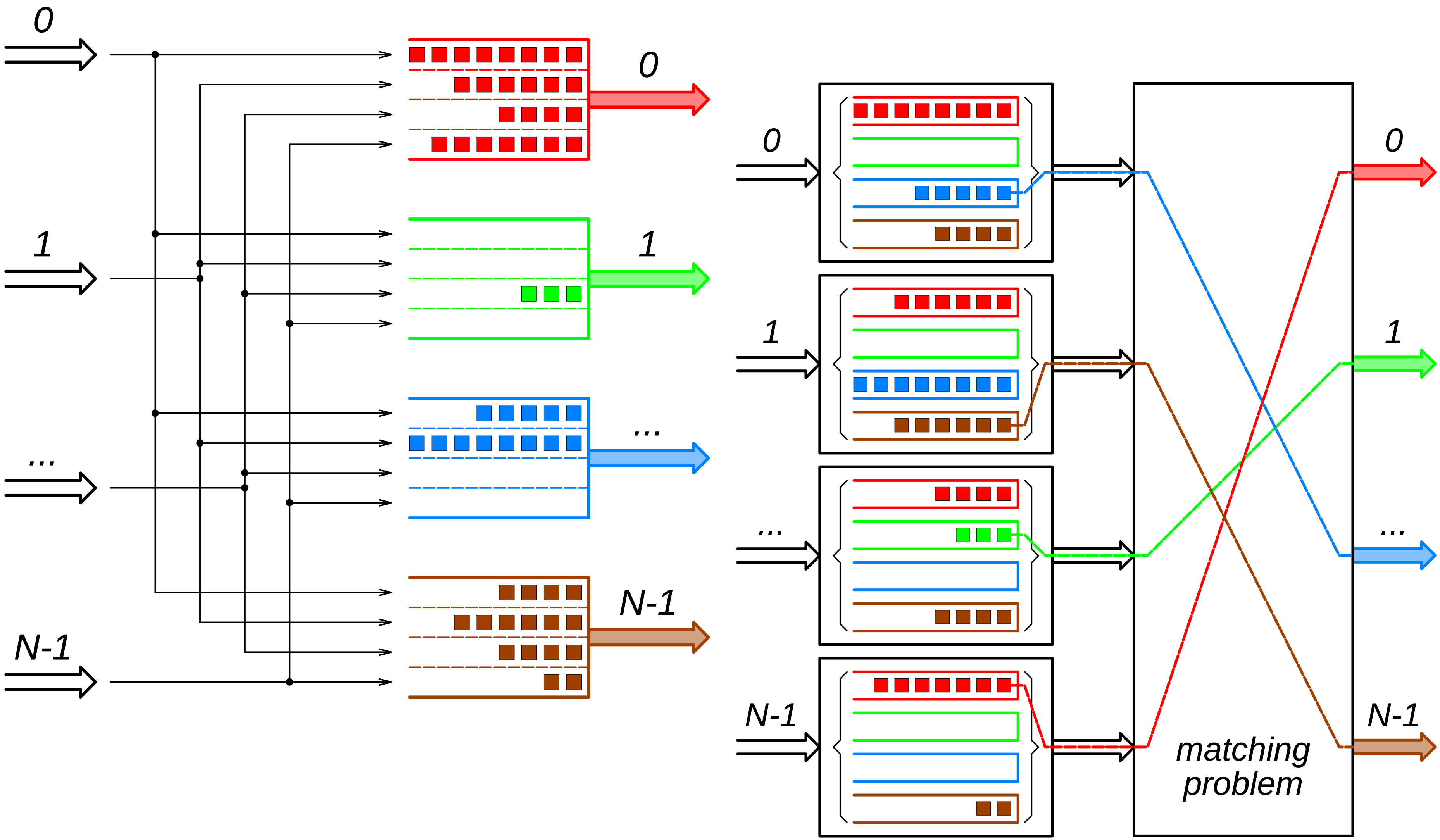
Per-Connection (per-flow) Queueing & Flow Control



(this slide intentionally left blanc)


0
1
2
3
R
G
B
C
Input Buffers
with VOQ's
backpressure
backpressure

Compare to Output Q'ing or VOQs w.Unbuffered Fabric



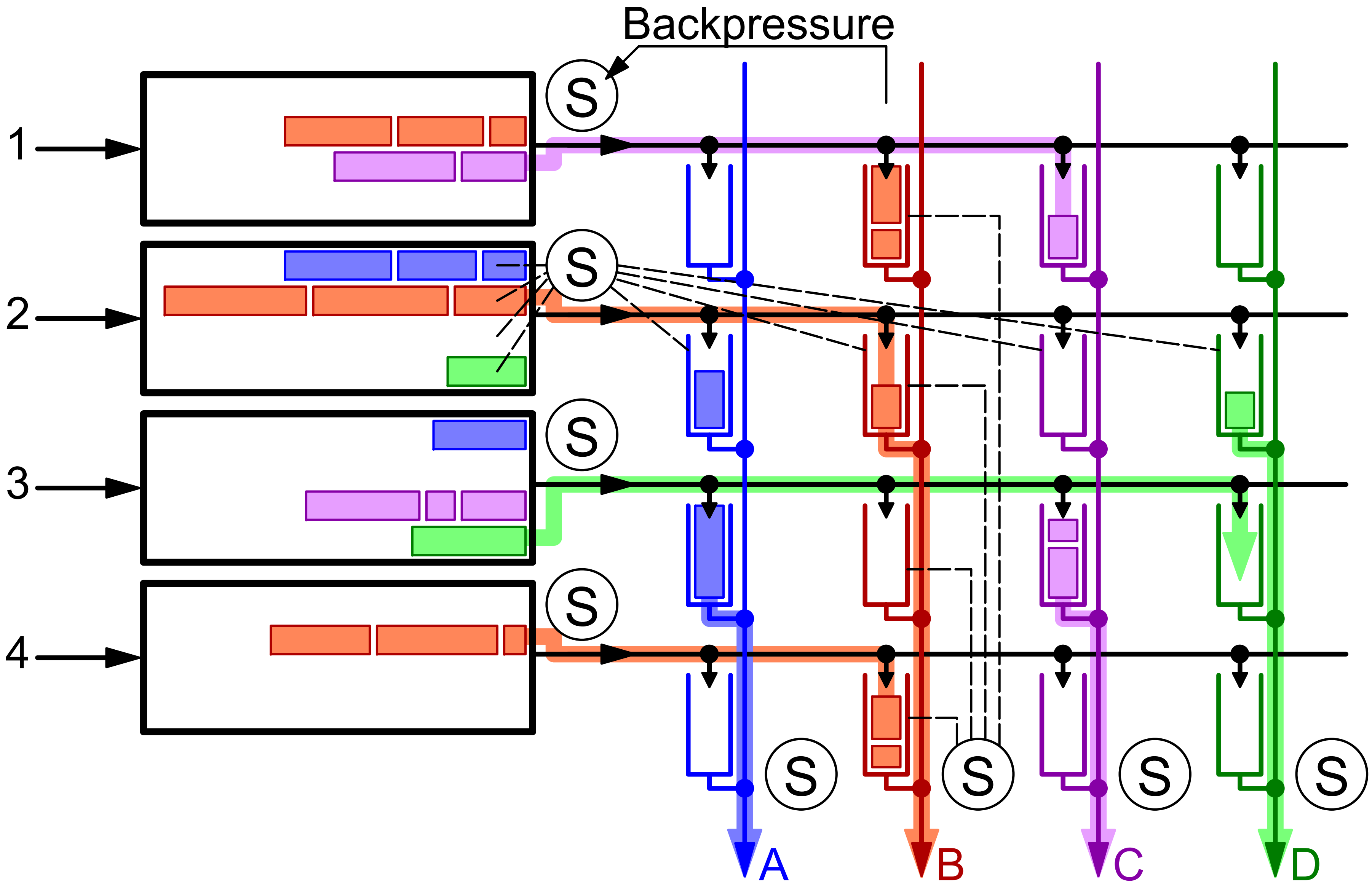
Example Application of per-flow queueing:

Combined Input-Crosspoint Queueing – "CICQ" or "Buffered Crossbars"

- Example application: per-flow queues, per-flow backpressure
  - switching fabric = crossbar
  - flow = input-output pair = crosspoint
  - small buffers inside the fabric → per-crosspoint queues
  - large buffers at the inputs → VOQ's
  - backpressure from the crosspoints to the VOQ's (per-flow) to keep the (small) crosspoint buffers from overflowing
- Loosely-coupled, independent, single-resource schedulers
  - per-output schedulers decide which flow (crosspoint queue) to serve among the non-empty ones in each output's column
  - per-input schedulers decide which flow to serve among the ones with non-empty VOQ and with credits available in each input's row

⇒ Approximate "matchings" yield better scheduling efficiency

Buffered Crossbar (Comb. Input-Crossp. Q'ng – CICQ)



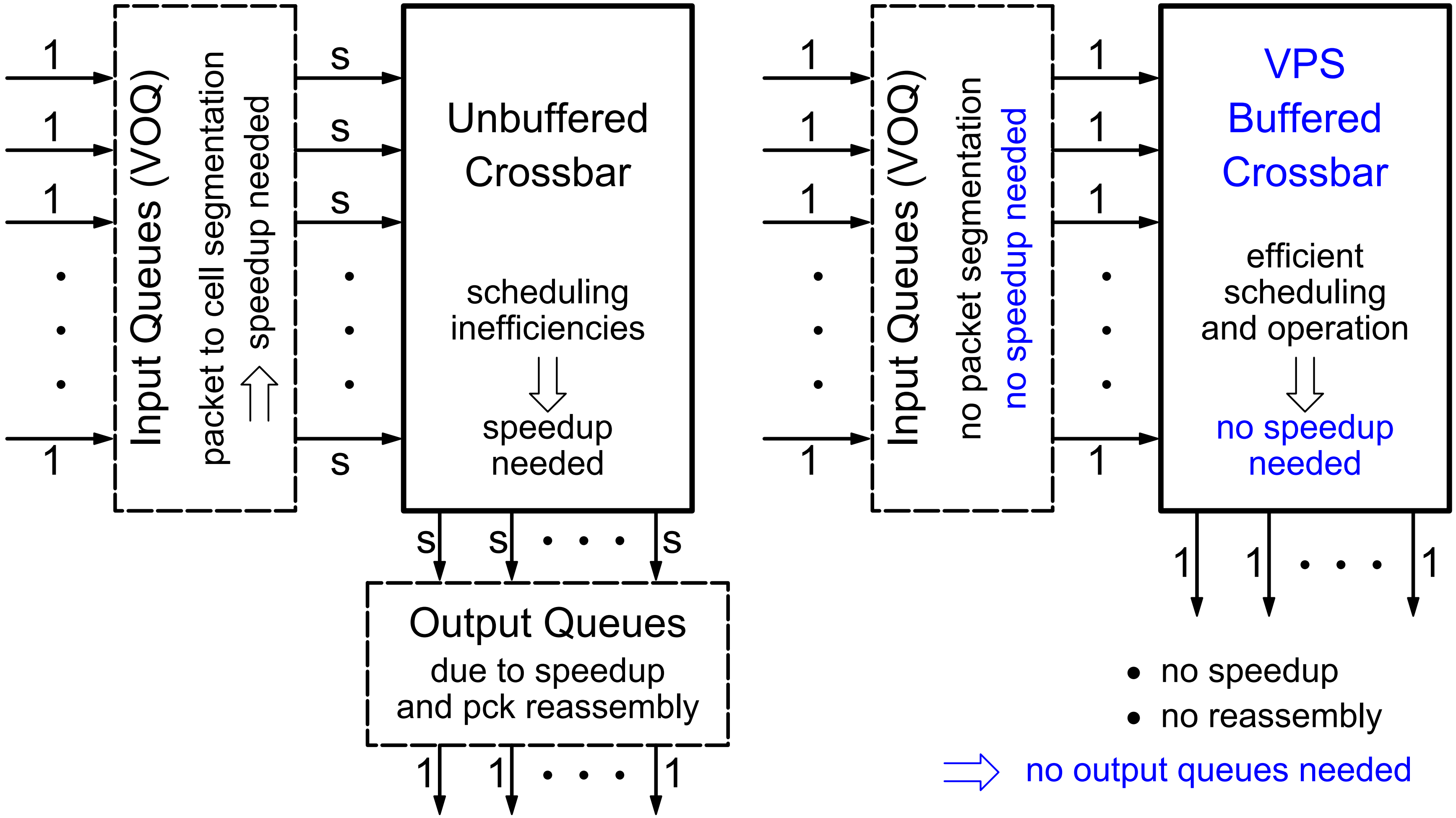
Buffered Crossbars (CICQ) – References:

- D. Stephens, H. Zhang: "Implementing Distributed Packet Fair Queueing in a Scalable Switch Architecture", INFOCOM 1998
- T. Javidi, R. Magill, and T. Hrabik: "A High-Throughput Scheduling Algorithm for a Buffered Crossbar Switch Fabric", ICC 2001
- R. Rojas-Cessa, E. Oki, and H. Jonathan Chao: "CIXOB-k: Combined Input-Crosspoint-Output Buffered Switch", GLOBECOM 2001
- Abel, Minkenberg, Luijten, Gusat, Iliadis: "A Four-Terabit Packet Switch Supporting Long Round-Trip Times", IEEE Micro, Jan. 2003
- N. Chrysos, M. Katevenis: "Weighted Fairness in Buffered Crossbar Scheduling", IEEE Wrksh. High Perf. Switching & Routing (HPSR) 2003

⇒ M. Katevenis, G. Passas, D. Simos, I. Papaefstathiou, N. Chrysos: "Variable Packet Size Buffered Crossbar (CICQ) Switches", ICC 2004

- G. Passas, M. Katevenis: "Packet Mode Scheduling in Buffered Crossbar (CICQ) Switches", IEEE W.High Perf.Sw.Rtng (HPSR) 2006

Variable Packet Size (VPS) Buffered Crossbars



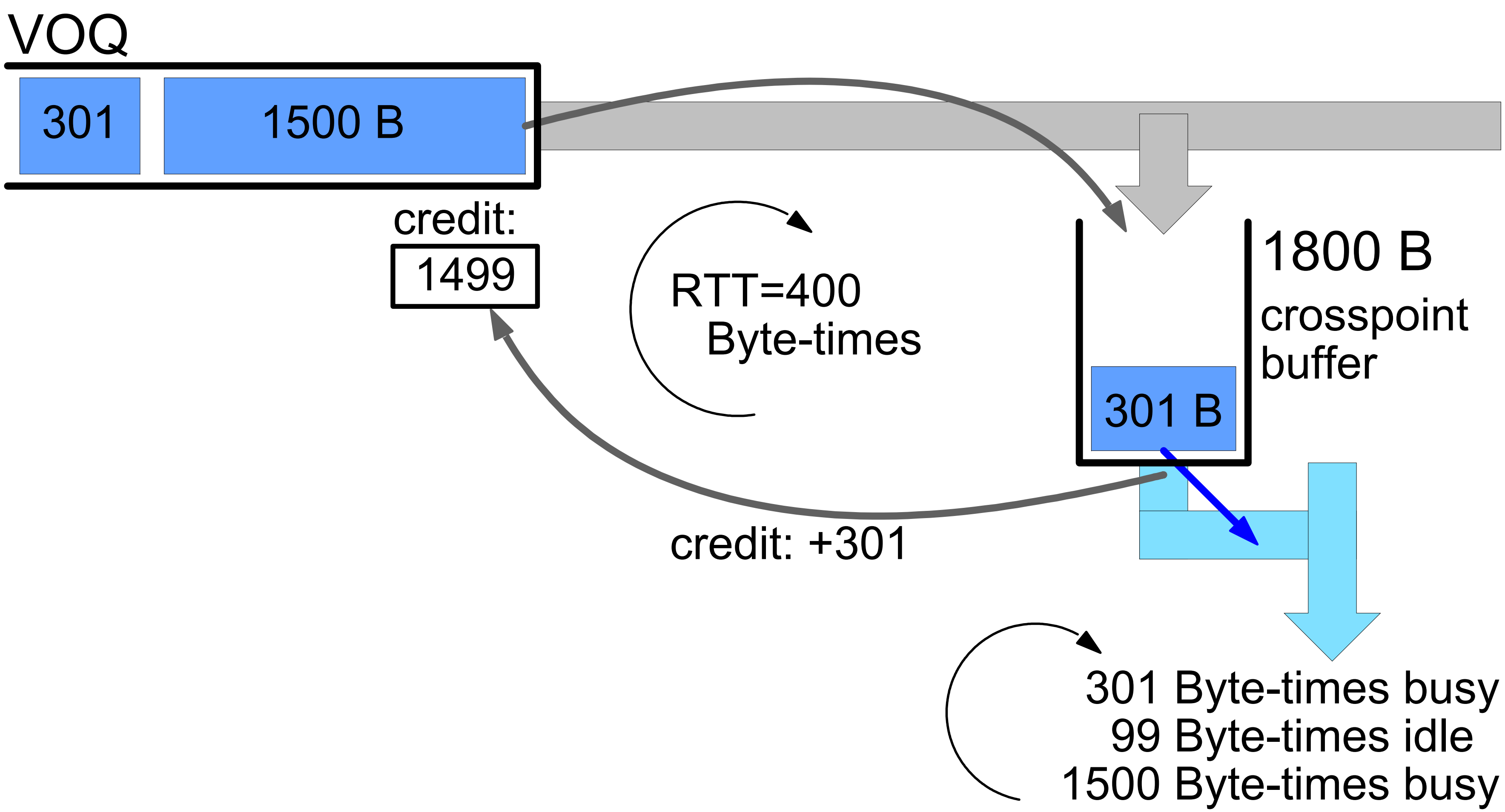
- s = 2 to 3 approximately
- assuming same core speed

⇒ Buffered crossbar yields 2 to 3 times faster ports (and cut-through), at lower cost (no output buffers, except when output sub-ports needed)

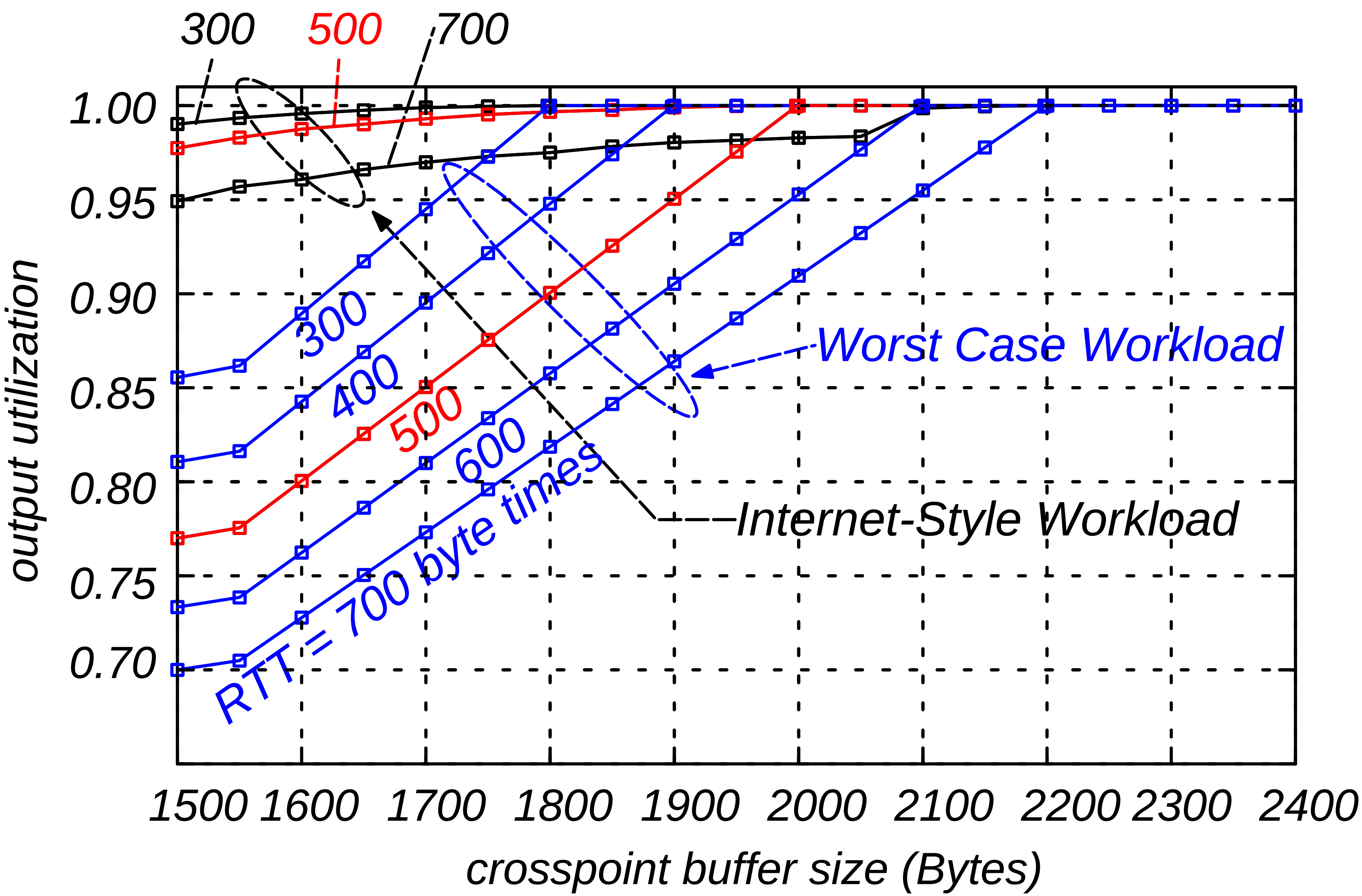
Crosspoint Buffer Sizing for Variable-Size Packets

- For full throughput under worst-case single active flow:

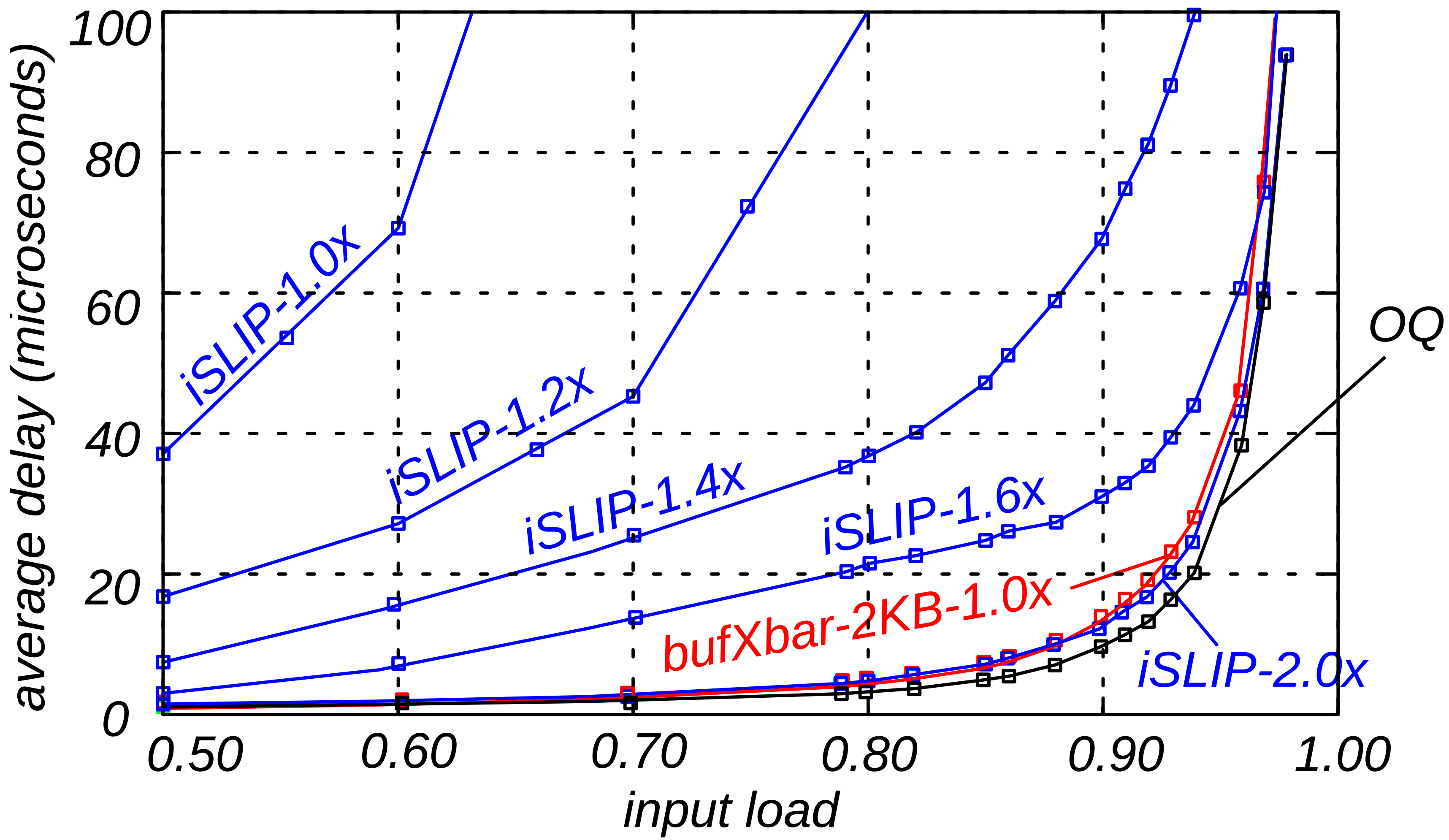
$$\text{CrosspBufSize} \geq \text{MaxPacketSize} + \text{RTTwindow}$$



Crosspoint Buffer ≥ MaxPckSize + RTTwindow

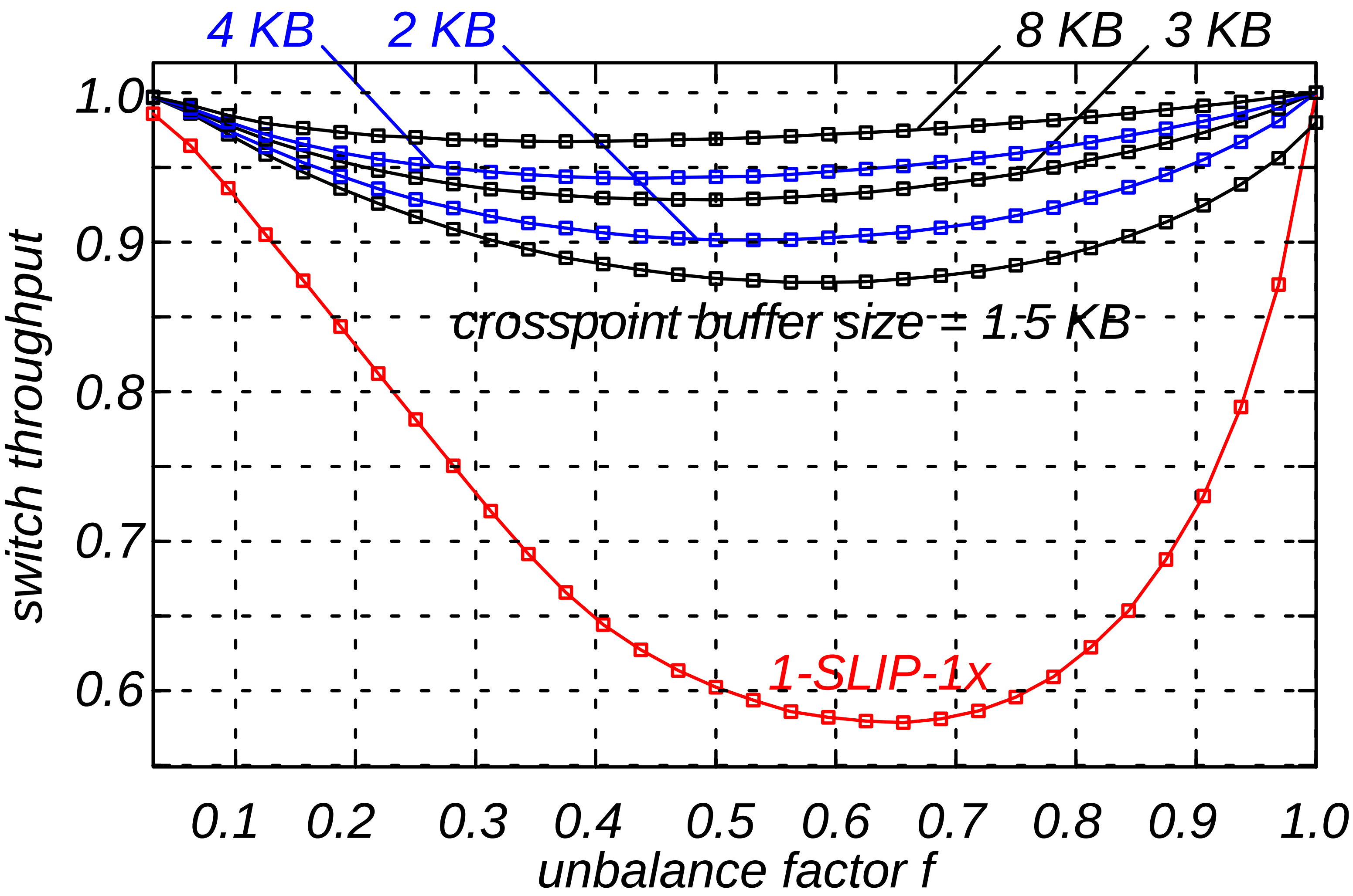


No Speedup needed to approach Output Queuing



- Uniform destinations
- Internet-style synthetic workload; 40-1500 byte packet sizes
- Unbuffered crossbar w. SAR: one-iteration iSLIP, 64-byte segments

Saturation Throughput under Unbalanced Traffic



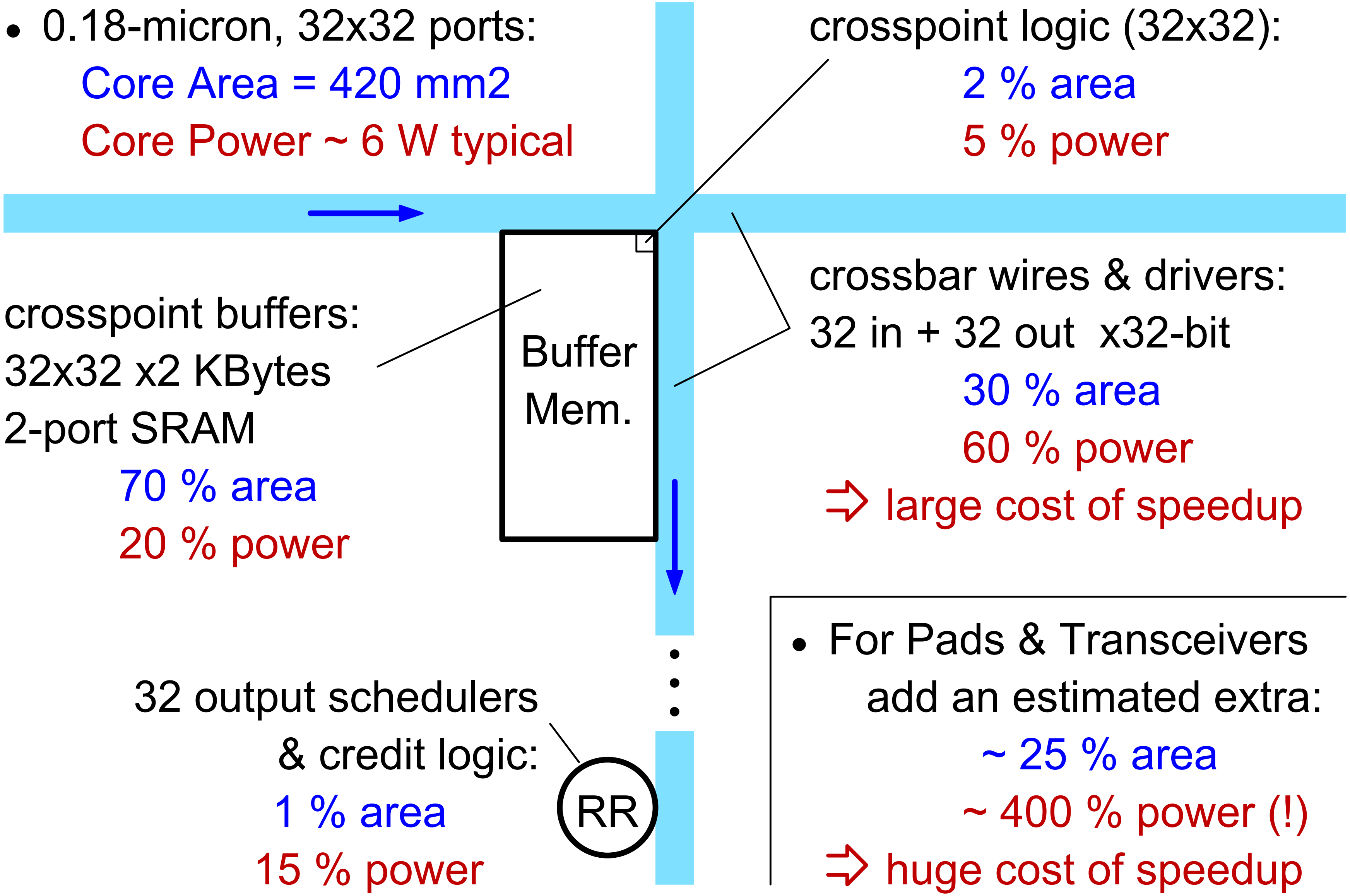
- Poisson arrivals, Pareto sizes (40-1500)
- For iSLIP, packet sizes are multiples of 64 B (⇒ no SAR overhead)

A VPS Buffered Crossbar Chip Design

- 32x32 ports, 300 Gbps aggregate throughput
- 2 KBytes / crosspoint buffer × 1024 crosspoints
- Variable-size packets (multiples of 4 Bytes)
- 32-bit datapaths
- Cut-through at the crosspoints
- Fully designed, in Verilog
- Core only, no pads & transceivers
- Fully verified: Verilog versus C++ performance simulator
- Crosspoint logic = 100 FF + 25 gates (simplicity!)
- Synthesized: Synopsys
- Placed & routed: Cadence Encounter, 0.18 μm UMC
  - Clock frequency: 300 MHz @ 0.18 μm (operates at maximum SRAM clock frequency)

Core Area, Power Allocation:

- 0.18-micron, 32x32 ports:
  - Core Area = 420 mm2
  - Core Power ~ 6 W typical



crosspoint logic (32x32):
- 2 % area
- 5 % power

crosspoint buffers:
32x32 x2 KBytes
2-port SRAM
- 70 % area
- 20 % power

crossbar wires & drivers:
32 in + 32 out x32-bit
- 30 % area
- 60 % power
- ⇨ large cost of speedup

32 output schedulers
& credit logic:
- 1 % area
- 15 % power

- For Pads & Transceivers add an estimated extra:
  - ~ 25 % area
  - ~ 400 % power (!)
  - ⇨ huge cost of speedup