

Queue and Flow Control Architectures for Interconnection Switches

1. Basic Concepts and Queueing Architectures
2. High-Throughput Multi-Queue Memories
3. Crossbars, Scheduling, & Combination Queueing
4. Flow and Congestion Control in Switching Fabrics

Manolis Katevenis

FORTH and Univ. of Crete, Greece

<http://archvlsi.ics.forth.gr/~kateveni/534>

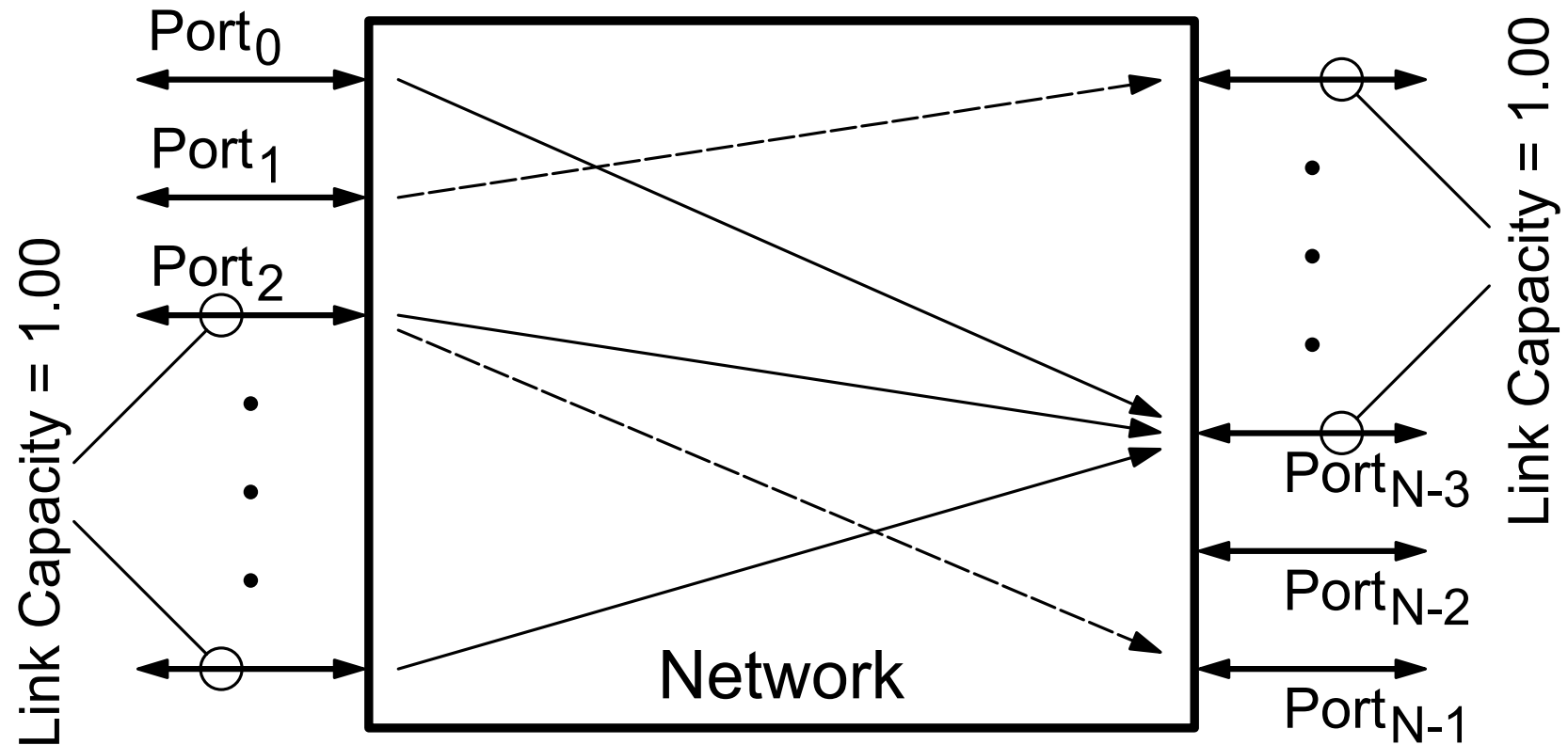
1. Basic Concepts and Queueing Architectures

Table of Contents:

- **1.1 Problem Statement**
 - scalable, distributed, multi-party communication
- **1.2 Basic Concepts and Terminology**
 - feasible traffic, internal blocking
 - output contention, buffering, flow control, admission control
 - circuit versus packet switching
 - time vs. space switching, multiplexor, crossbar, buffer memories
- **1.3 Queueing Architectures – Family 1:**
 - shared buffer, output queueing, crosspoint queueing
- **1.4 Queueing Architectures – Family 2:**
 - input queueing: head-of-line blocking, per-output queues

1.1 Problem Statement

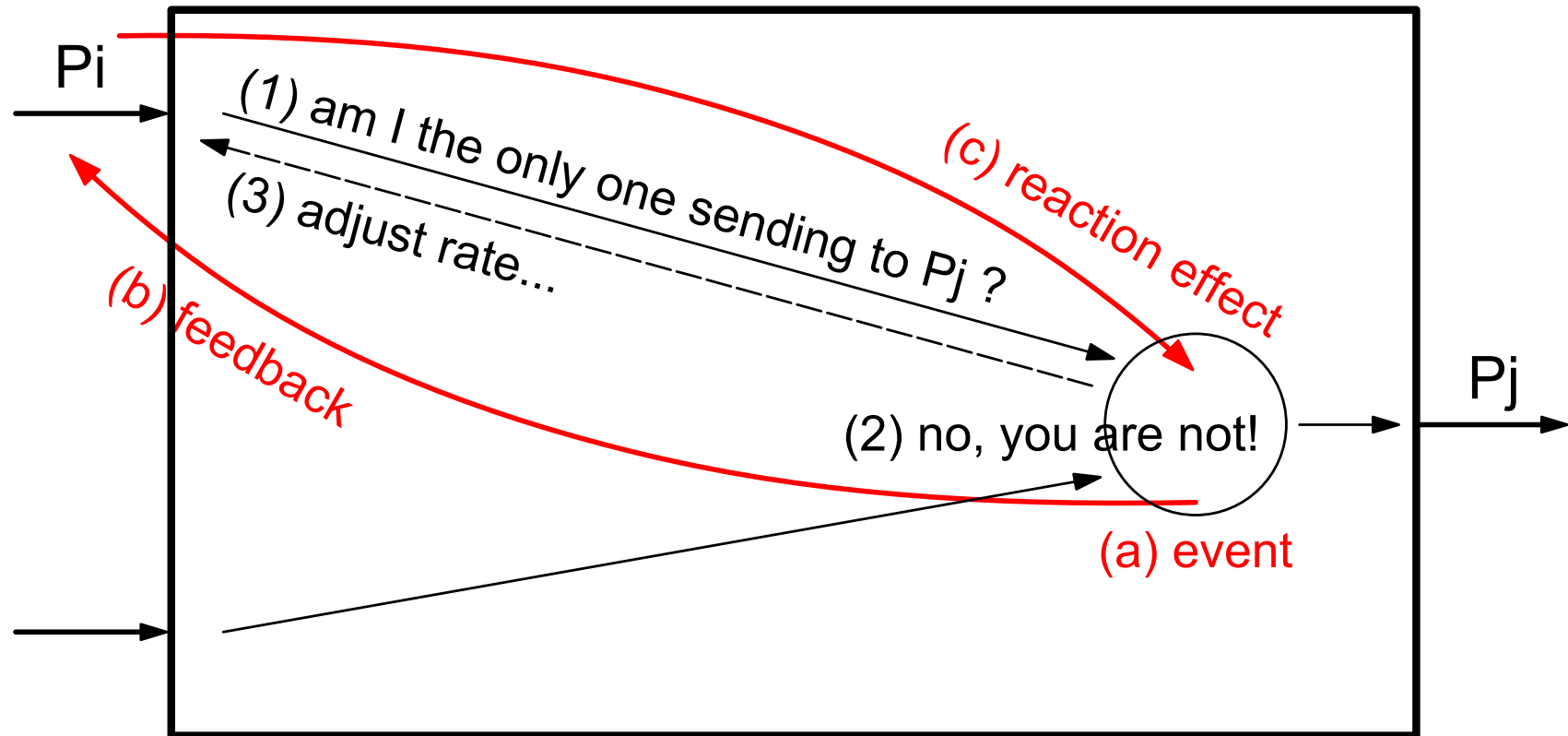
Scalable, Distributed, **Multi-party** Communication



Output Contention: \sum incoming rates $>$ outgoing link capacity

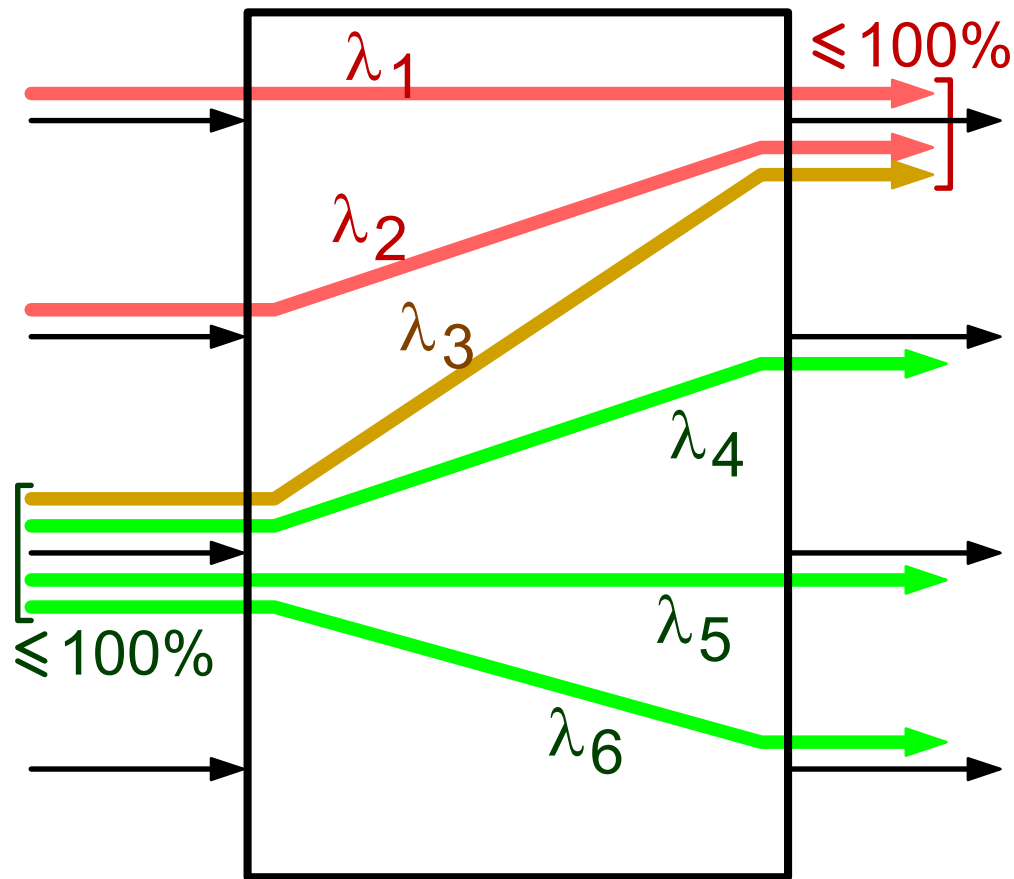
1.1 Problem Statement

Scalable, Distributed, Multi-party Communication



Reaction Effect Delay = Round-Trip Time (**RTT**)

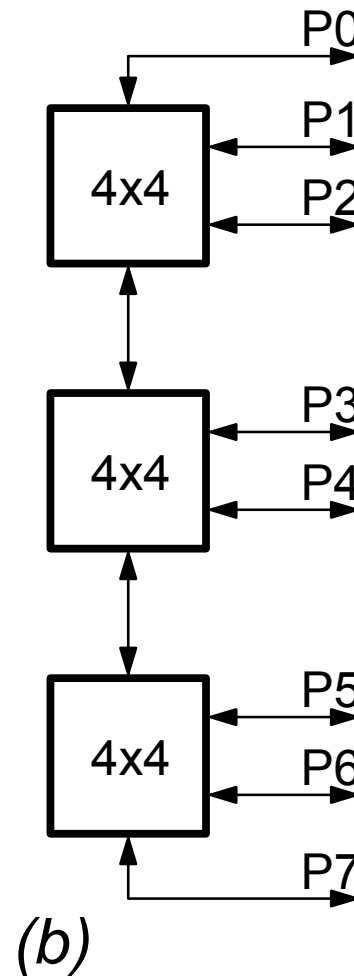
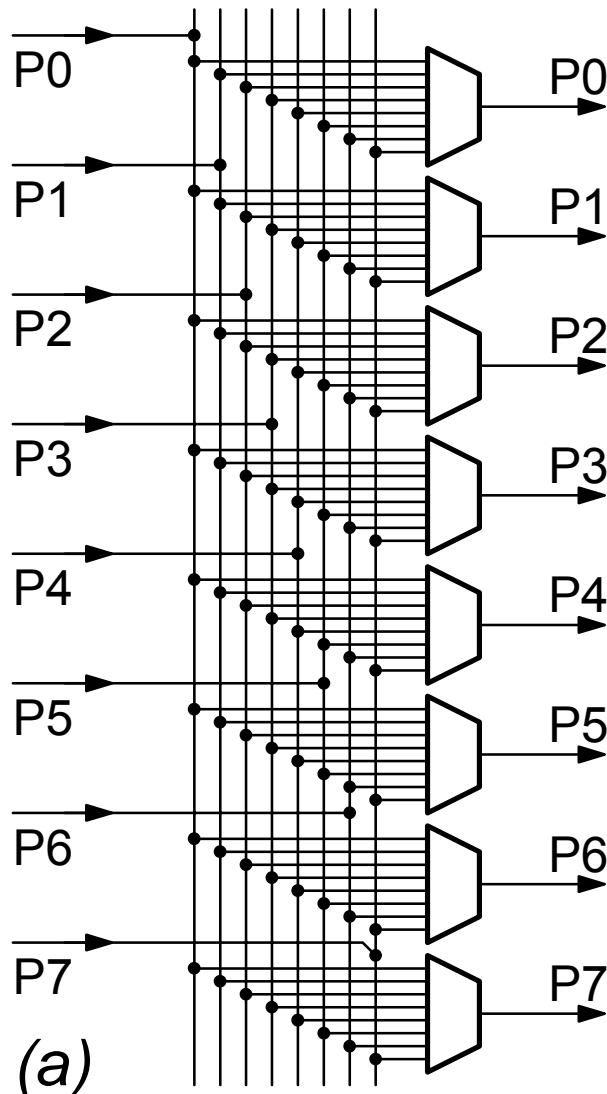
Example: Interdependent Constraints



- $\lambda_1 + \lambda_2 + \lambda_3 \leq 100\%$
(output contention) \Rightarrow
 $33\% + 33\% + 33\%$? (fairness)
- $\lambda_3 + \lambda_4 + \lambda_5 + \lambda_6 \leq 100\%$
(input rate limitation) \Rightarrow
 $25\% + 25\% + 25\% + 25\%$?
- $\lambda_3 = 25\% \Rightarrow \lambda_1 + \lambda_2 = 75\% \Rightarrow$
 $\lambda_1 = \lambda_2 = 37.5\%$
(max-min fairness) ?
- or $\lambda_3 = 0$, $\lambda_1 = \lambda_2 = 50\%$,
 $\lambda_4 = \lambda_5 = \lambda_6 = 33.3\%$
(maximum utilization) ?

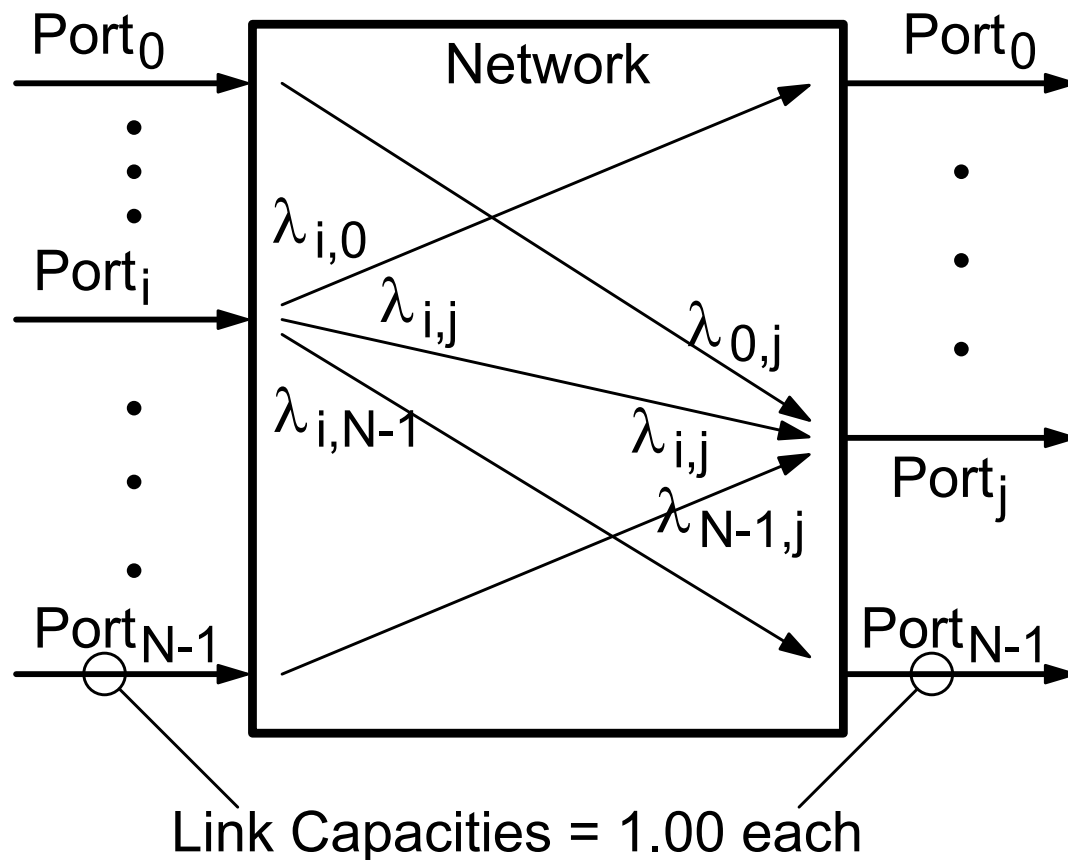
1.1 Problem Statement

Scalable, Distributed, Multi-party Communication



- (a) all-to-all connectivity costs $O(N^2)$
- (b) lower-cost scalability is feasible – with or without compromise on performance?

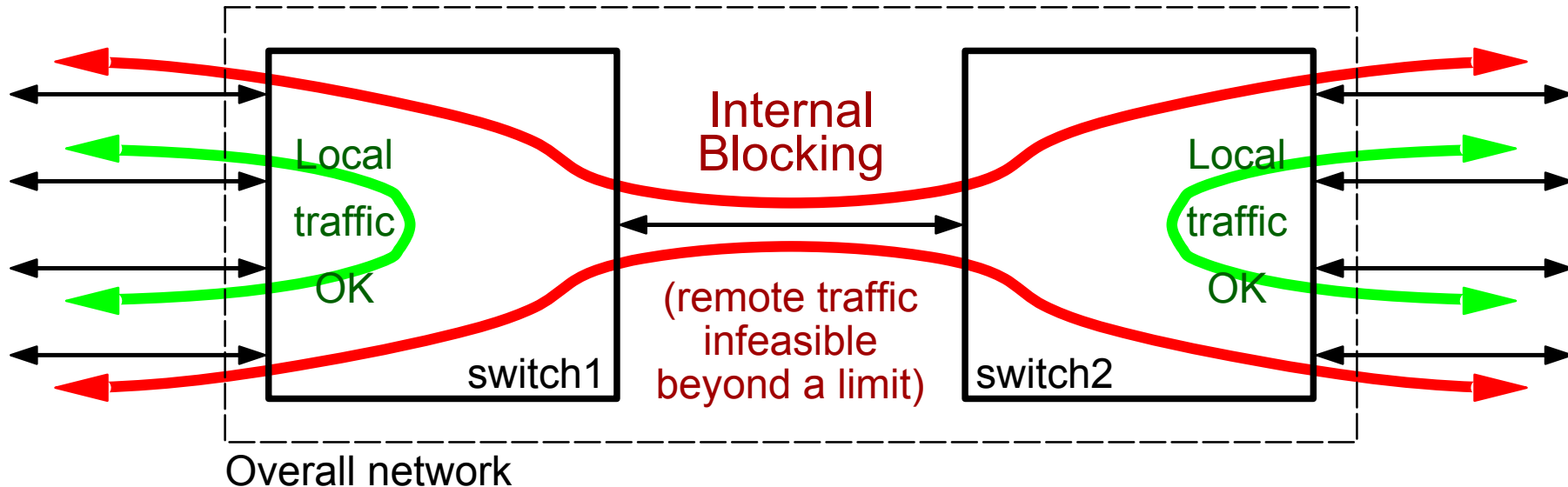
1.2 Basic Concepts & Terminology: Feasible Traffic



- Rates satisfying:
- $\forall i: \sum_j \lambda_{i,j} \leq 1$, i.e. do not violate input link capacities; and:
- $\forall j: \sum_i \lambda_{i,j} \leq 1$, i.e. do not violate output link capacities – do not create “*output contention*”.

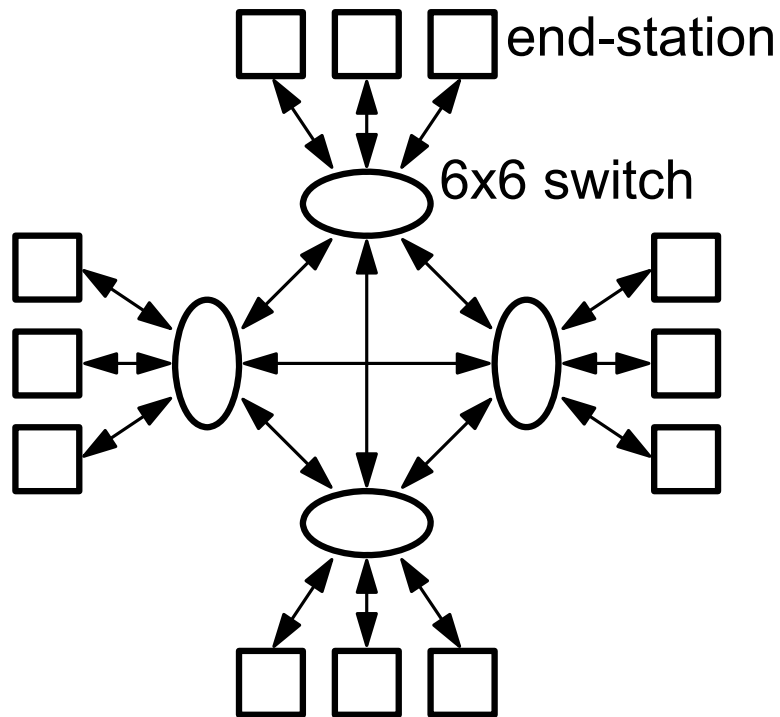
- Flow control & congestion management strive to determine and enforce feasible rates at the sources – not at all an easy problem...

1.2 Basic Concepts & Terminology: Internal Blocking



- Externally feasible traffic for overall network, but...
- Internally *not* feasible, due to internal link oversubscription.
- Int. blocking in a network is output contention in a subnetwork, but:
- Output contention is the customer's responsibility, while...
- Internal blocking is the network provider's responsibility.

Internal Blocking: a quiz and a preview of a key Result



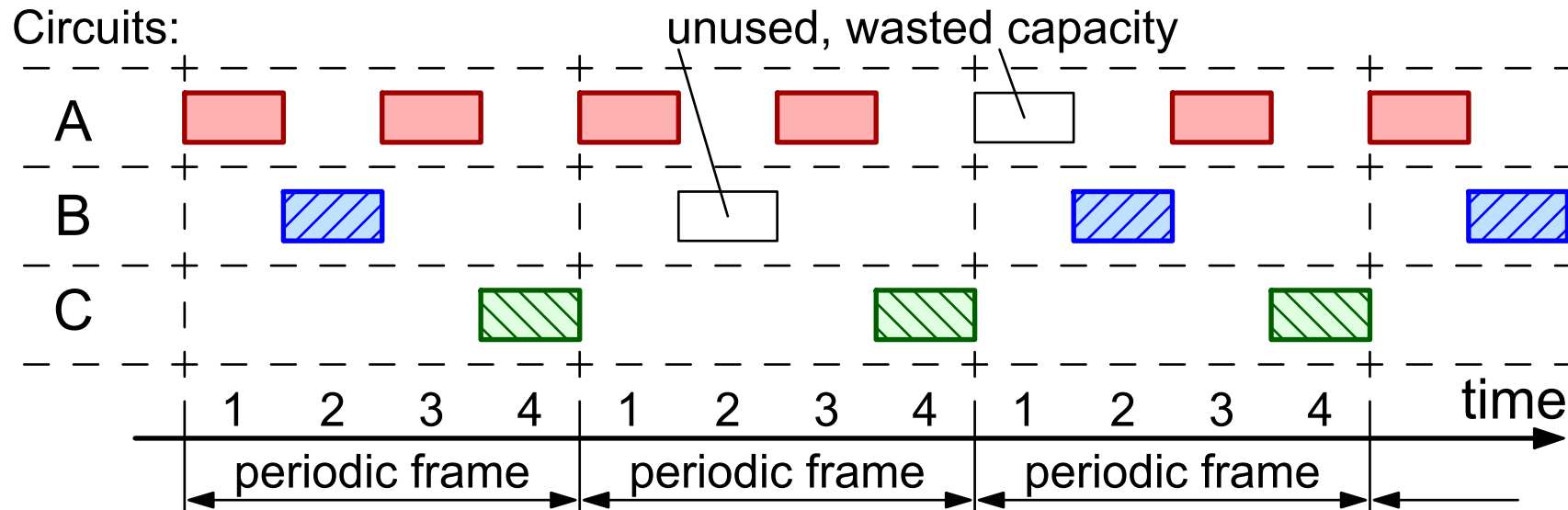
- does this network have internal blocking?
- (a)... if each flow is always routed through a fixed path?
- (b)... if routing paths can adapt to traffic patterns?
- hint: consider the traffic crossing a 45° bisection ("bisectional throughput")

Key Result (see later): A $N \times N$ network made of $(N/2) \cdot \log_2 N$ or less 2×2 switch elements will *always* have internal blocking. The Benes network, using multipath routing, has $\approx N \cdot \log_2 N$ 2×2 switch elements and is internally non-blocking.

Dealing with **Contention** for Link Throughput

- *Option 1*: Ensure contention **never appears**
 - preschedule everything – fixed-rate traffic – “*circuit switching*”
- *Option 2*: Allow dynamically varying rates – “*packet switching*”
- 2(a): Dealing with **Short-term** contention
 - ⇒ manageable volume of excess traffic ⇒ either:
 - buffer excess packets, temporarily, in memories, or:
 - drop excess packets – and possibly retransmit later
 - OK in some applications, and if we ensure it rarely happens, e.g. if it only happens on memory overflow, or w. massive overspeed
- 2(b): Dealing with **Long-term** contention
 - ⇒ unmanageable volume if excess traffic allowed to persist
 - either, beforehand, use admission control
 - increased latency before traffic allowed to start or change rate
 - or, after-the-fact, use flow control – congestion management
 - need large (RTT) buffer space(s) and multiple queues

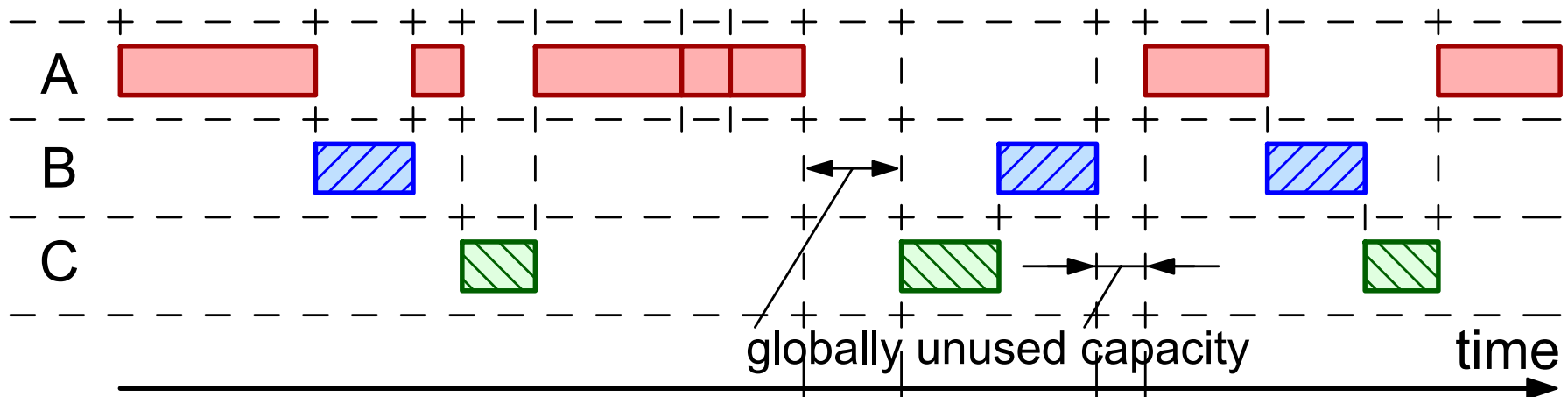
1.2 Basic Concepts & Terminology: Circuit Switching



- Originates from telephonic circuits, digitized and time-multiplexed
- Fixed-rate, prescheduled at connection set-up time – like trains
- Data-only – no headers needed: time-slot position in frame implicitly provides circuit ID (flow ID) and routing information
- + Simplicity: *static, off-line* routing decisions and contention resolution
- Partitioned Capacity: throughput is statically partitioned among circuits: unused capacity in one circuit is wasted – cannot be used by other ckts

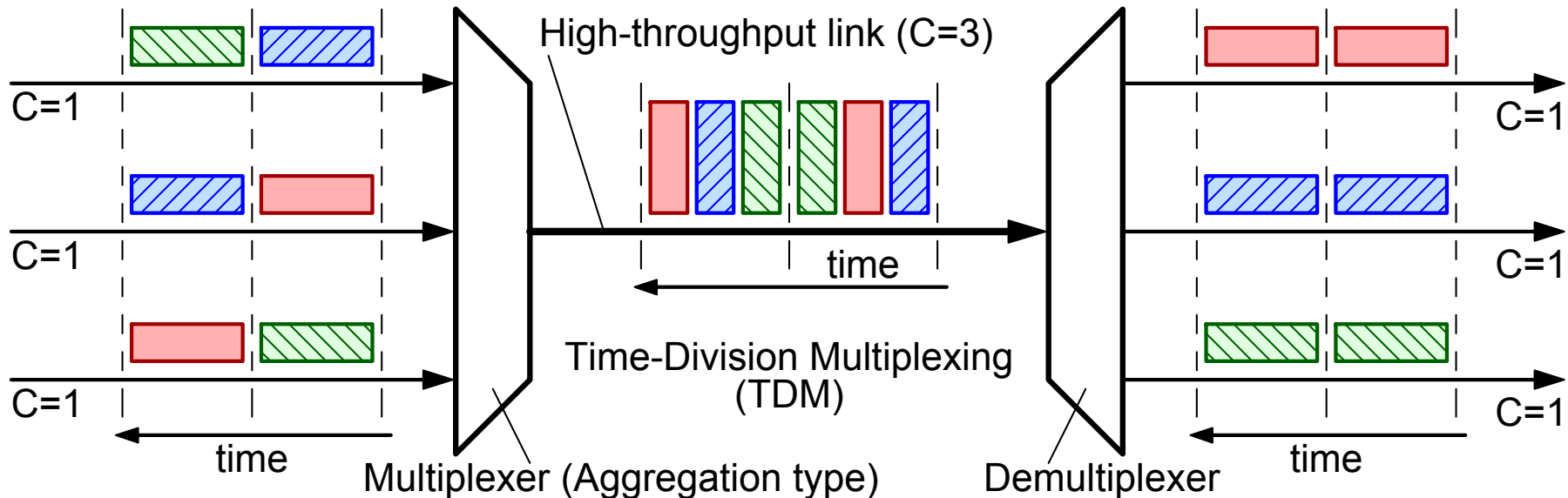
1.2 Basic Concepts & Terminology: Packet Switching

Flows:



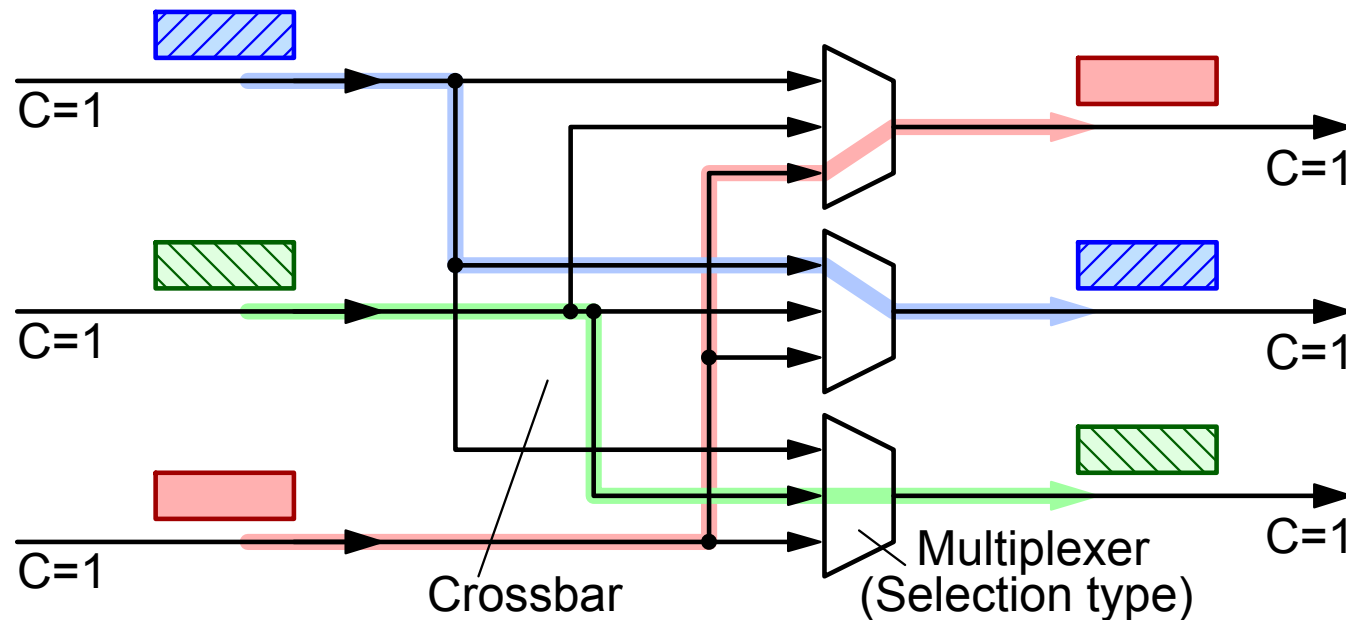
- Varying or unpredictable traffic – like automobiles
- Self-describing packets: header provides destination address
- + Transmission capacity of link is dynamically shared among flows
- Demanding: *dynamic, on-line, run-time* routing decisions and contention resolution

1.2 Basic Concepts & Terminology: Time Switching



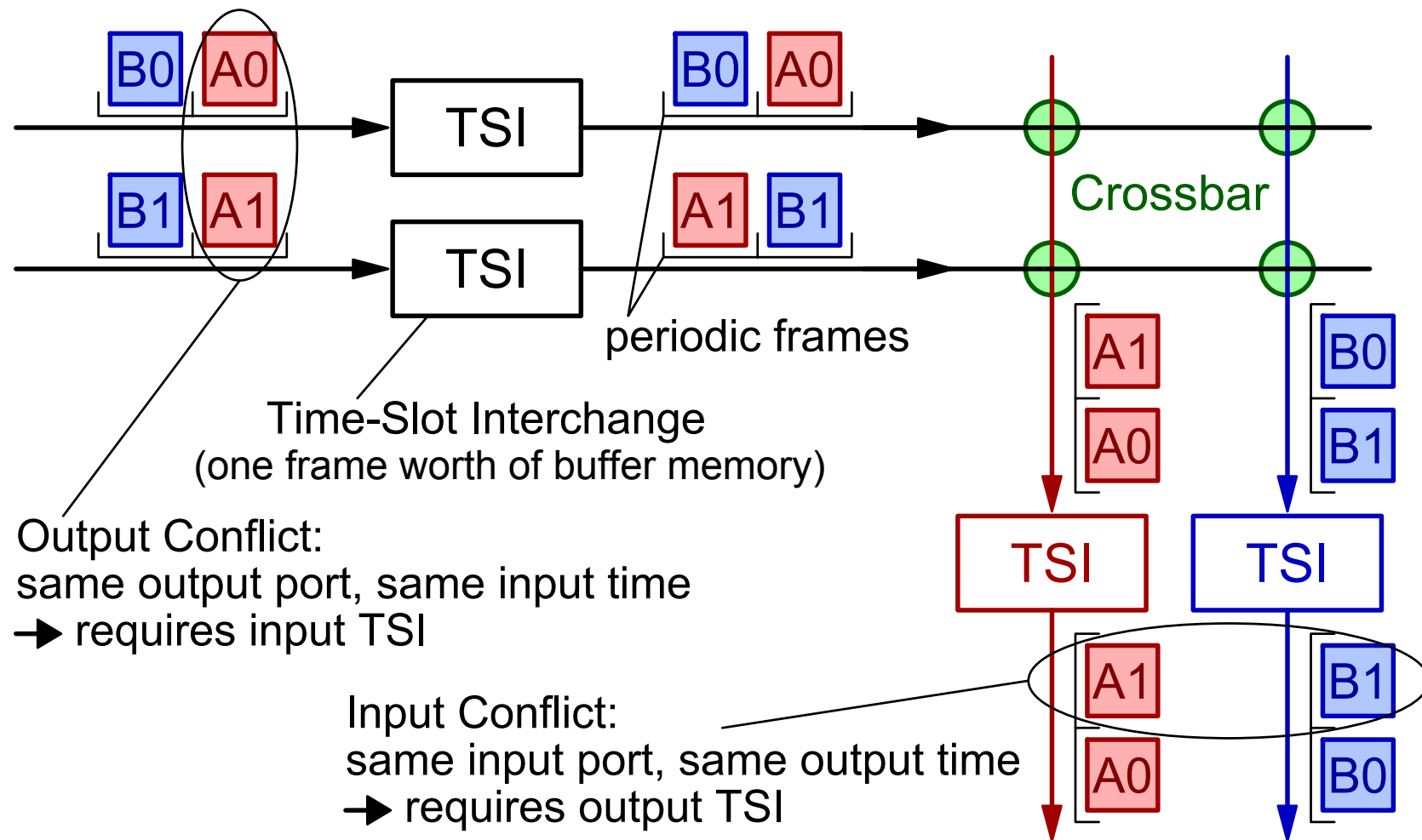
- All packets pass through a single point in space, at different times
- Similar to time-sharing – multiprogramming on a single processor
- Buses are in this category (distributed multiplexor, built w. tristate drivers)
- + Economize on datapaths, wires, memories
- + Easy to share aggregate capacity among competing flows
- Non-scalable: infeasible beyond technology limit for aggregate capacity

1.2 Basic Concepts & Terminology: Space Switching



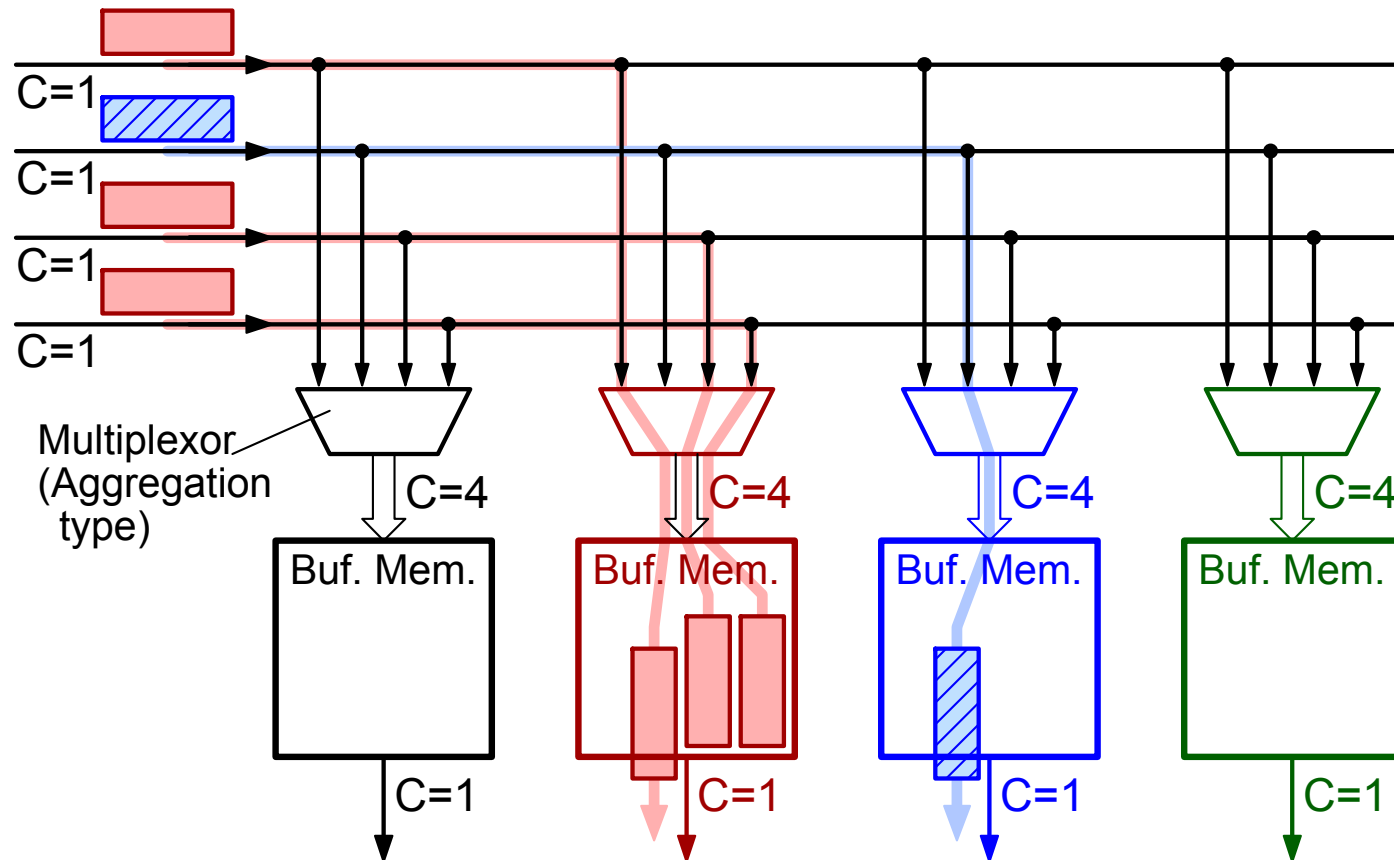
- Packets at a given time pass through different paths in space
- Similar to multiprocessing on parallel processors
- Crossbars are in this category (single-stage space switches)
- + Scalable: use when aggregate throughput > upper limit of time switching
- Partitioned memories, wires \Rightarrow harder to route, schedule, load balance

Combination Example: Time-Space-Time Circuit Switch



- Time switching (TSI's) needed to resolve output and input conflicts

Contention Resolution: Buffer Memories



- Memories are needed to temporarily buffer packets that cannot proceed due to (hopefully short-term) output contention

1.3 Queueing Architectures – family 1

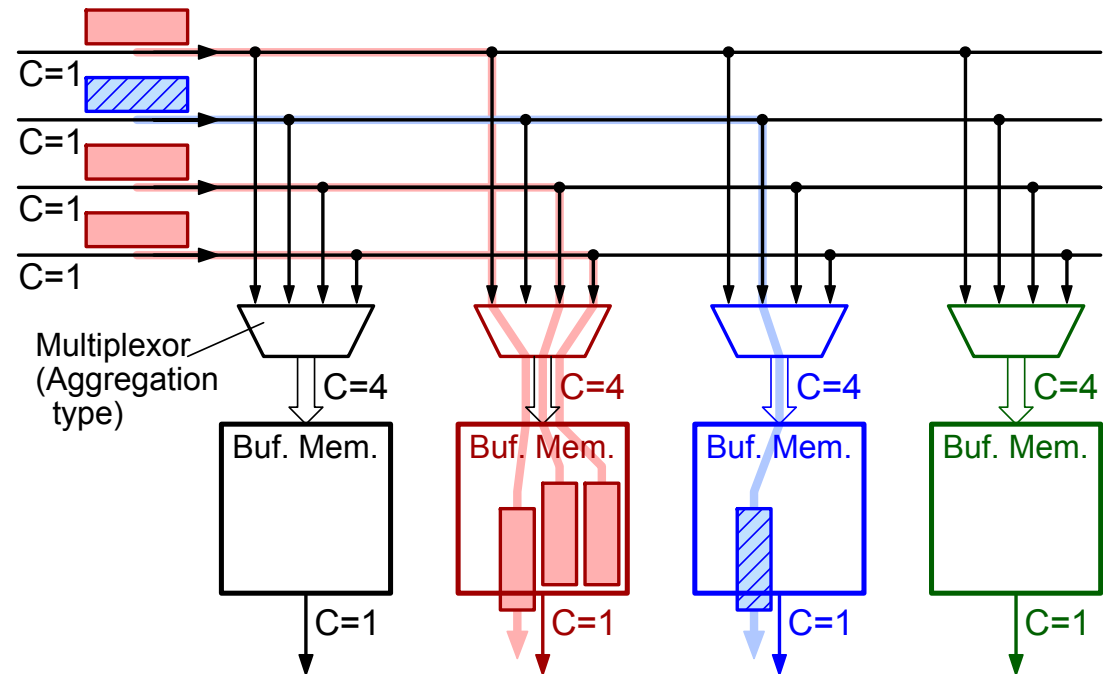
Output Queueing – the “reference” architecture

- Packets are buffered in per-output memories, right next to their desired output

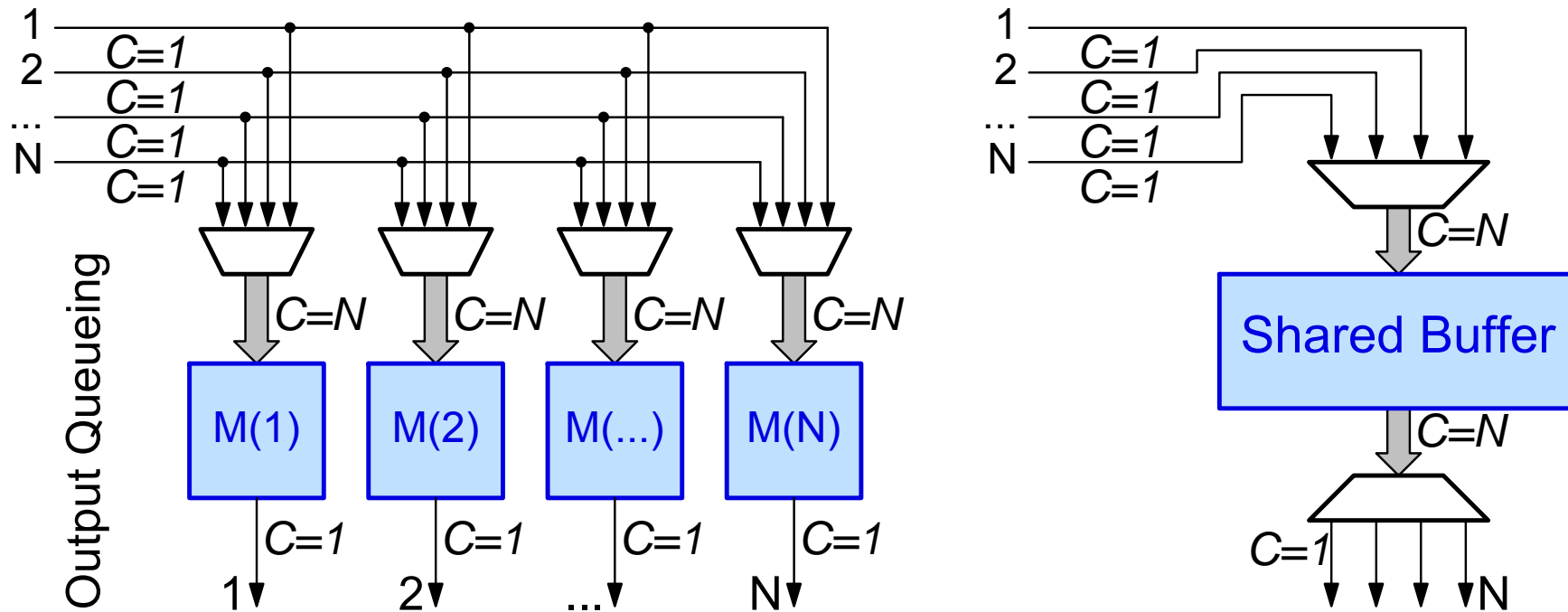
⇒ “Work Conserving”

Operation: an output will never remain idle while even a single packet destined to it exists in some switch buffer memory ⇒

- + Minimum possible delay
- + Full (100%) utilization of outgoing link capacity
- + Adaptable to any quality-of-service (QoS) policy: organize queues and scheduler as desired within each per-output buffer; but...
- Wasteful in buffer-memory throughput – see shared buffer arch, below
- Partitioned buffer space is less efficient than shared – see below...



Shared Buffer – the “best” architecture (when feasible)

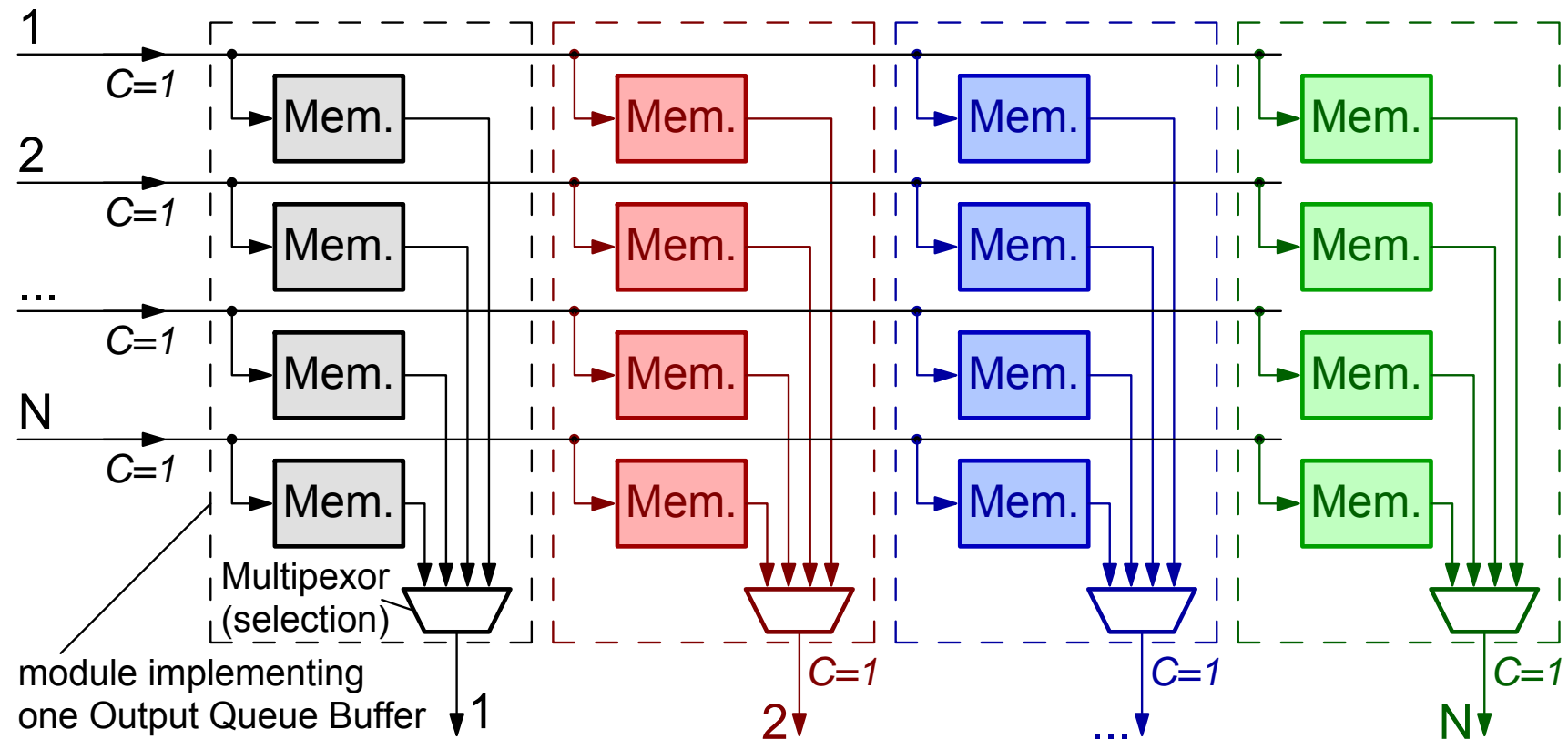


- + Aggregate memory throughput = $2 \cdot N$, versus $N \cdot (N+1) = N^2 + N$ for Outp.Q
- + Same high performance & minimum delay as Outp.Q with proper data str.
- + Shared buffer space is more efficiently used than partitioned in Outp.Q'ng
- Non-scalable: requires building a buffer memory with throughput = $2 \cdot N$ (Outp.Q'ng is not scalable either: requires mem's of thruput $(N+1)$ each)

Memory Throughput determines Feasibility & Cost

- Is memory throughput arbitrarily scalable by increasing its width?
 - Not for memory widths exceeding the packet size!
 - Multi-packet-wide memories are in reality full-fledged switches
 - Example: Internet traffic consists of $\sim 60\%$ min-size packets, of size 40 Bytes (320 bits) each; assume mem. cycle time = 2 ns
 \Rightarrow peak memory throughput = 500 Maccesses/s = 500 Mpackets/s
 $= 500 \text{ M} \times 320 \text{ bits/s} = 160 \text{ Gbps}$; for 10 Gbps links, this allows the shared buffer arch. to scale only up to $2 \cdot N = 16 \Rightarrow 8 \times 8$ switch
- On-chip memory: power consumption \sim throughput
 - e.g. 130 nm $\Rightarrow \approx 1.5$ to 2 mW / Gbps (for small mem. blocks: consumption dominated by sense amp's; large blocks: by size)
- Off-chip memory: wire, pin, and chip count \sim throughput
 - RAM chip address and data throughput ≈ 500 to 800 Mbps/pin
 - pin & wire count determine pkg size, board area, power cons.

Crosspoint Queueing – *scalable but very costly impl. of OQ*

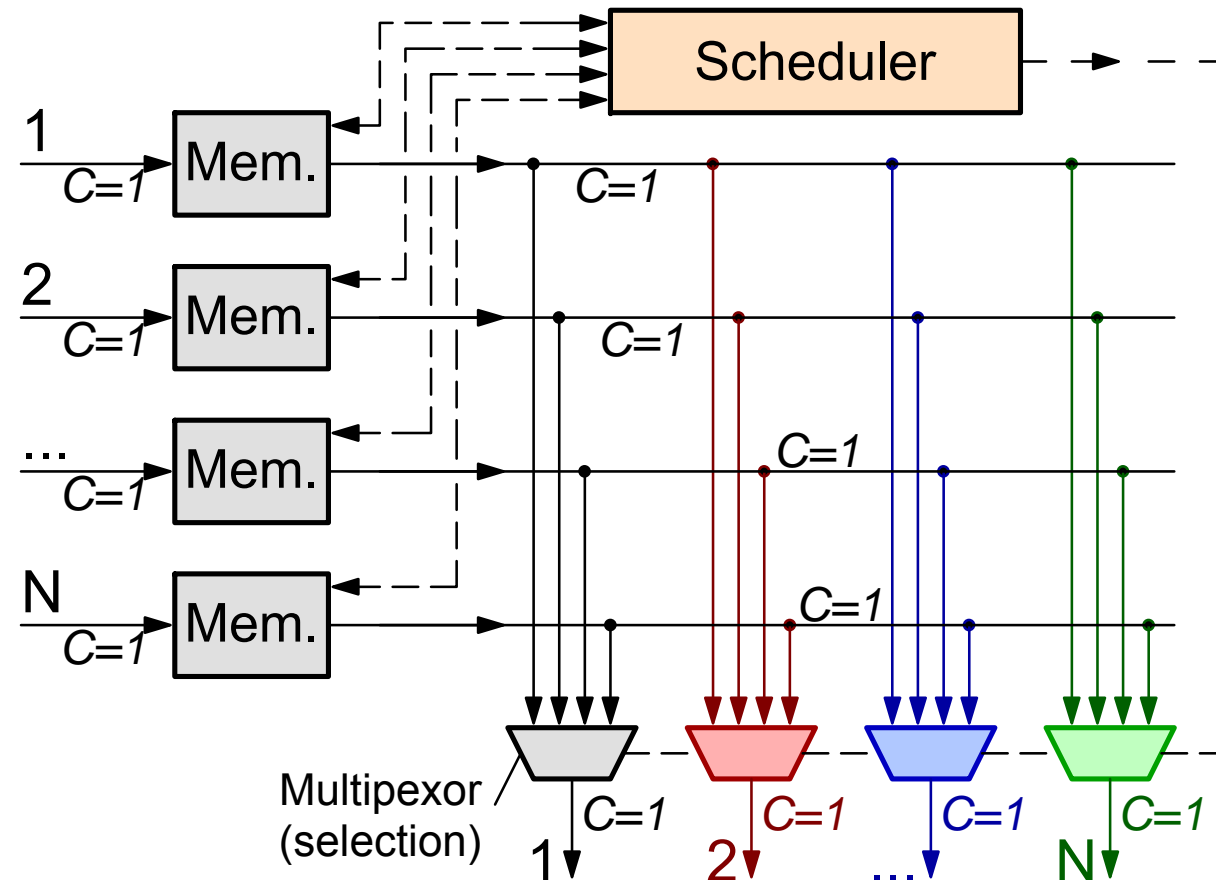


- + Same high performance & minimum delay as Output Q'ng or Shared Buf.
- + Scalable: each memory needs throughput of only 2, *independent of N*
- Very expensive: total memory thruput = $2 \cdot N^2$, versus $2 \cdot N$ for shared buf.
- *Highly* partitioned memory: very poor buffer space utilization

1.4 Queueing Architectures – family 2

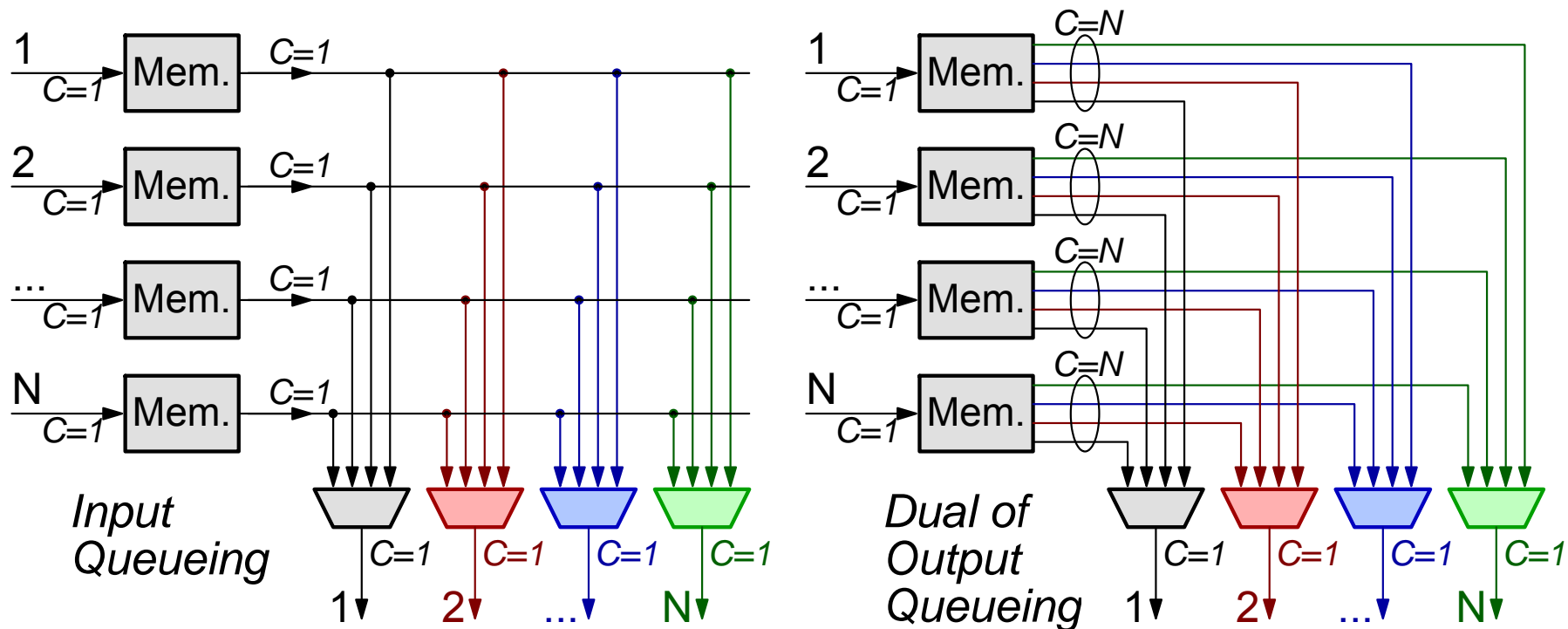
Input Queueing – the “practical” architecture

- Per-input buffer memories
- + Scalable: each memory needs throughput = 2
 \Rightarrow feasible
independ. of N
- + Low cost:
 total memory thruput = $2 \cdot N$
 – same as shared buffer



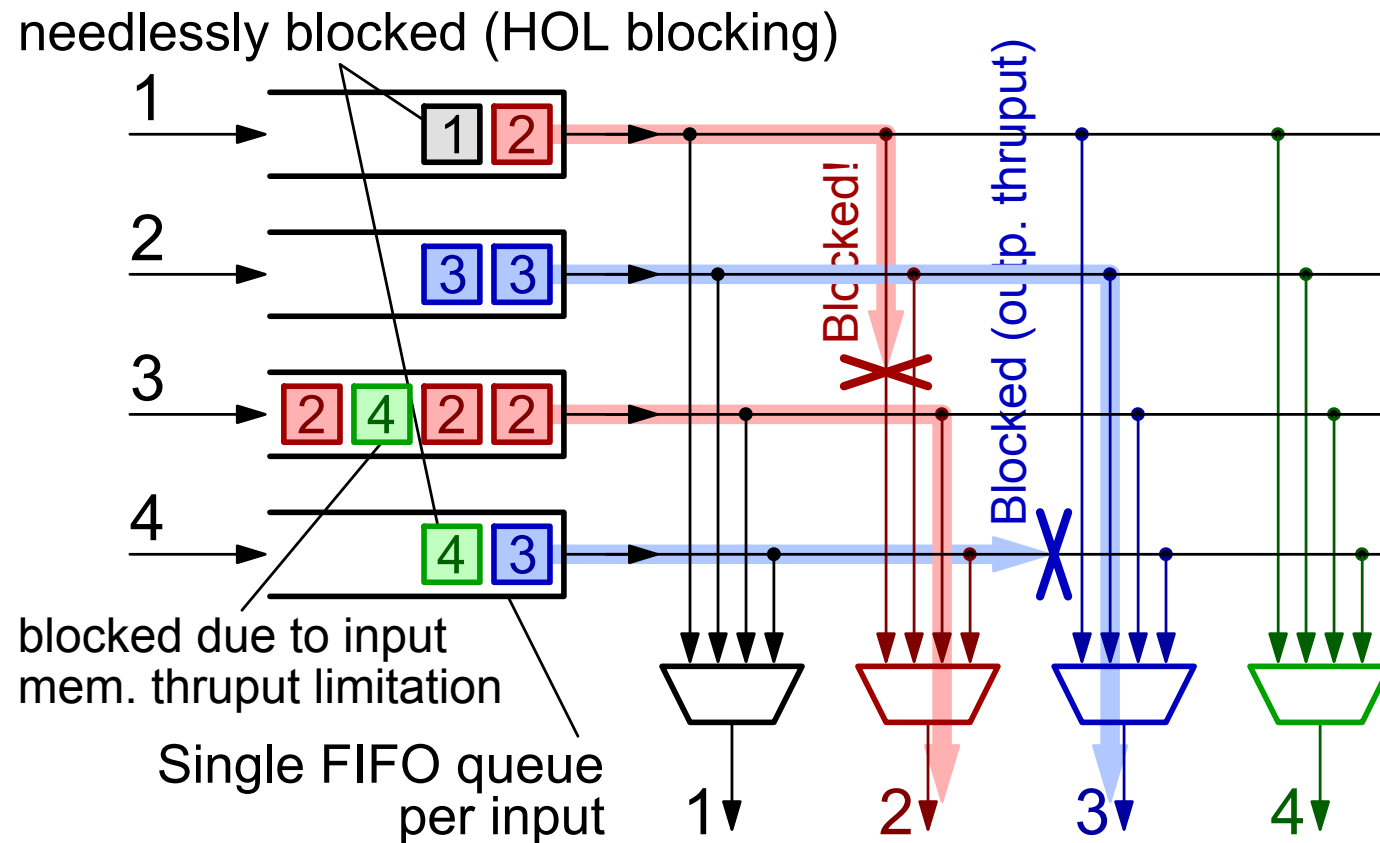
- Performance suffers a lot, unless (i) multiple queues per input, and (ii) sophisticated scheduler, and usually (iii) other modifications to be seen later (small crosspoint Q's, or internal speedup and OQ's)

Input Queueing is *not* the dual of Output Queueing



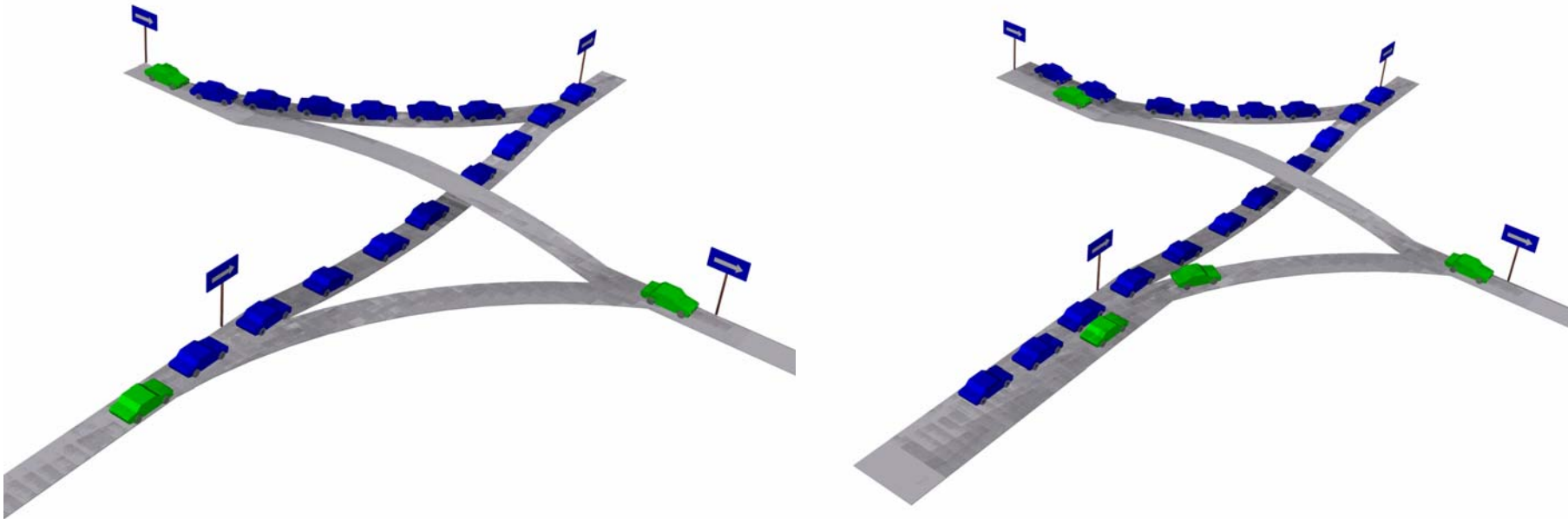
- **Asymmetry** between packet arrival and packet departure conflicts:
- Simultaneous **arrivals** may conflict with each other (packets destined to the same output), and the switch is obliged to accept them.
- Simultaneous **departures** are scheduled by the switch \Rightarrow can be made to *not* originate from the same input – albeit at potential performance cost.

Head-of-Line (HOL) Blocking



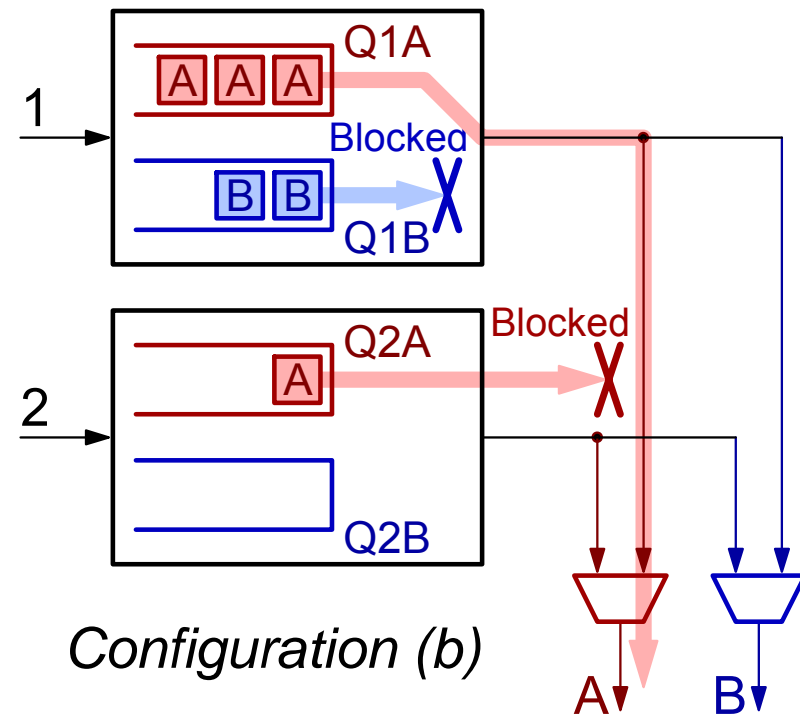
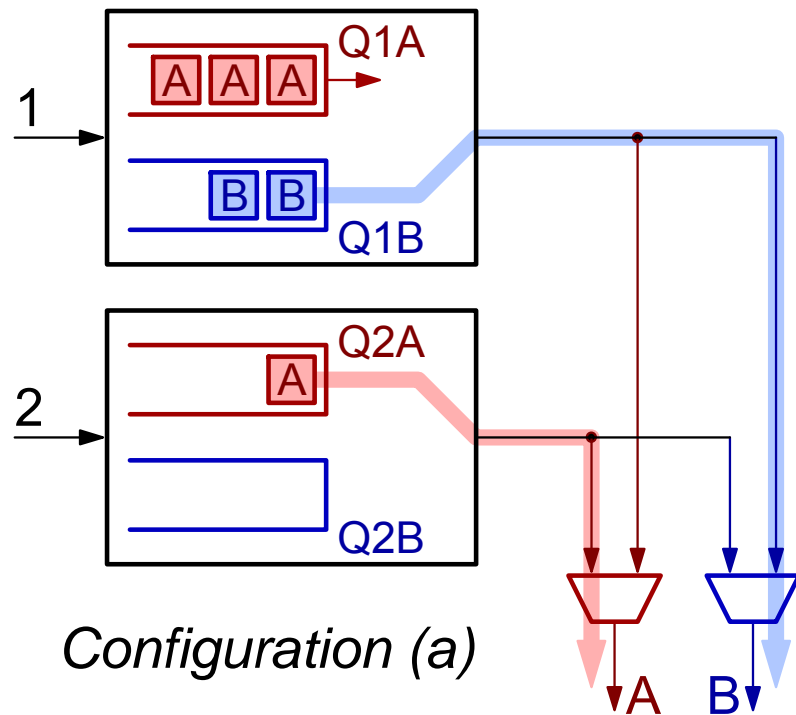
- Whenever one First-in-First-Out (FIFO) queue feeds multiple destinations, beware of the danger of head-of-line (HOL) blocking (called HOL-blocking when bottleneck is the FIFO organization – *not* when other bottleneck, e.g. memory read throughput or output port throughput)

Multiple Lanes needed to resolve HOL blocking



- As with cars in the road network, HOL blocking spreads the negative effects of congestion (blue cars) to other, unrelated traffic (green cars)
- Multiple “lanes” (queues, virtual circuits) (as many as the congested destinations???) can resolve this problem when properly architected (when packets heading to a common congestion point are prevented from occupying more than one lane) – how should we do this?...

Crossbar Scheduling with Multiple Queues is tricky



- Per-output queues at the inputs avoid HOL blocking – (a): even if **Q1A cells** are older, the younger **Q1B cells** can bypass them, since they reside in a separate Q and the scheduler can see them.

- Which of the two configurations should the scheduler choose?
- (a) for higher aggregate thruput?
- ... but then, flow **Q1A** will starve!

Summary of Queueing Architectures – $N \times N$ switch

Architecture	per-mem thruput	num.of mem's	tot.mem thruput	mem.sp utilizatn	Perfor- mance	Com- plexity	Con- clusions
Shared Buffer	$2 \cdot N$	1	$2 \cdot N$	best	best	multiple queues	best if feasible
Output Q'ing	$N+1$	N	N^2+N	medium	best	simple	refer'nc only
Crosspt Q'ing	2	N^2	$2 \cdot N^2$	worst	best	simple	simple scalable
Inp.Q singleQ	2	N	$2 \cdot N$	medium	worst	simple	simple, poor perf
Inp.Q multiQ	2	N	$2 \cdot N$	medium	medium	multiQ's, schedul'r	textbook only
Variants (later)	2 to 4	$2 \cdot N++$	$4 \cdot N$ to $8 \cdot N++$	medium	very good	multiQ's, schedul'r	practical scalable