CS 268: Lecture 12 (Router Design)

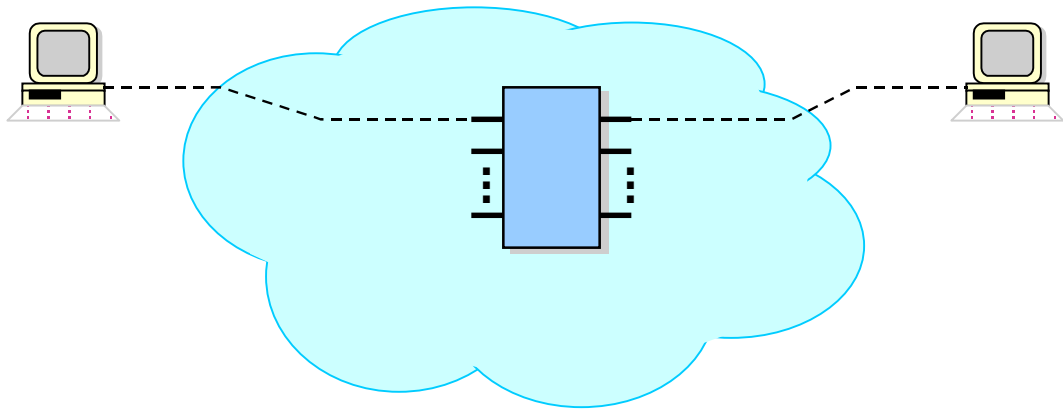
Ion Stoica

March 18, 2002

Midterm Exam (March 20): Sample Questions

- E2E principle
  - Describe the end-to-end principle. Give one example in which implementing a particular functionality at a lower layer breaks this principle, and one example in which it does not. Explain.
- Fair Queueing
  - (a) What problem does Fair Queueing address? Describe the Fair Queuing algorithm.
  - (b) What is the system virtual time and what it is used for?
- Differentiated Services
  - Compare Assured and Premium services. How is each of them implemented at edge and core routers?

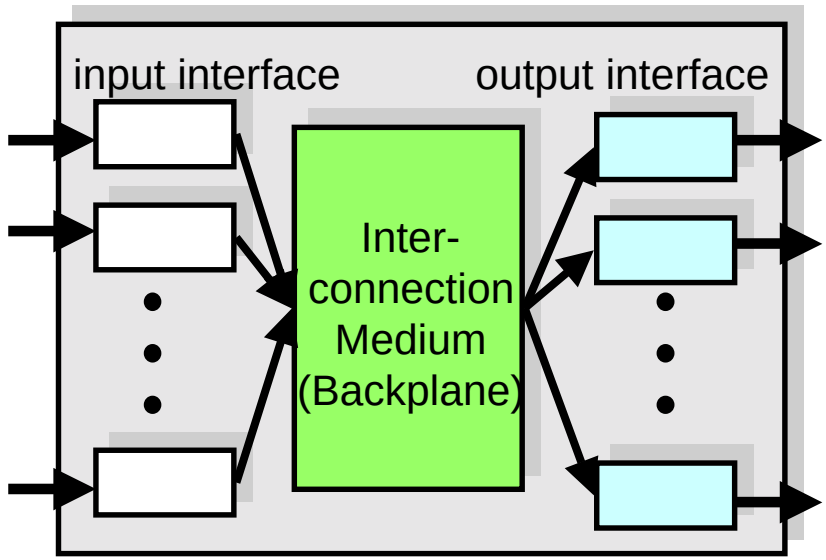
IP Router



- A router consists
  - A set of input interfaces at which packets arrive
  - A se of output interfaces from which packets depart
- Router implements two main functions
  - Forward packet to corresponding output interface
  - Manage congestion

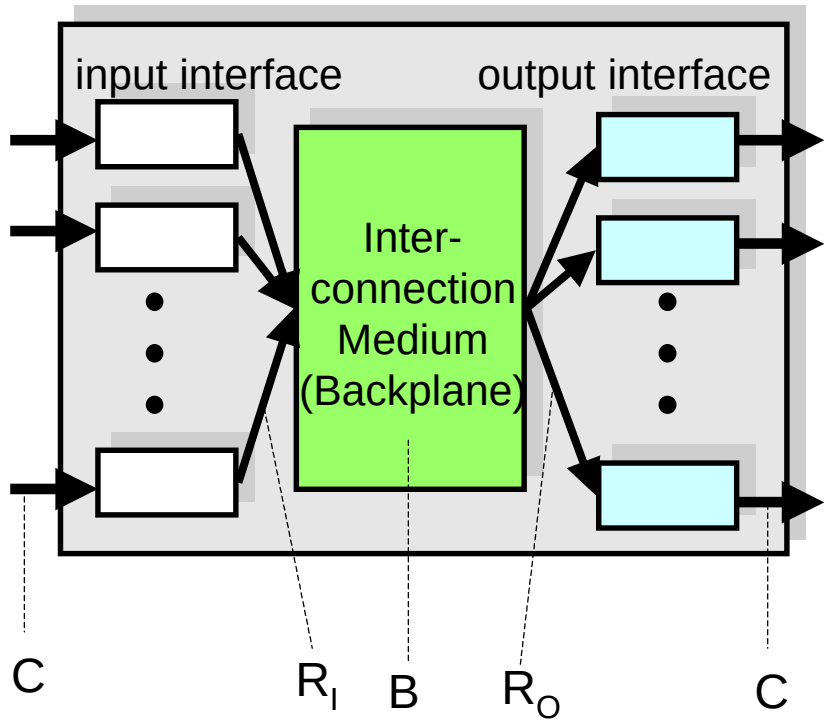
Generic Router Architecture

- Input and output interfaces are connected through a backplane
- A backplane can be implemented by
  - Shared memory
    - Low capacity routers (e.g., PC-based routers)
  - Shared bus
    - Medium capacity routers
  - Point-to-point (switched) bus
    - High capacity routers



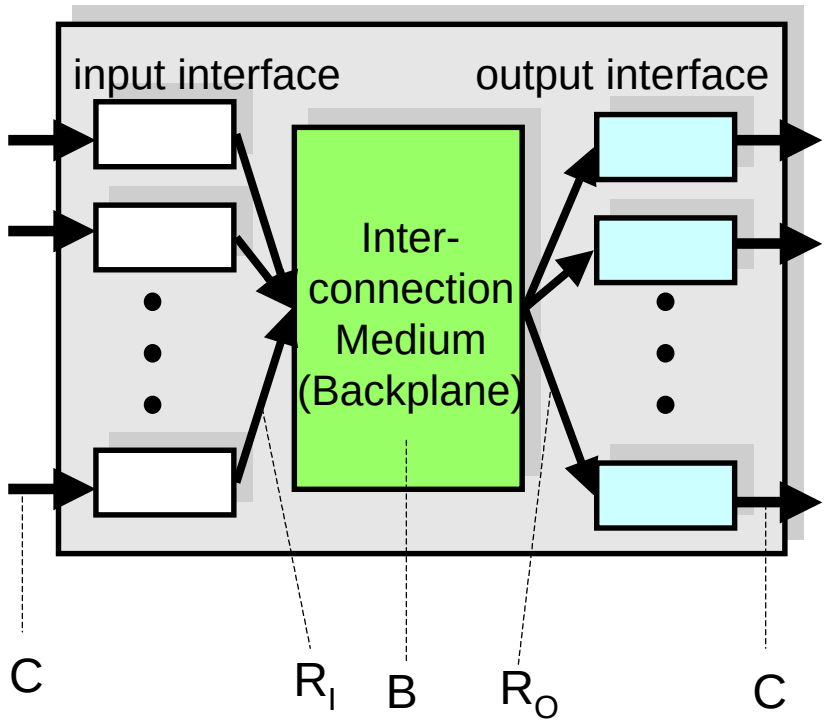
Speedup

- C – input/output link capacity
- R_I – maximum rate at which an input interface can send data into backplane
- R_O – maximum rate at which an output can read data from backplane
- B – maximum aggregate backplane transfer rate
- Back-plane speedup: B/C
- Input speedup: R_I/C
- Output speedup: R_O/C



Function division

- Input interfaces:
  - Must perform packet forwarding – need to know to which output interface to send packets
  - May enqueue packets and perform scheduling
- Output interfaces:
  - May enqueue packets and perform scheduling

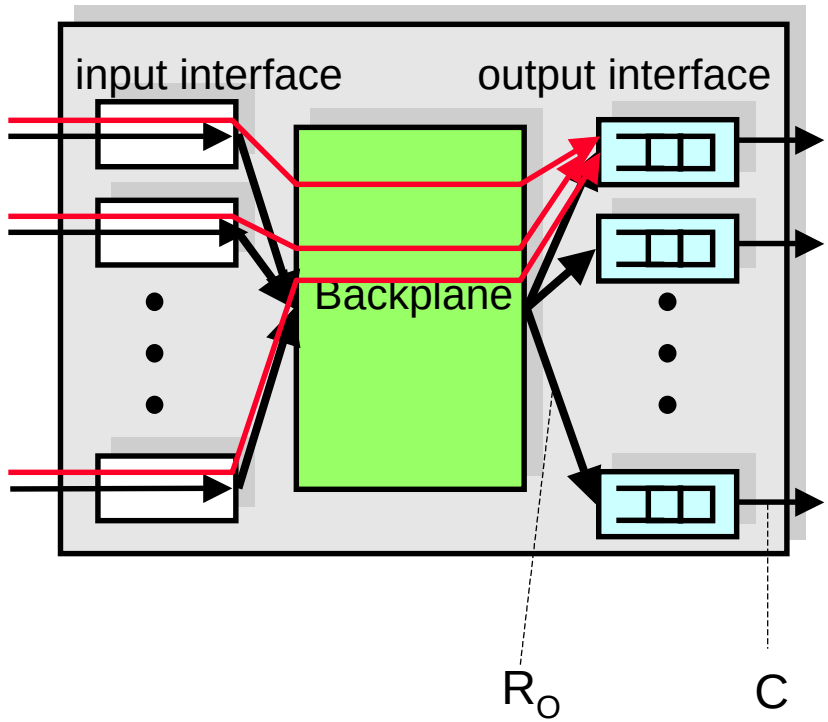


Three Router Architectures

- Output queued
- Input queued
- Combined Input-Output queued

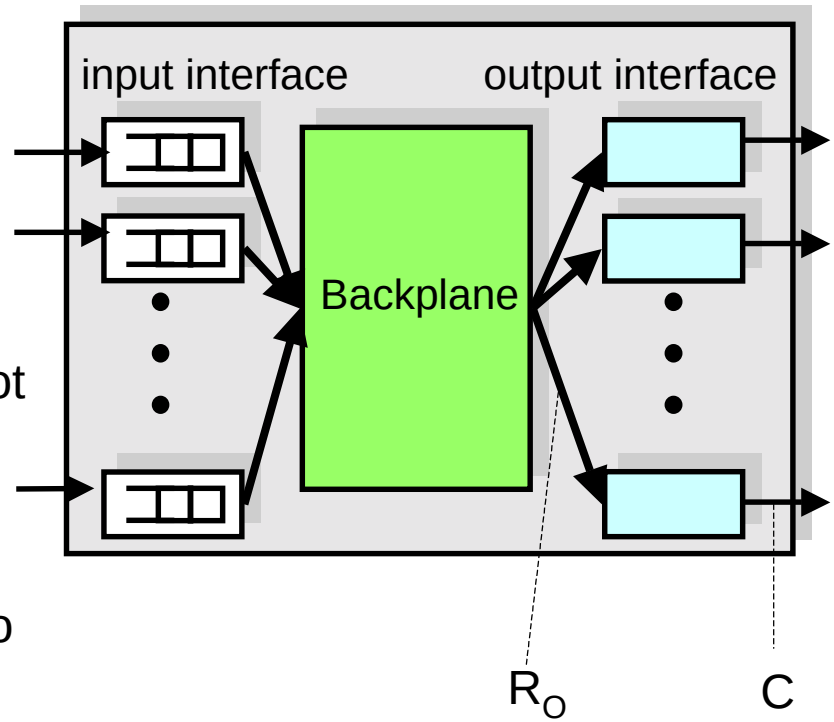
Output Queued (OQ) Routers

- Only output interfaces store packets
- Advantages
  - Easy to design algorithms: only one congestion point
- Disadvantages
  - Requires an output speedup of N, where N is the number of interfaces → not feasible



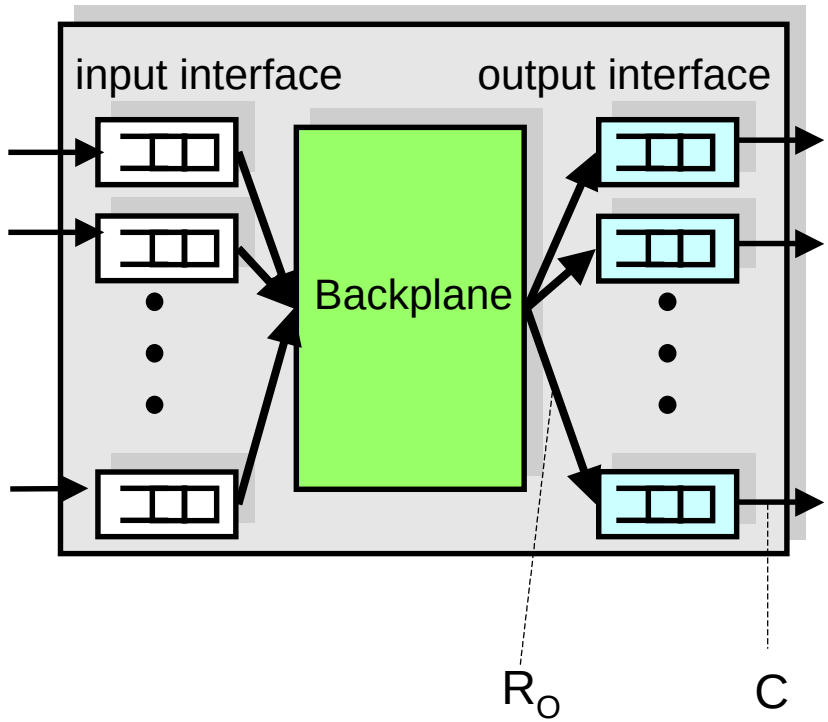
Input Queueing (IQ) Routers

- Only input interfaces store packets
- Advantages
  - Easy to built
    - Store packets at inputs if contention at outputs
  - Relatively easy to design algorithms
    - Only one congestion point, but not output…
    - need to implement backpressure
- Disadvantages
  - Hard to achieve utilization → 1 (due to output contention, head-of-line blocking)
    - However, theoretical and simulation results show that for realistic traffic an input/output speedup of 2 is enough to achieve utilizations close to 1



Combined Input-Output Queueing (CIOQ) Routers

- Both input and output interfaces store packets
- Advantages
  - Easy to built
    - Utilization 1 can be achieved with limited input/output speedup (<= 2)
- Disadvantages
  - Harder to design algorithms
    - Two congestion points
    - Need to design flow control
  - Note: recent results show that with a input/output speedup of 2, a CIOQ can emulate any work-conserving OQ [G+98,SZ98]

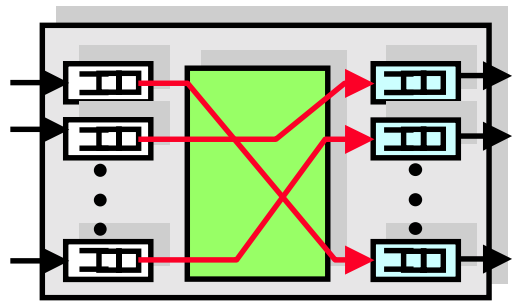


Generic Architecture of a High Speed Router Today

- Combined Input-Output Queued Architecture
  - Input/output speedup <= 2
- Input interface
  - Perform packet forwarding (and classification)
- Output interface
  - Perform packet (classification and) scheduling
- Backplane
  - Point-to-point (switched) bus; speedup N
  - Schedule packet transfer from input to output

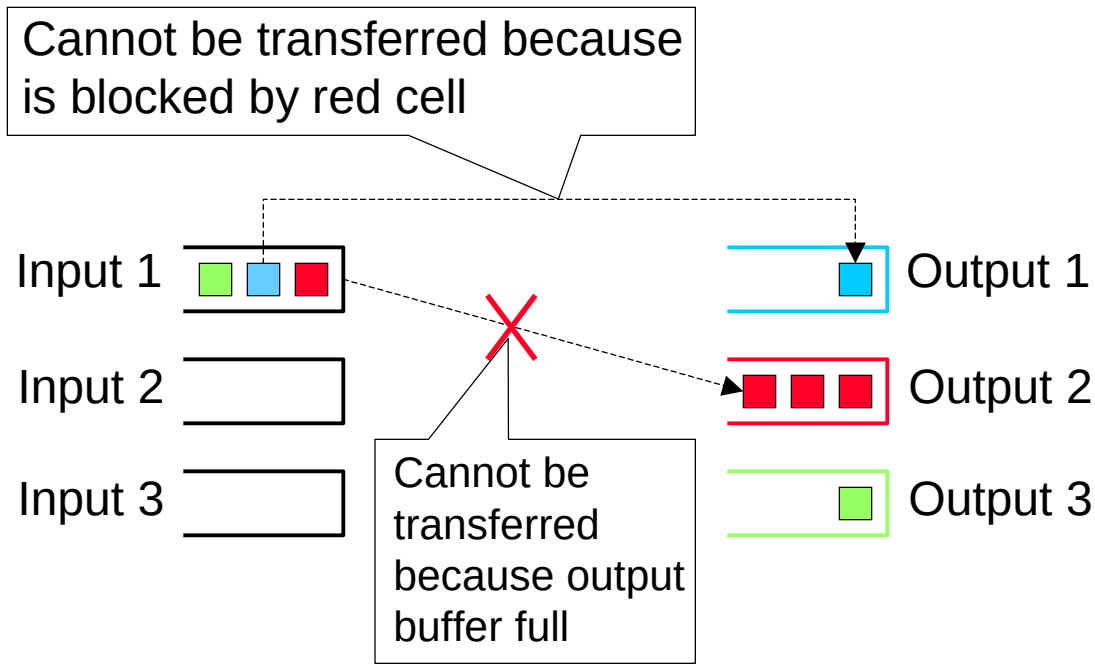
Backplane

- Point-to-point switch allows to simultaneously transfer a packet between any two disjoint pairs of input-output interfaces
- Goal: come-up with a schedule that
  - Meet flow QoS requirements
  - Maximize router throughput
- Challenges:
  - Address head-of-line blocking at inputs
  - Resolve input/output speedups contention
  - Avoid packet dropping at output if possible
- Note: packets are fragmented in fix sized cells (why?) at inputs and reassembled at outputs
  - In Partridge et al, a cell is 64 B (what are the trade-offs?)



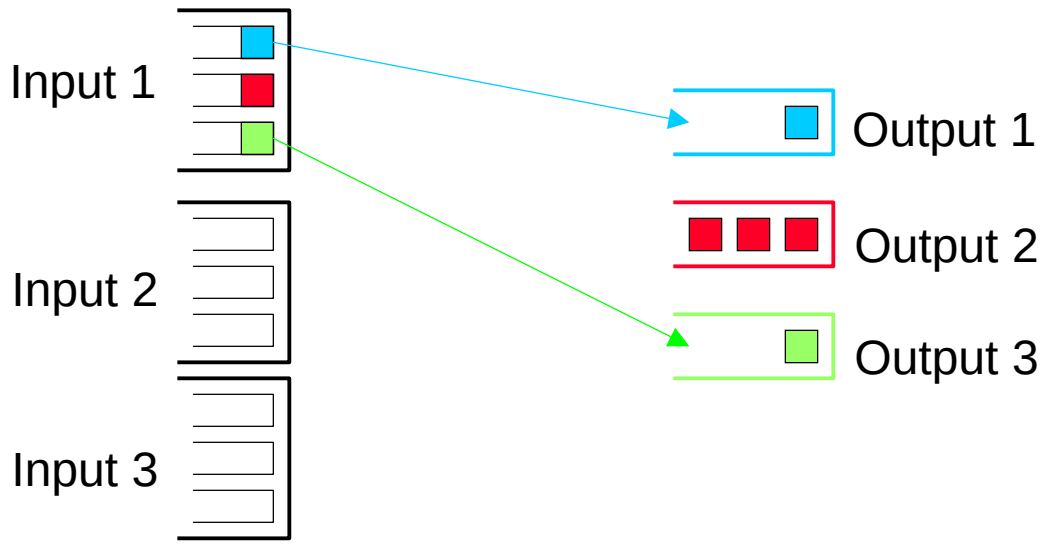
Head-of-line Blocking

- The cell at the head of an input queue cannot be transferred, thus blocking the following cells



Solution to Avoid Head-of-line Blocking

- Maintain at each input N virtual queues, i.e., one per output



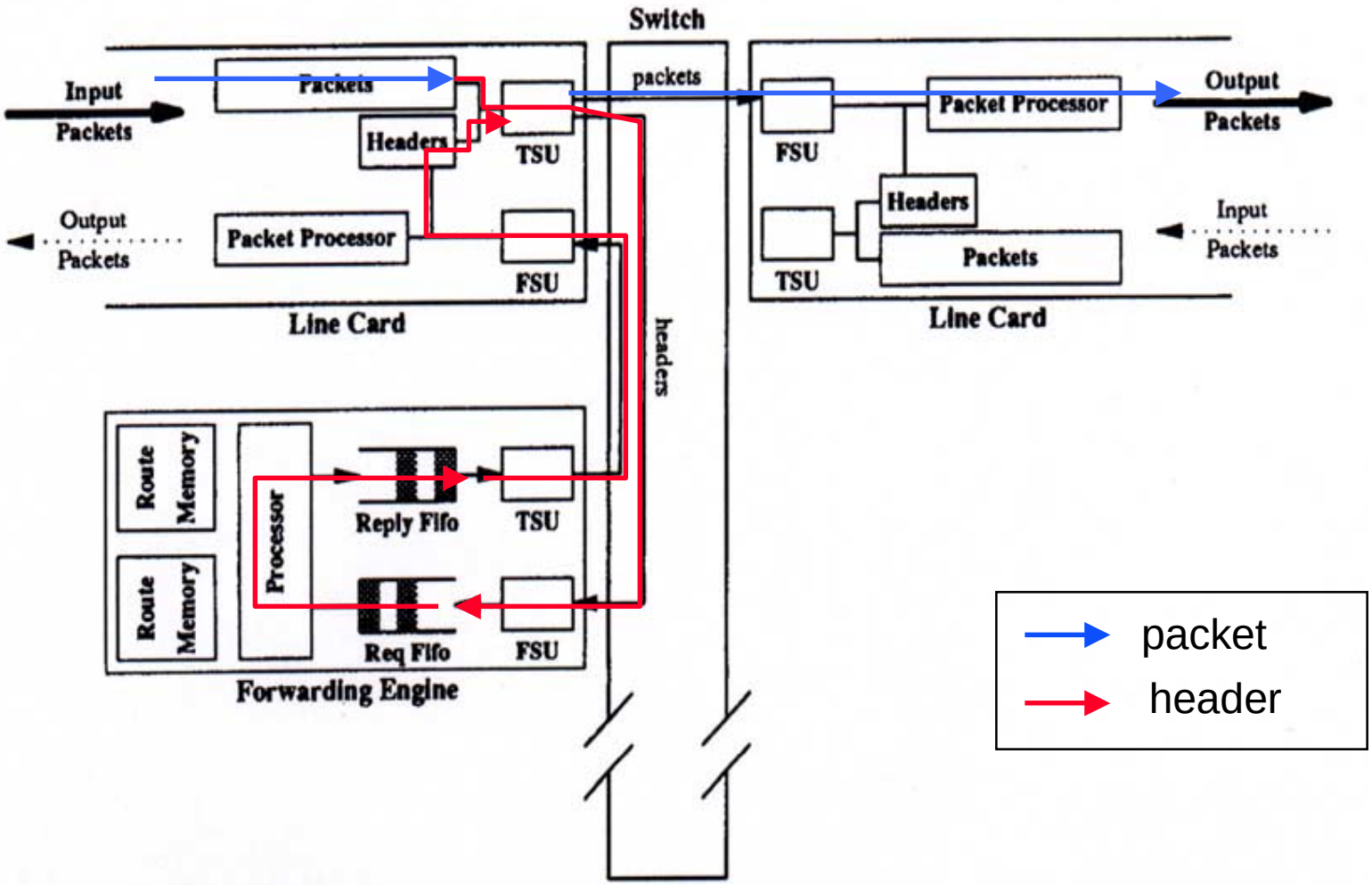
Cell transfer

- Schedule:
  - Ideally: find the maximum number of input-output pairs such that:
    - Resolve input/output contentions
    - Avoid packet drops at outputs
    - Packets meet their time constraints (e.g., deadlines), if any
- Example
  - Assign cell preferences at inputs, e.g., their position in the input queue
  - Assign cell preferences at outputs, e.g., based on packet deadlines, or the order in which cells would depart in a OQ router
  - Match inputs and outputs based on their preferences
- Problem:
  - Achieving a high quality matching complex, i.e., hard to do in constant time

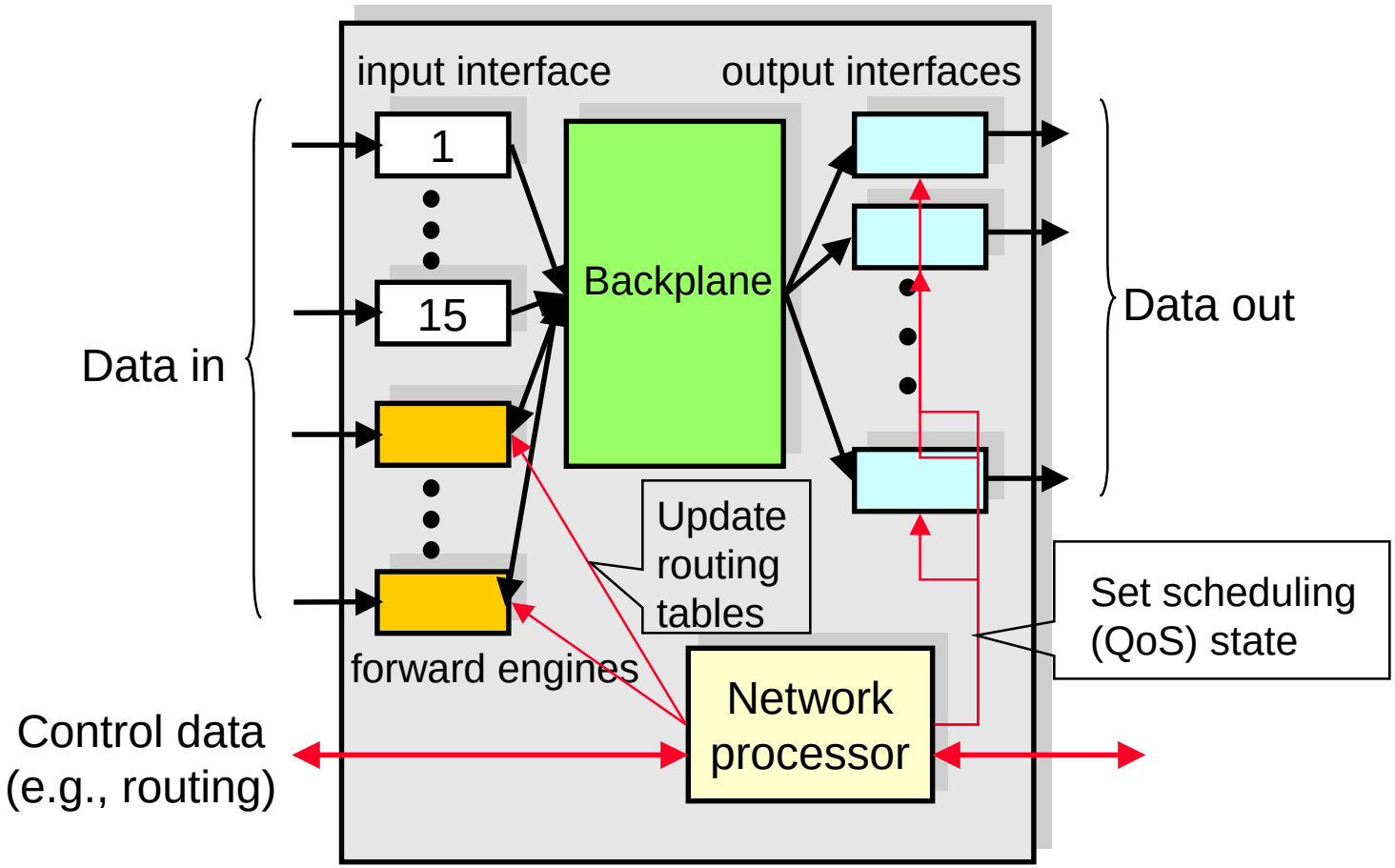
A Case Study [Partridge et al ’98]

- Goal: show that routers can keep pace with improvements of transmission link bandwidths
- Architecture
  - A CIOQ router
  - 15 (input/output) line cards: C = 2.4 Gbps
    - Each input card can handle up to 16 (input/output) interfaces
    - Separate forward engines (FEs) to perform routing
  - Backplane: Point-to-point (switched) bus, capacity B = 50 Gbps (32 MPPS)
    - B/C = 20, but 25% of B lost to overhead (control) traffic

Router Architecture



Router Architecture



Router Architecture: Data Plane

- Line cards
  - Input processing: can handle input links up to 2.4 Gbps (3.3 Gbps including overhead)
  - Output processing: use a 52 MHz FPGA; implements QoS
- Forward engine:
  - 415-MHz DEC Alpha 21164 processor, three level cache to store recent routes
    - Up to 12,000 routes in second level cache (96 kB); ~ 95% hit rate
    - Entire routing table in tertiary cache (16 MB divided in two banks)

Router Architecture: Control Plane

- Network processor: 233-MHz 21064 Alpha running NetBSD 1.1
  - Update routing
  - Manage link status
  - Implement reservation
- Backplane Allocator: implemented by an FPGA
  - Schedule transfers between input/output interfaces

Data Plane Details: Checksum

- Takes too much time to verify checksum
  - Increases forwarding time by 21%
- Take an optimistic approach: just incrementally update it
  - Safe operation: if checksum was correct it remains correct
  - If checksum bad, it will be anyway caught by end-host
- Note: IPv6 does not include a header checksum anyway!

Data Plane Details: Slow Path Processing

1. Headers whose destination misses in the cache
2. Headers with errors
3. Headers with IP options
4. Datagrams that require fragmentation
5. Multicast datagrams
   - Requires multicast routing which is based on source address and inbound link as well
   - Requires multiple copies of header to be sent to different line cards

Control Plane: Backplane Allocator

- Time divided in epochs
  - An epoch consists of 16 ticks of data clock (8 allocation clocks)
- Transfer unit: 64 B (8 data click ticks)
- During one epoch, up to 15 simultaneous transfers in an epoch
  - One transfer: two transfer units (128 B of data + 176 auxiliary bits)
- Minimum of 4 epochs to schedule and complete a transfer but scheduling is pipelined.
  1. Source card signals that it has data to send to the destination card
  2. Switch allocator schedules transfer
  3. Source and destination cards are notified and told to configure themselves
  4. Transfer takes place
- Flow control through inhibit pins

The Switch Allocator Card

- Takes connection requests from function cards
- Takes inhibit requests from destination cards
- Computes a transfer configuration for each epoch
- 15X15 = 225 possible pairings with 15! Patterns

Allocator Algorithm

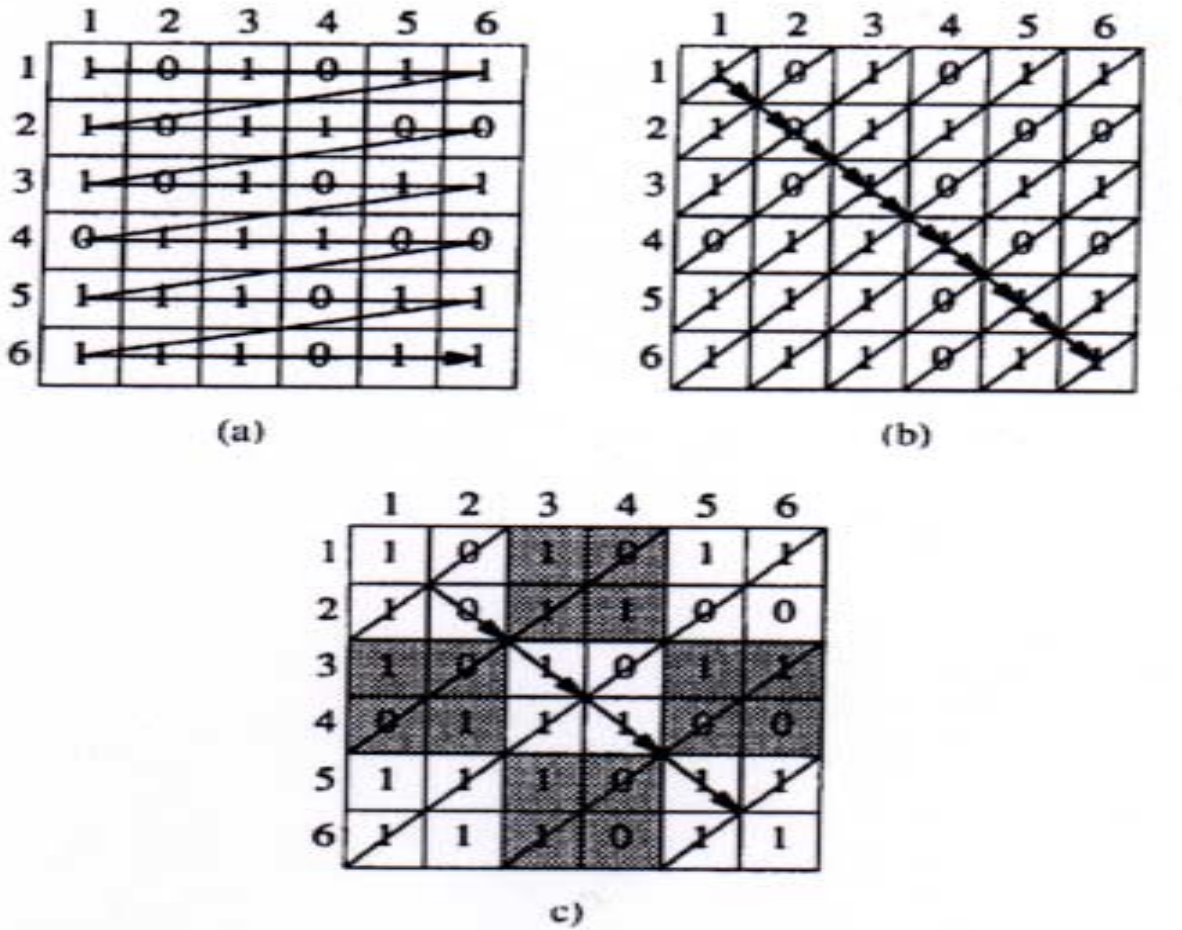


Fig. 4. Simple and wavefront allocators. (a) Simple. (b) Wavefront. (c) Group wavefront.

The Switch Allocator

- Disadvantages of the simple allocator
  - Unfair: there is a preference for low-numbered sources
  - Requires evaluating 225 positions per epoch, which is too fast for an FPGA
- Solution to unfairness problem: Random shuffling of sources and destinations
- Solution to timing problem: Parallel evaluation of multiple locations
- Priority to requests from forwarding engines over line cards to avoid *header contention* on line cards

Summary: Design Decisions (Innovations)

1. Each FE has a complete set of the routing tables
2. A switched fabric is used instead of the traditional shared bus
3. FEs are on boards distinct from the line cards
4. Use of an abstract link layer header
5. Include QoS processing in the router