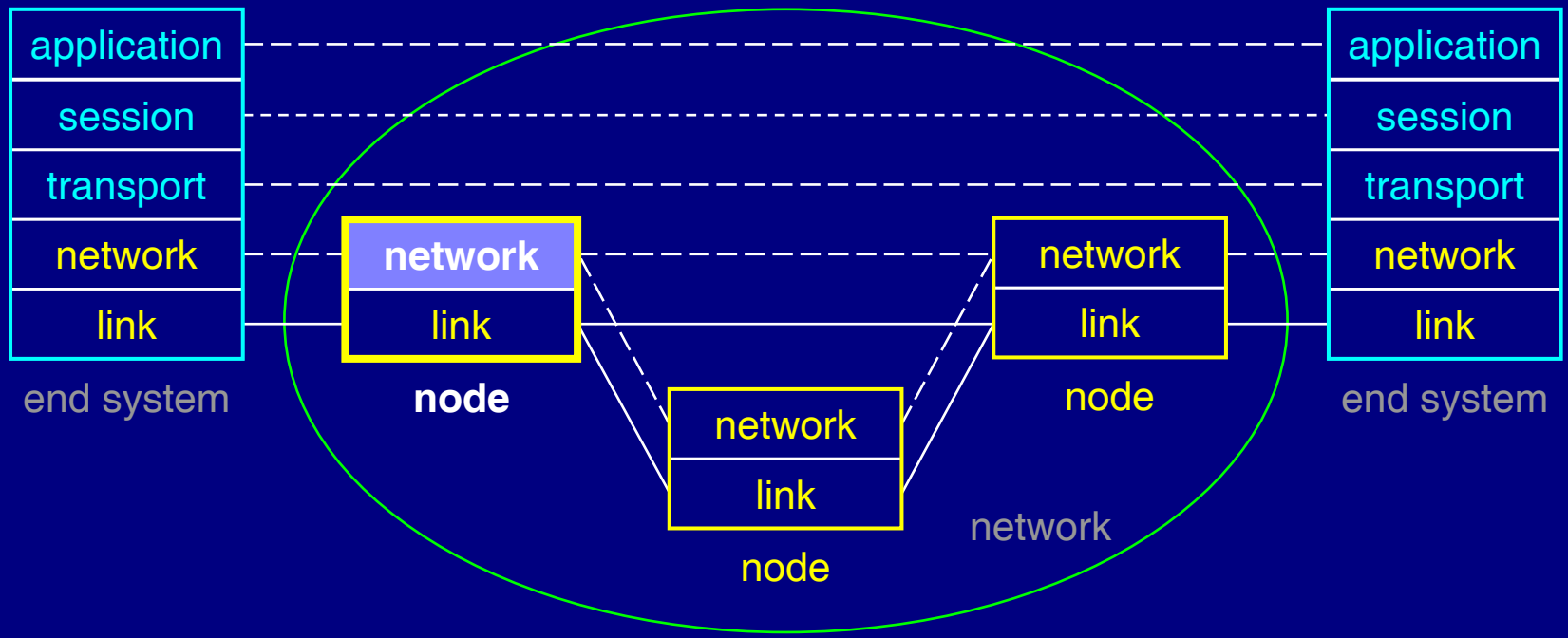
Switches and Routers

1. Introduction
2. Fundamentals and design principles
3. Network architecture and topology
4. Network control and signalling
5. Network components
   - 5.1 links
   - 5.2 switches and routers
6. End systems
7. End-to-end protocols
8. Networked applications
9. Future directions

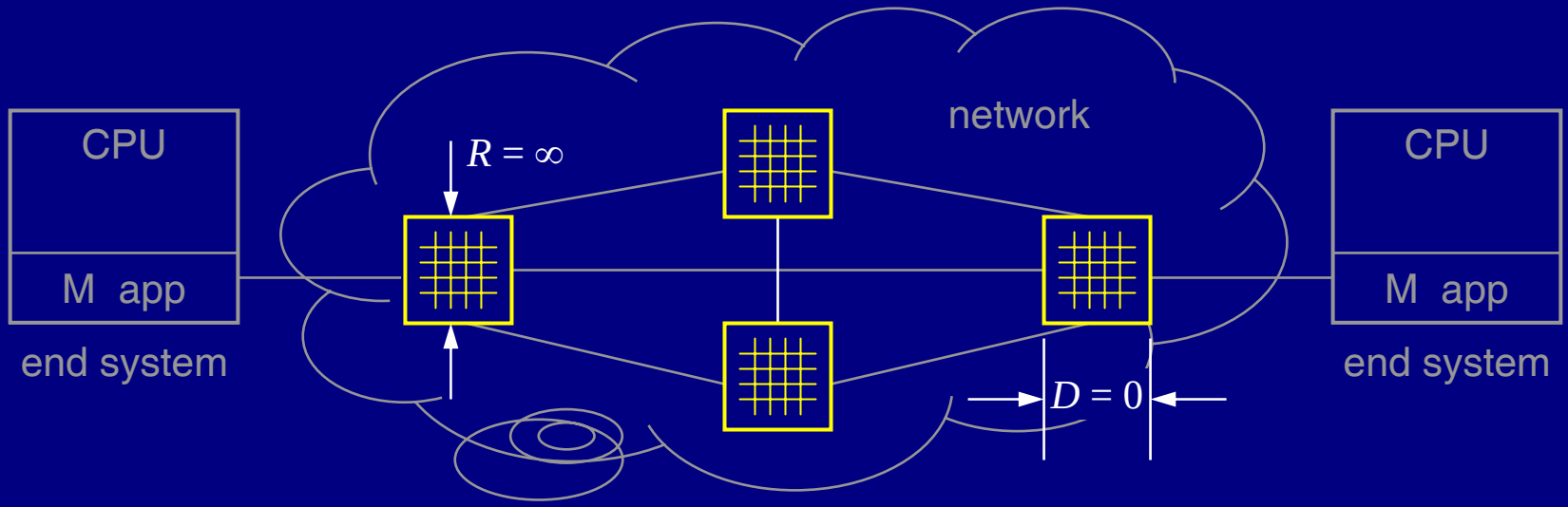
Switches and Routers



5.2. Switches and routers
5.3. Fast packet switches
5.4. Switch fabric architecture
5.5. Fast datagram switches
5.6. Programmable, active, and higher layer processing

Switches and Routers

Ideal Network
Network Node Principle

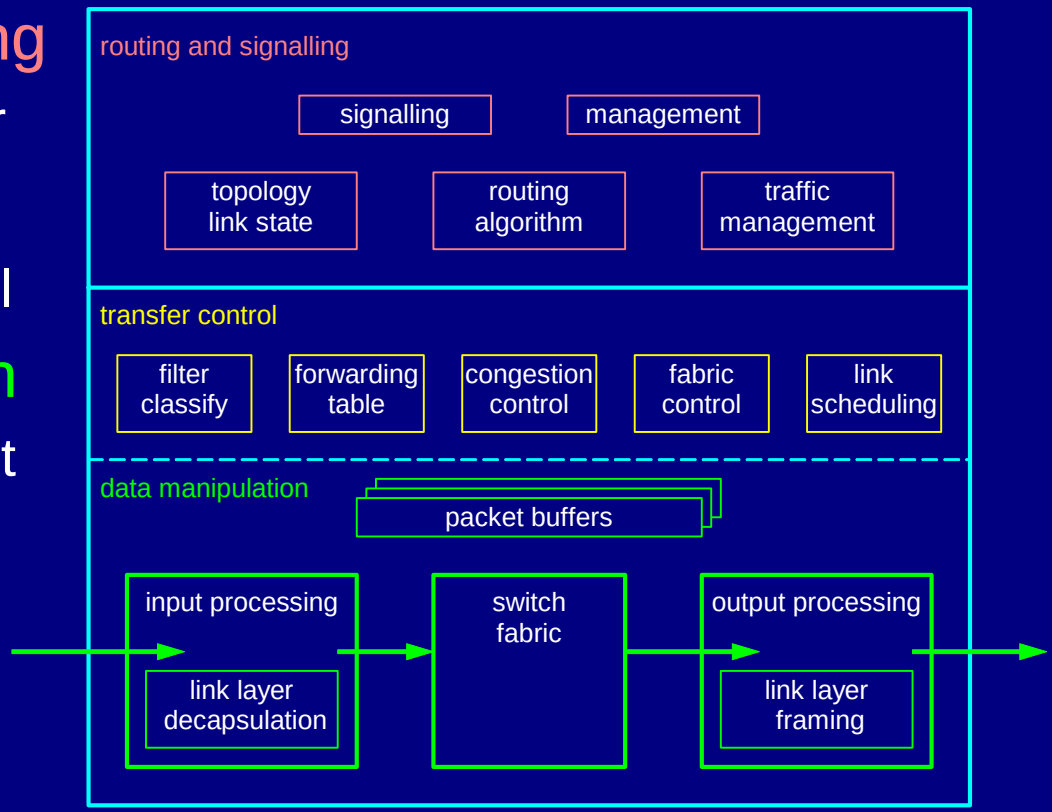


Network Node Principle S-II

Network nodes must support high-bandwidth low-latency, end-to-end flows, as well as their aggregation. High-speed network nodes should provide a scalable number or high-bandwidth, low delay interconnections.

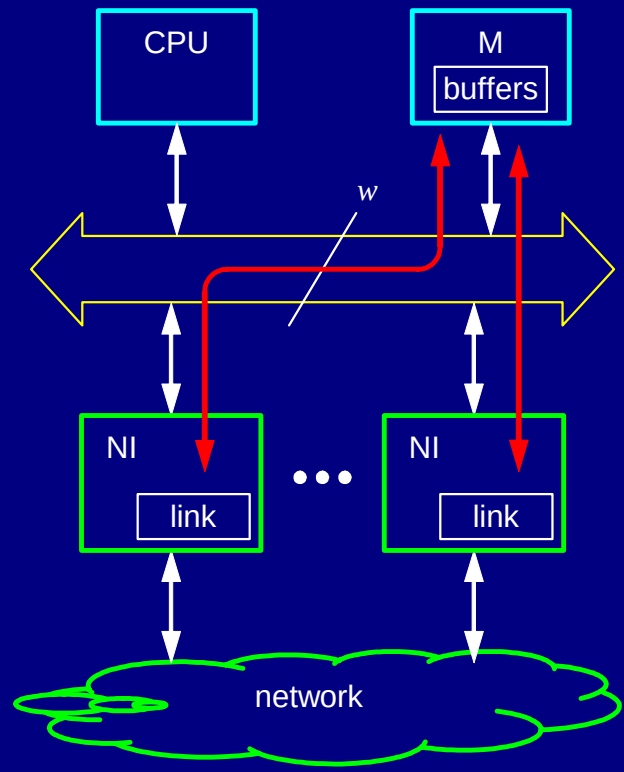
Switch Functions
Control, Transfer Control, Data

- Routing / signalling
  - per flow or longer
- Transfer control
  - per packet control
- Data manipulation
  - per byte or packet



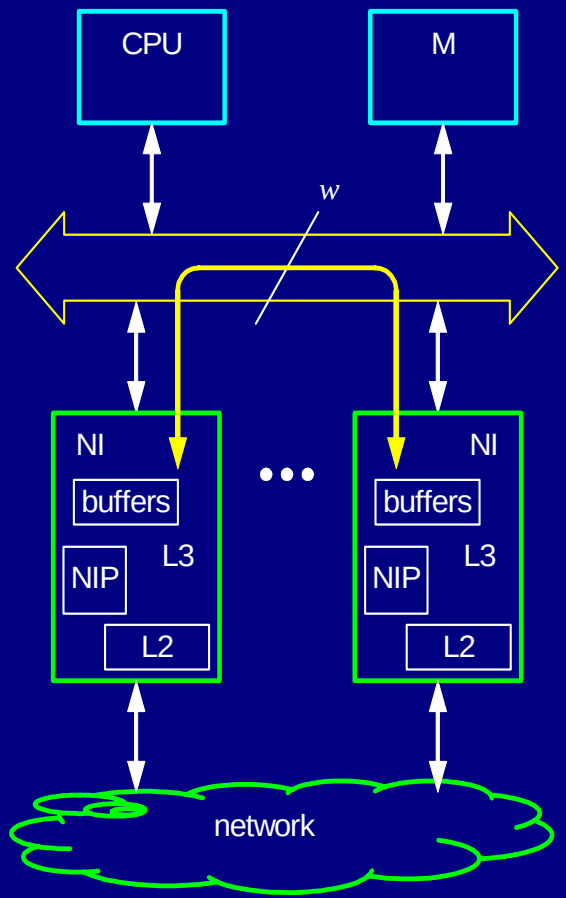
Store-and-Forward Routers
Second Generation

- Second generation (1980s)
  - delays
    - store-and-forward
    - contention for CPU
    - non-trivial header processing
  - buffering
    - in general purpose memory
    - contention for memory
  - shared bus interconnect
    - packets traverse bus twice
    - severely limits # of ports
    - DMA transfers help



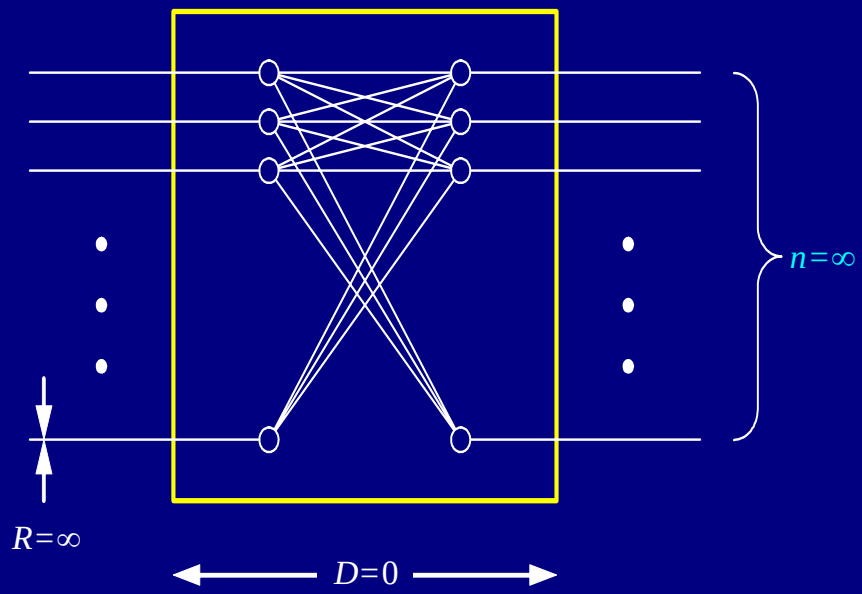
Store-and-Forward Routers
Third Generation

- Third generation (1990s)
  - NI (network interface)
    - packet processing
    - packet buffers
  - third party bus transfer
    - bus is switch fabric
    - single transfer per packet
      - still significant bottleneck



Ideal Switch Architecture
Bandwidth, Latency, Ports

- Infinite bandwidth
- Zero latency
- Unlimited number of ports



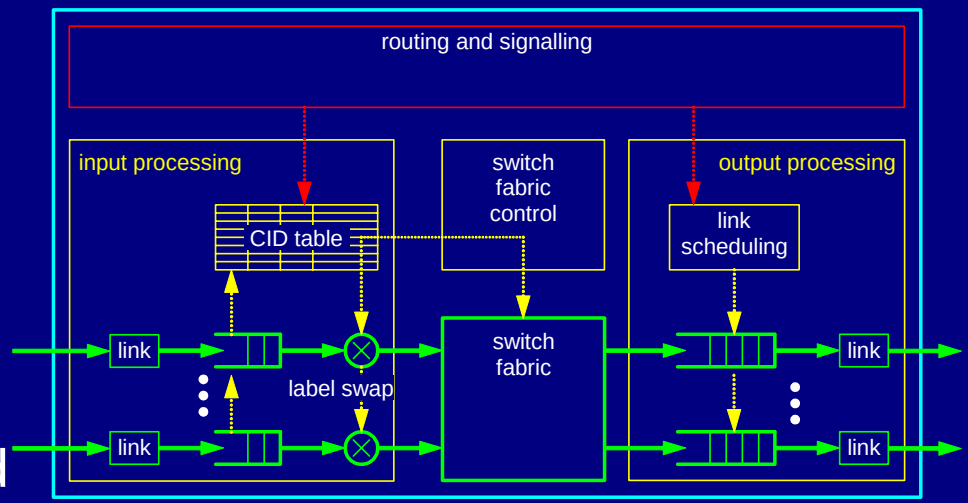
Store-and-Forward & Queueing Delay Minimisation S-II.3

Store-and-forward delays should be avoided, and per packet queueing should be minimised. In the ideal case, nodes should pipeline and cut through packet with zero per packet delays.

Switches and Routers
Fast Packet Switches

Fast Packet Switch
Architecture

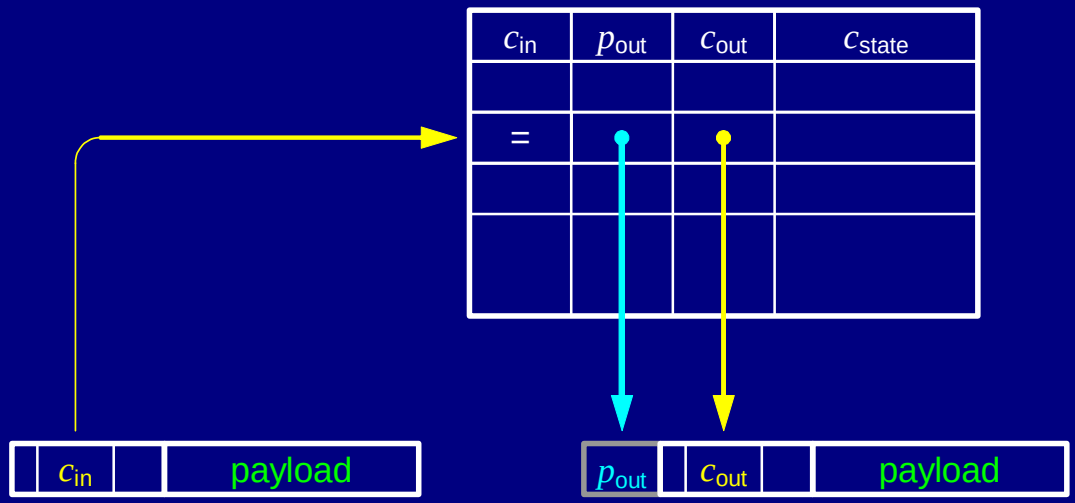
- Connection state
  - simple per packet processing
- Switch fabric
  - eliminate blocking
  - no store-and-forward



Fast Packet Switch Critical Path Principle S-1b*c*

Simplify and optimise per byte and per packet data manipulation and transfer control functions for implementation in a hardware critical path. Stare maintained per connection streamlines per packet processing.

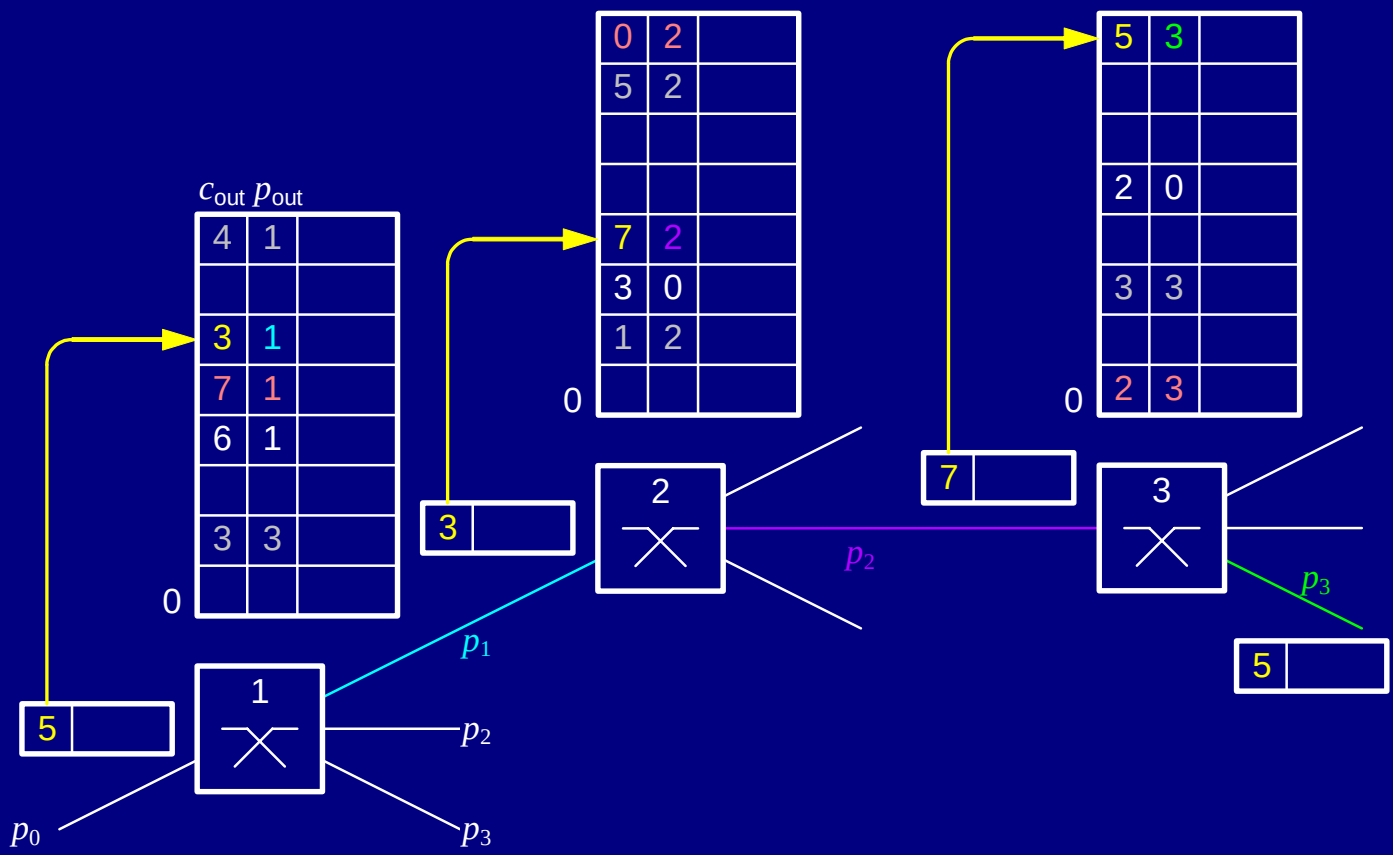
Fast Packet Switch
Label Swapping



- Connection id is index into table
- Table entry gives egress link and next hop id
  - port may be prepended for self-routing fabrics

  examples: ATM, MPLS

Fast Packet Switch
Label Swapping Example



Fast Packet Switch
Connectionless vs. Connection Tradeoff

- Connection-oriented fast packet switching
  - requires round trip connection setup latency
  - achieved higher data rate due to simple label swap
    - IP lookup was a bottleneck in 1980s

Connectionless vs. Connection Tradeoff S-5A

The latency of connection setup must be traded against the increase in packet processing rate and resultant bandwidth increase achievable due to the less complex label swapping.

Packet Structure
Size and Variability

- PDU size and structure are critical to performance

Packet Size and Variability Tradeoff S-8A

Packet size is a balance of a number of parameters that affect performance. Trade the statistical multiplexing and fine-grained control benefits of small packets against the efficiency of large packets. Trade the benefits to switch design of fixed cell size against the overhead of SAR and efficiency of variable packet size. Multiple discrete packet sizes increase flexibility at the cost of additional switch complexity. Hop-by-hop SAR localizes packet size optimizations at the cost of additional per hop delay and complexity.

Packet Size and Structure
Variability

- Fixed size (cells)
  - \+ easier to design switches
  - – difficult to predetermine the best size
- Variable size
  - – more difficult to design switches
  - \+ no need for agreement on size
  - \+ less need for fragmentation/segmemtation
- Discrete sizes: advantages of both fixed and variable
  - integral multiples, e.g. 64B, 128B, 192B...
  - power-of-2 scaling with data rate

    e.g. 128B @ OC-3, 256B @ OC-12, 512B @ OC-48

Packet Size and Structure

Size

- Small packets
  - \+ efficient statistical multiplexing
  - – high header/payload overhead
  - – short interarrival time challenge per packet processing
    - note: this is one major reason ATM failed
- Large packets
  - \+ significantly easier per packet processing
  - – less efficient statistical multiplexing
  - – larger queueing delays
  - \+ efficient transport of large data blocks
  - – inefficient transport of signalling and control messages

    e.g. TCP ACKs

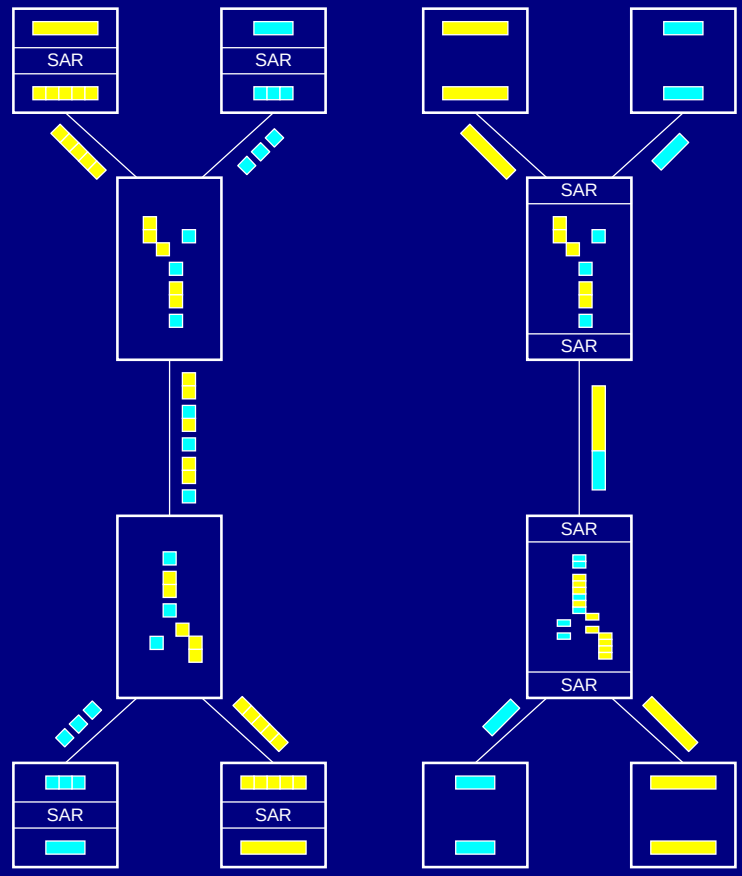
Packet Size and Structure
Granularity

- Important to match granularity to packet processing
  - + byte / octet (8 bits)
    - control fields should align to 8-bit boundaries
  - + word (typically 32 bits)
    - most end-system processing at word granularity
    - payload should align to 32-bit boundaries
  - + end system data unit
    - system buffers and memory structures
    - power-of-2 size will likely be integral fraction
  - + commodity memory components
    - power-of-2 size
  - – note: ATM cell was *none* of these

Packet Size and Structure

Segmentation and Reassembly

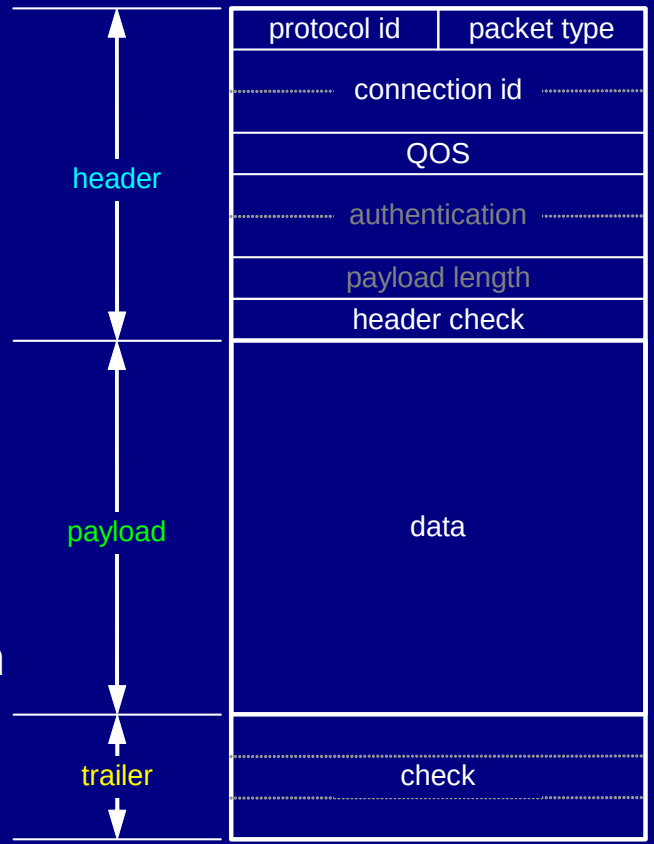
- Hop-by-hop
  - – significant overhead per switch
  - – each switch can use cell size optimised to its architecture
- End-to-end
  - + more efficient in the network
  - + requires global agreement on packet size



Packet Size and Structure

Packet Format

- Header
  - fields that *determine* packet processing
- Payload
  - TPDU transport protocol data unit
- Trailer
  - fields that are *dependent* on packet processing
    - e.g. cheksum to allow cut-through



Packet Size and Structure
Control Fields

- Control field structure and encoding is critical
  - simple encoding (bit vectors vs. code points)
  - byte/octet granularity and alignment
  - field length
    - fixed when possible
    - variable length prepended with length (skip vs. hunt)

Packet Control Field Structure S-8B

Optimise packet header and trailer fields for efficient processing. Fields should be simply encoded, byte aligned, and fixed length where possible. Variable-length fields should be prepended with their length.

Packet Structure

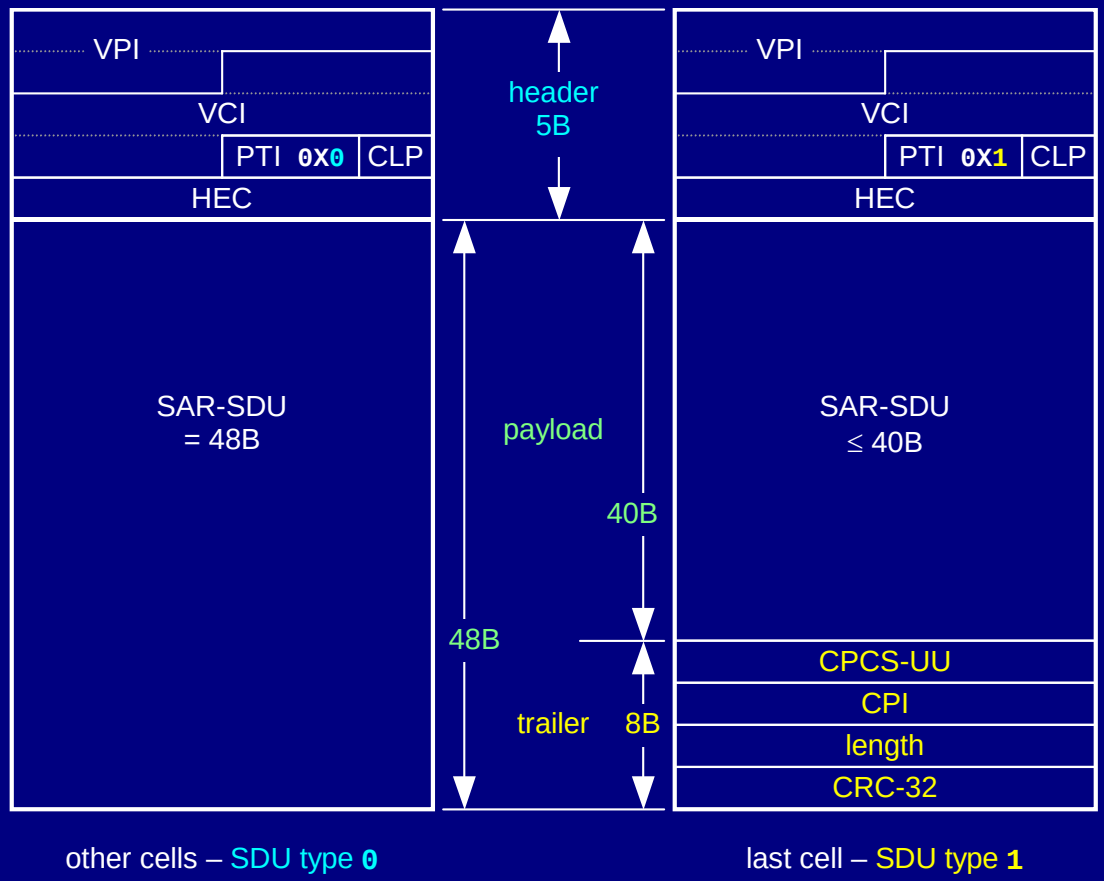
Example$_{5.4}$ ATM Cells

- ATM cell format
  - fast packet switching
  - fine-grained statistical multiplexing
- Size determined by ITU committee compromise
  - 48B = avg(32, 64)
    - 64 from US = min of proposals for data (and voice)
    - 32 from European PTTs to avoid voice echo cancellers
- Problems:
  - header tiny to keep overhead low; no room for seq #
  - nothing a power of 2
  - 48B + 5B = 53B; not even a multiple of 8

Packet Structure

Example 5.4 ATM AAL-5 Cells

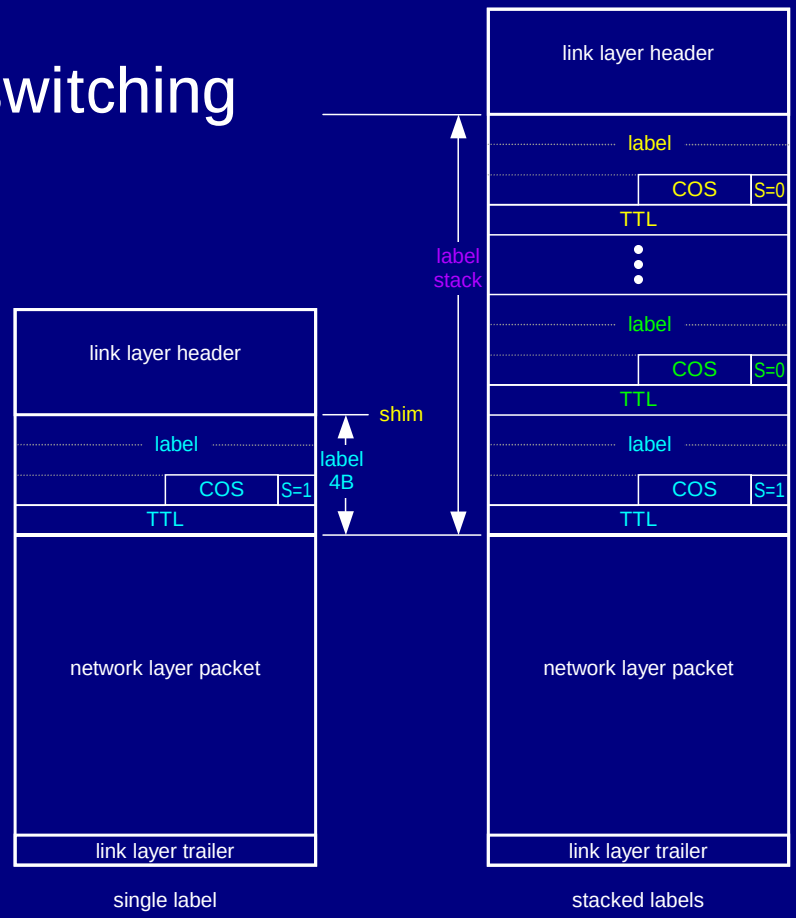
- AAL 3/4 for data
  - 4B AAL header
    - useless seq#
  - 44B payload
    - not div 8
- AAL 5
  - no header
  - trailer last frag
  - forced on ITU



Packet Structure

Example 5.5 MPLS Shim

- MPLS multiprotocol label switching
  - fast packet switching for IP
- Add label shim
  - switches swap label
  - stacked labels
    - allows net hierarchy (ala VP/VC)



Traffic Management
Connection-Oriented Fast Packet Switching

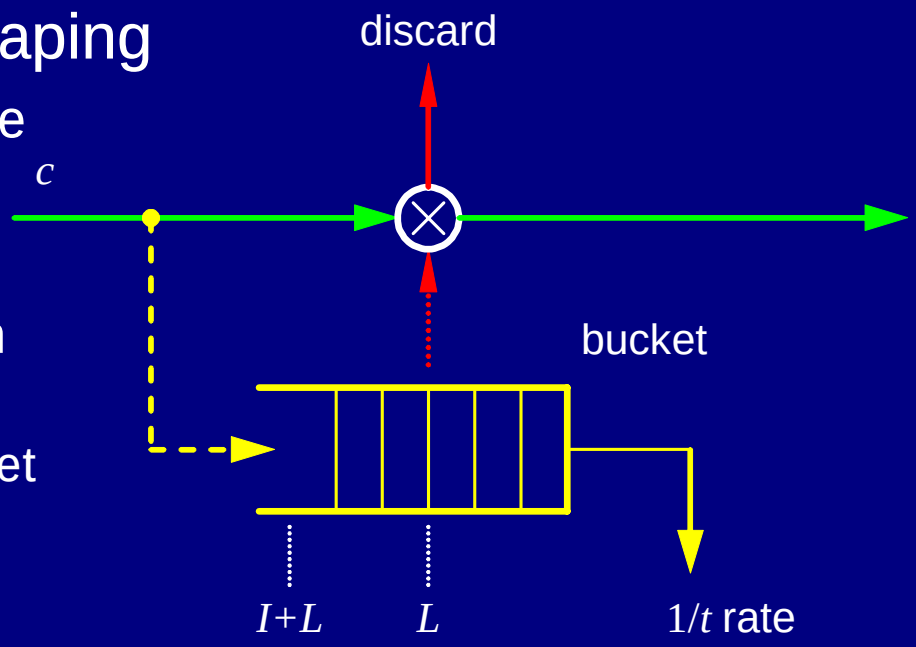
- Traffic management
  - required to protect high-performance flow characteristics
  - functions in critical path must be implemented at line speed
    - simple hardware required for third generation (1990s)

Switch Traffic Management S-II.2

In a resource-constrained environment, switches must support admission control with resource reservations to provide guaranteed service traffic classes and policing to prevent other traffic form interfering.

Traffic Management
Connection-Oriented Fast Packet Switching

- Policing and traffic shaping
  - leaky bucket to enforce
    - average rate
    - peak rate
  - simple implementation
    - packet arrival ⇒ token placed in bucket
    - bucket drained at
      - average rate
      - peak rate
    - if threshold L exceeded
      - packet dropped (may be marked if aggregate capacity available)
      - I may be used to limit burst lengths



Traffic Management
Connection-Oriented Fast Packet Switching

- Congestion avoidance
  - CAC
    connection admission control
  - policing at switch input
  - shaping at switch output
  - ECN
    explicit congestion notification
- Congestion control
  - drop policy
  - PPD
    partial packet discard
  - EPD
    early packet discard

Congestion Avoidance and Control

S-6C*c*

Bound offered load by admission control, traffic policing, and traffic shaping. When impending congestion is detected by building queues, notify sources to throttle. When congestion occurs, drop end-to-end frames if larger than network packets.

Traffic Management
Overengineering vs. Optimality

- Traffic management can be very complex
- Overengineering can dramatically simplify

Overengineering vs. Optimality S-2B

Optimal traffic management policies use processing and memory, which also adds latency to connection establishment; trade these against the bandwidth wasted by overengineering and overprovisioning the network.

Switch Functional Partitioning
Design and Technology Alternatives

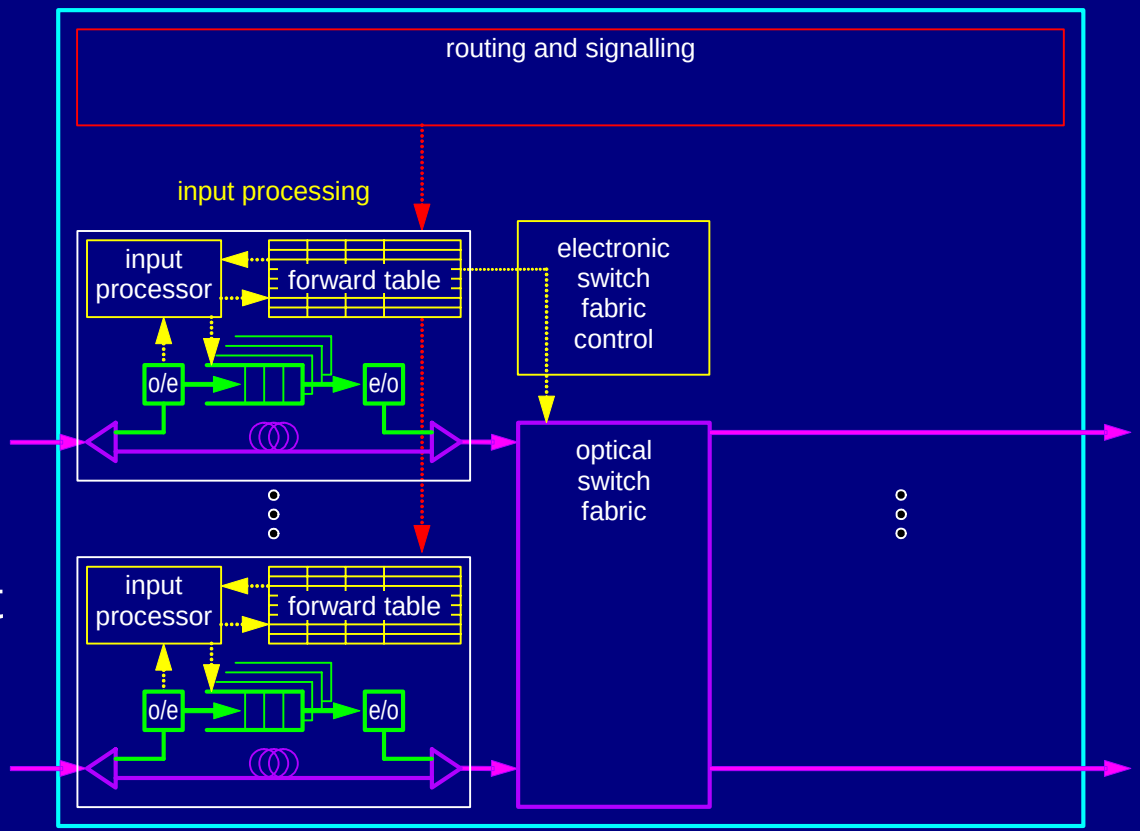
- Hardware vs. Software
  - control CPU vs. embedded controller vs. network processor
- Custom VLSI vs. ASIC vs. FPGA
- Technology
  - DRAM vs. SRAM vs. CAM
  - CMOS vs. GaAs
- Electronic vs. optical

Functional Partitioning and Assignment Principle S-1C

Carefully determine what switch functionality is to be implemented in scarce or expensive technology

Functional Partitioning
Electronic vs. Optical

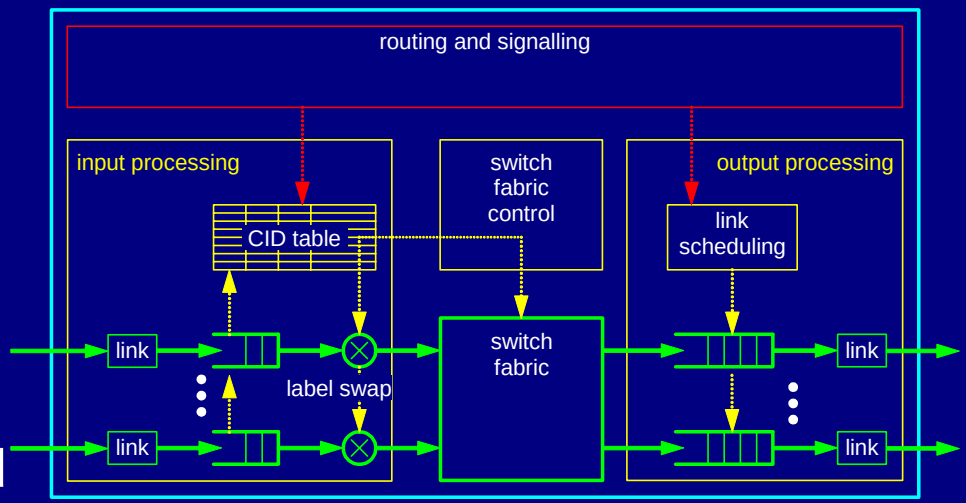
- Photonic processing infeasible
- Split header O/E/O
  - electronic control
  - optical packet delay



Switches and Routers
Switch Fabric Architecture

Fast Packet Switch
Switch Fabric Architecture

- Connection state
  - simple per packet processing
- Switch fabric
  - eliminate blocking
  - no store-and-forward



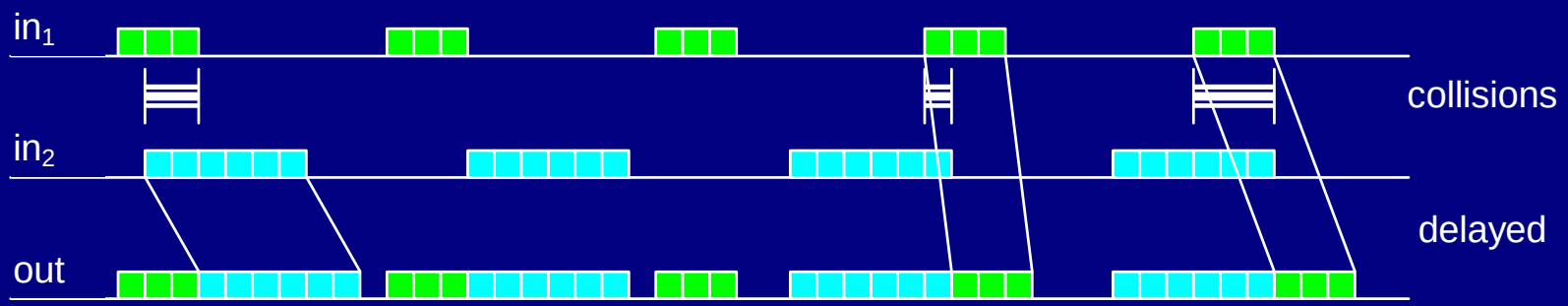
Switch Fabric Architecture
Blocking

- Blocking characteristics (among *different* outputs)
  - strictly nonblocking: under all conditions
  - wide-sense nonblocking: if particular algorithm is used
  - rearrangeably nonblocking: if existing paths are rearranged
  - virtually nonblocking: with extremely low probability

Nonblocking Switch Fabric Principle S-II.4*f*

A nonblocking switch fabric is the core or a high performance switch. Avoid blocking by space-division parallelism, internal speedup, and internal pipelined buffering with cut-through. Strict and wide-sense nonblocking avoids the complexity and delay of path rearrangement.

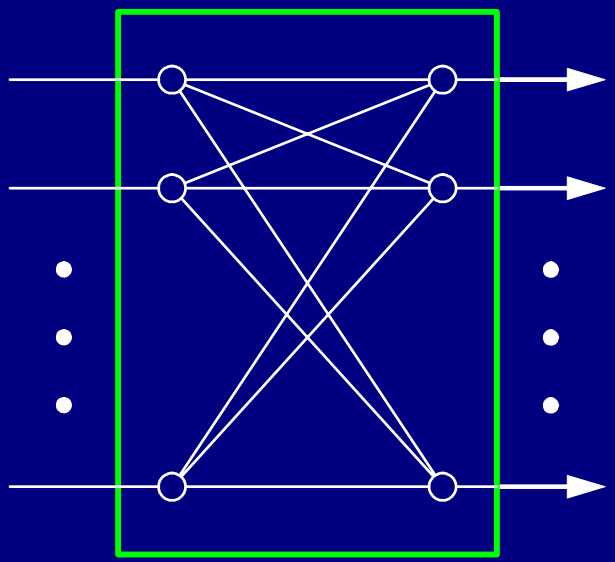
Switch Fabric Architecture
Contention and Buffering



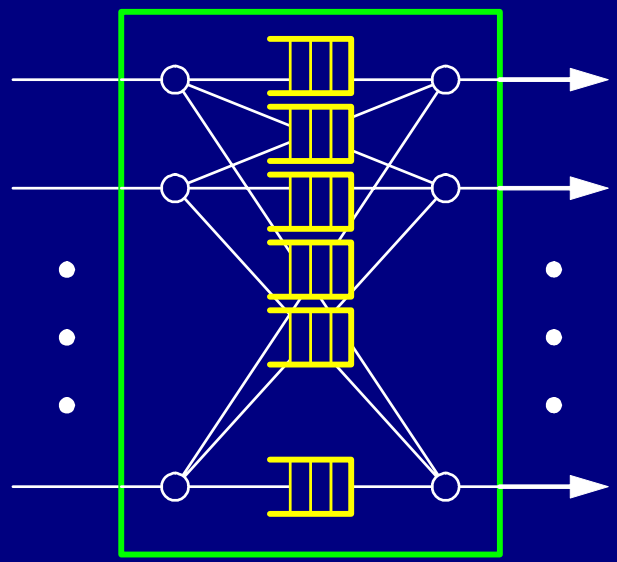
- Contention (burst collisions) in a non-blocking fabric
  - occurs when traffic destined for *same* output
  - requires buffering even for well-behaved traffic

Switch Fabric Architecture

Contention and Buffering

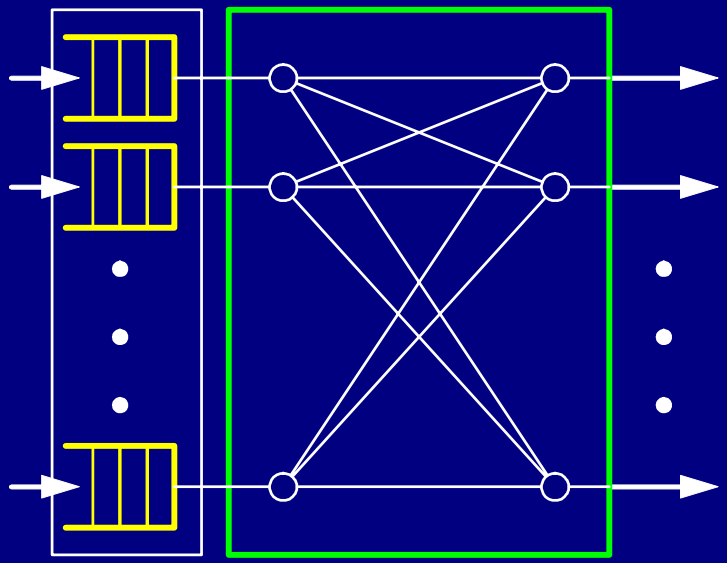


- Unbuffered fabric

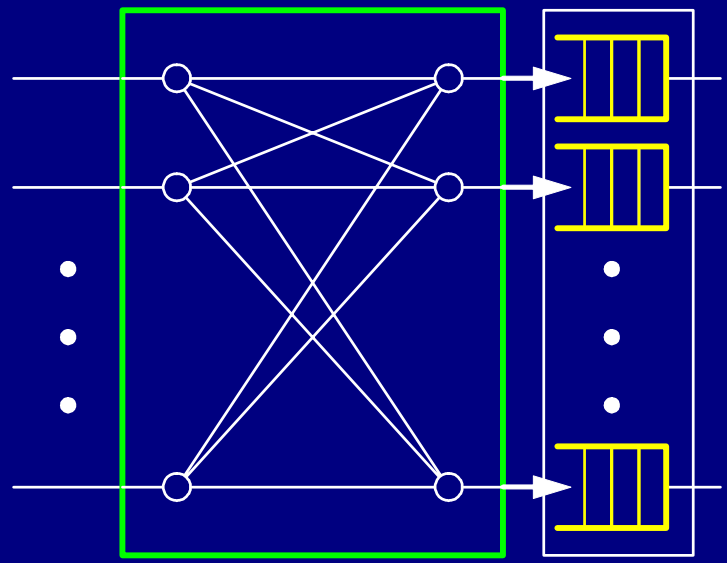


- Internal buffers

Switch Fabric Architecture
Contention and Buffering



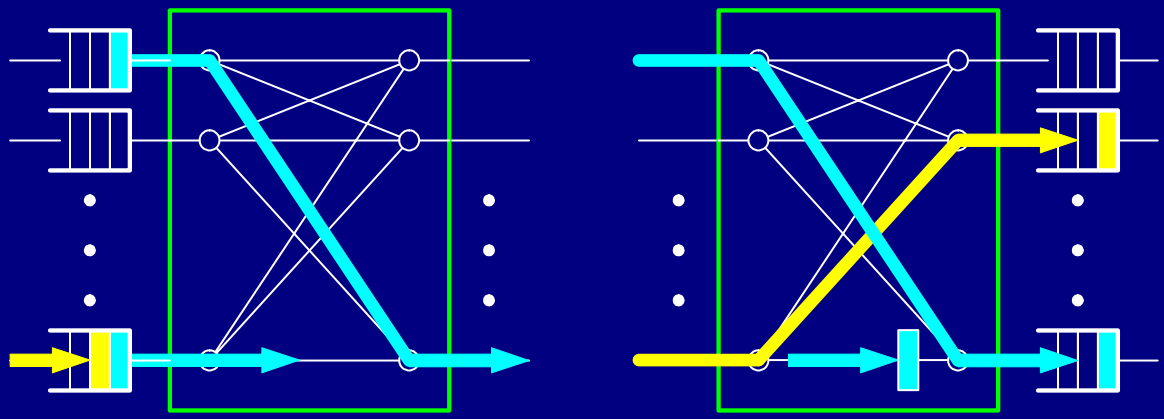
- Input queueing
  - suffers from head-of-line blocking



- Output queueing
  - requires either:
    - internal speedup
    - internal expansion

Switch Fabric Architecture

Head-of-Line Blocking

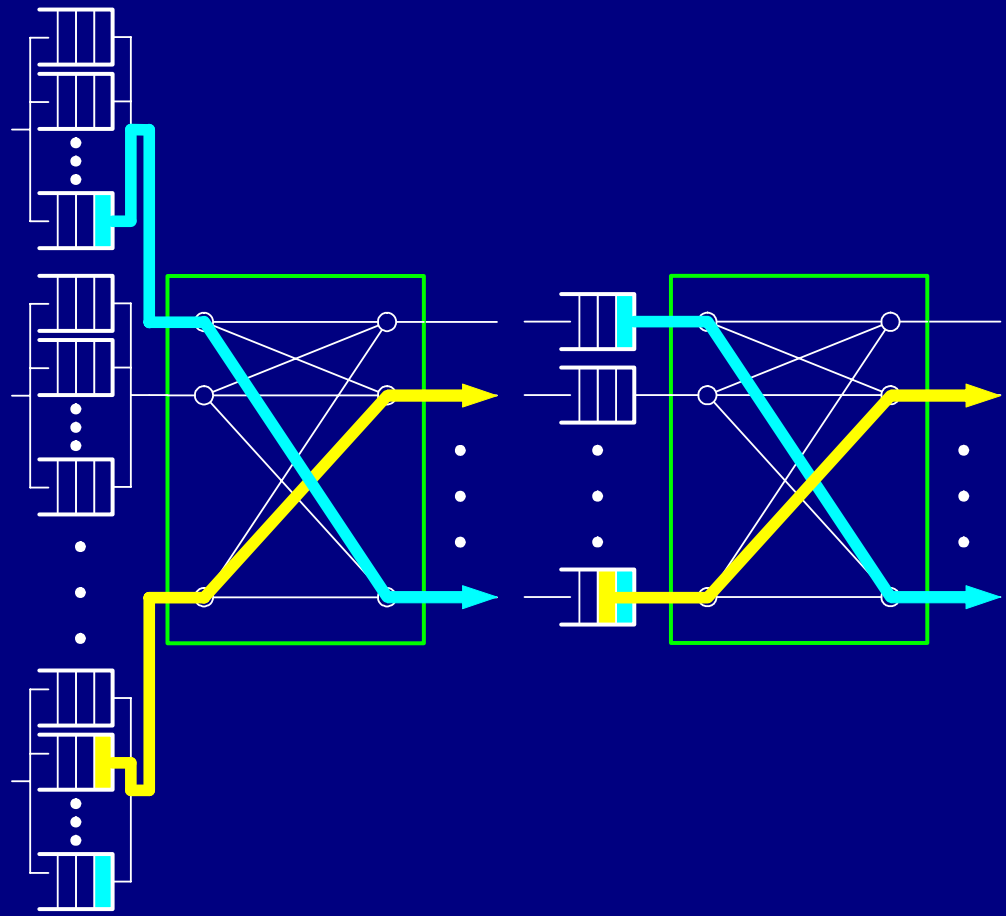


Head-of-Line Blocking Avoidance Principle S-II.4*q*

Avoid head-of-line blocking. Output queueing requires internal speedup, expansion, or buffering. Virtual output queueing requires additional queues or queueing complexity. The two techniques must be traded against one another, and can be used in combination.

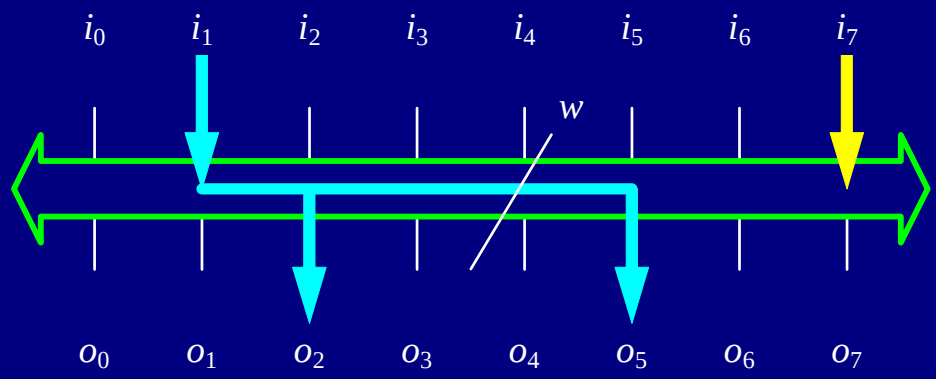
Switch Fabric Architecture
Virtual Output Queueing

- Virtual output queueing
  - parallel buffers
  - non-FIFO buffers



Switch Fabric Architecture

Single Stage: Bus as a Switch

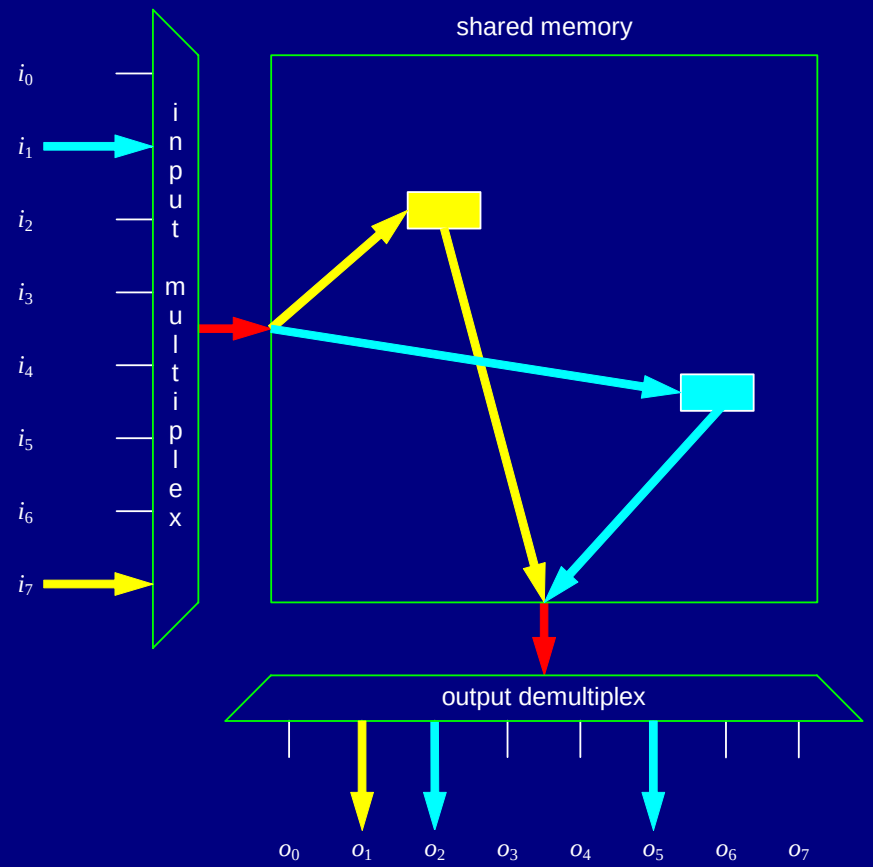


- Simple design shared medium bus
  - point of contention: only one input active at a time
    - 2nd/3rd generation routers
  - suitable for small switches
- Multicast
  - inherent broadcast

Switch Fabric Architecture

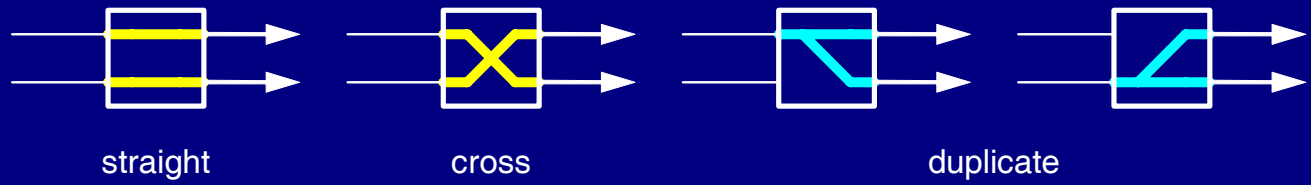
Single Stage: Shared Memory Switch

- Simple design
  - packets written by input
  - packets read by output
- Shared memory
  - point of contention
  - speedup necessary
    - but access times not scaling with Moore's
- Multicast
  - multiple writes or
  - multicast output demux

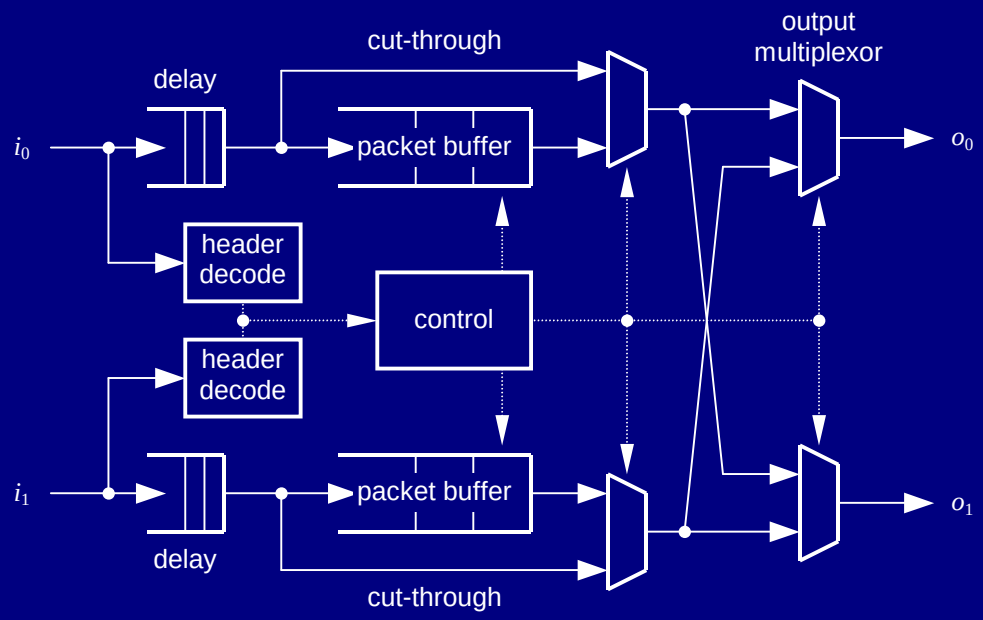


Switch Fabric Architecture

Single Stage: Basic 2×2 Switch Element

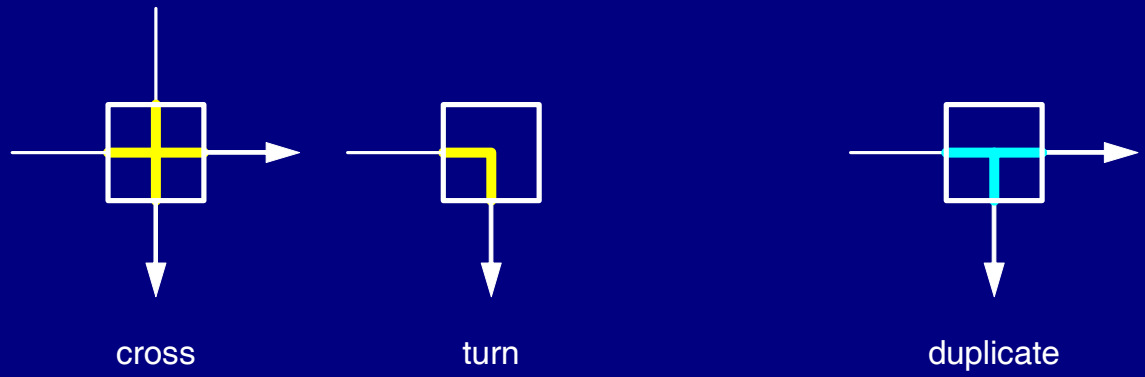


- States
  - point-to-point
    - straight
    - cross
  - multicast
- Types
  - buffered or unbuffered
  - self routing or externally controlled

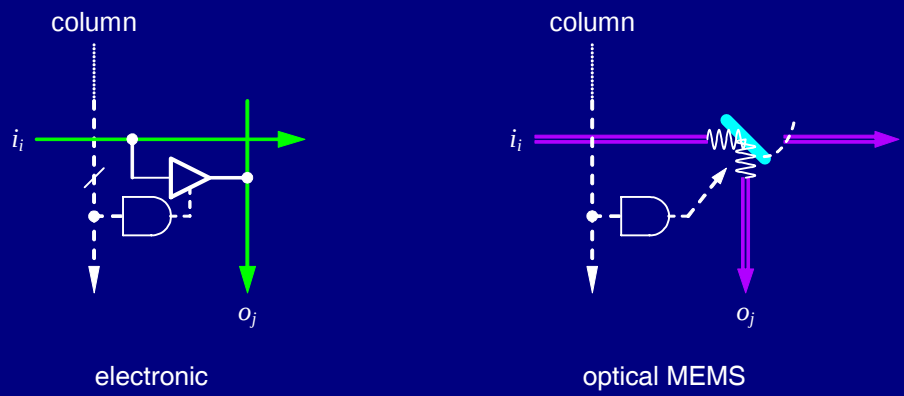


Switch Fabric Architecture

Single Stage: Crossbar Switch



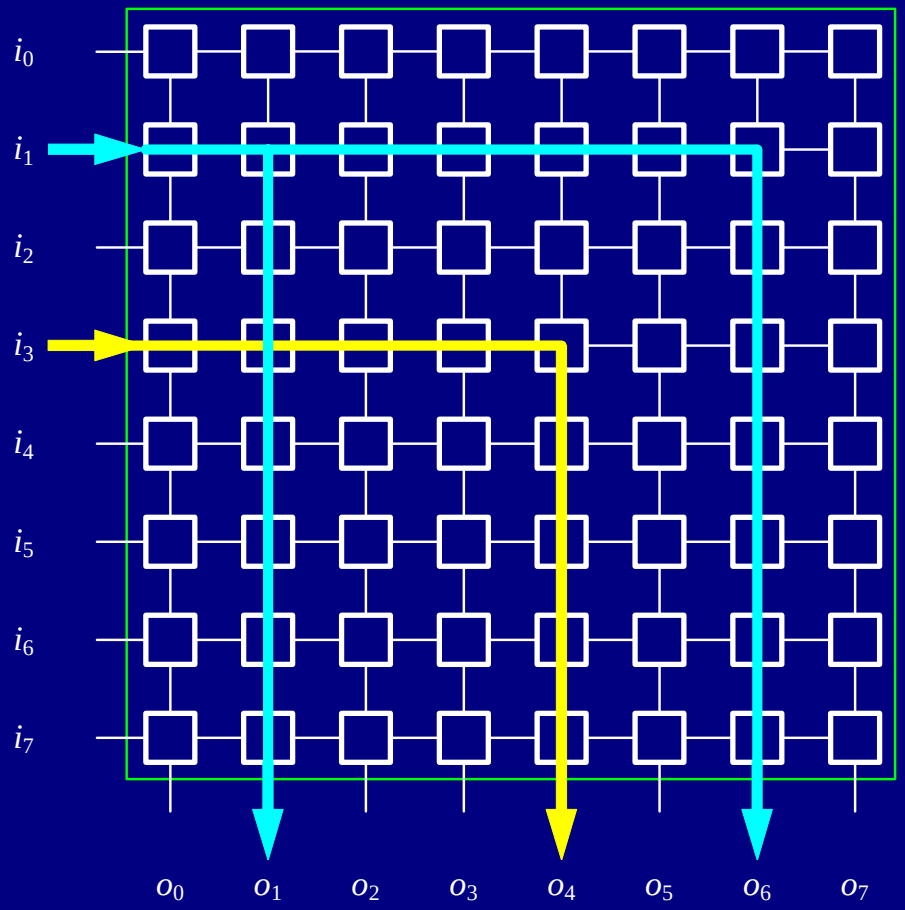
- Crosspoint switch element
  - electronic
    - multicast possible
  - optical MEMS
    - rotating mirror



Switch Fabric Architecture

Single Stage: Crossbar Switch

- Square array of crosspoint elements
  - $O(n^2)$ growth complexity
  - reasonable for moderate n
- Strictly nonblocking
- Multicast
  - inherent topology
  - requires arbitration



Switch Fabrics
Multistage Switches

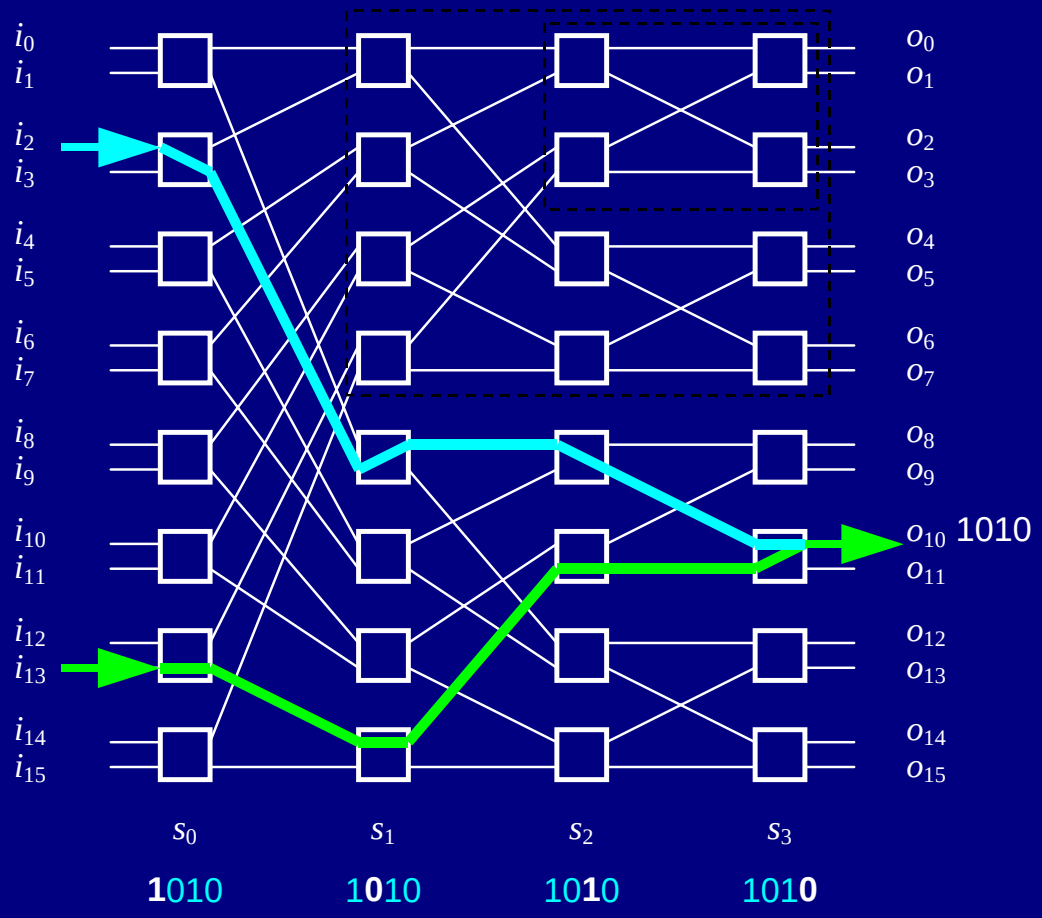
- Large switches built from single stage elements
  - 2×2 elements or $n \times n$ crossbars
  - $O(n \log n)$ growth complexity

Switch Scalability S-III.4

The construction of switches with a large port count requires scalable switch fabrics. Exploit regular structures for implementation in VLSI or optics as the basis for recursively constructed fabrics with logarithmic growth complexity.

Switch Fabrics
Multistage Switches

- Example
  - self-routing delta fabric



Switch Fabric Architecture
Native Multicast

- Multicast support in network conserves bandwidth
- Multicast support in switches
  - prevents multiple transmissions of a single multicast packet
  - reduces delay in subsequent packets

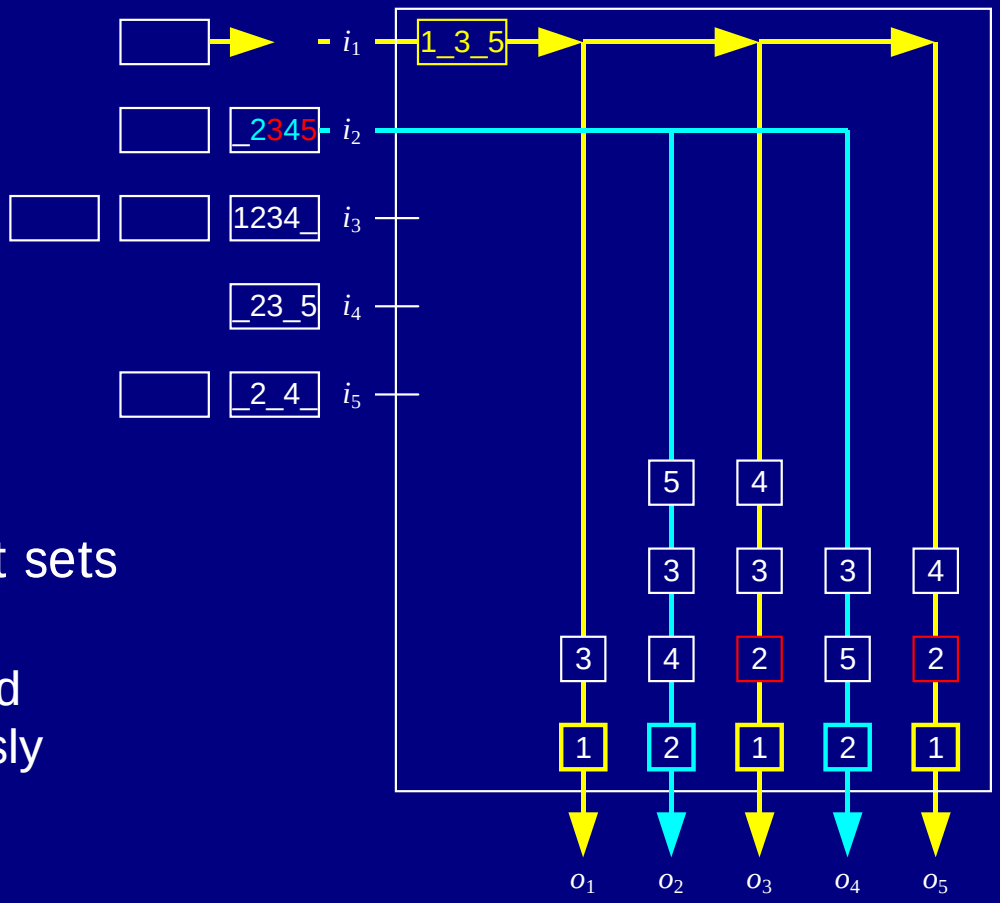
Switches Should Provide Native Multicast S-2C

Switches should provide native multicast support to conserve bandwidth and avoid latency of repeated transmission.

Switch Fabric Architecture

Native Multicast: Crossbar

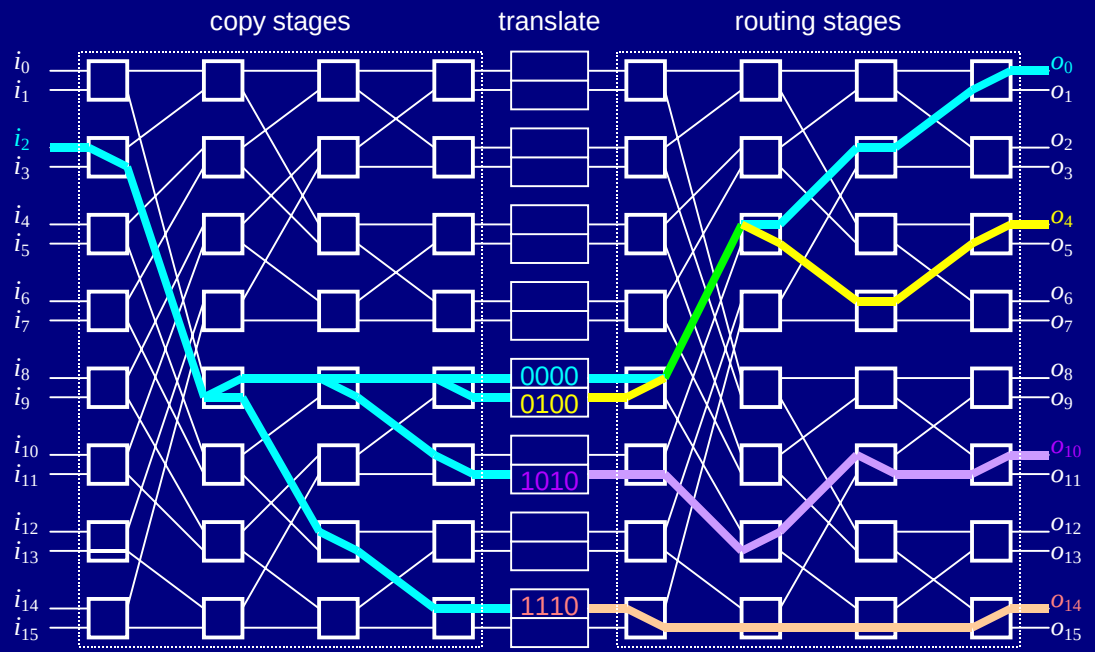
- Crossbar multicast
  - switch points duplicate
    - except MEMS mirrors
  - output arbitration needed to resolve incompatible output sets
    - fanout splitting: not all copies need xmit simultaneously



Switch Fabric Architecture

Native Multicast: Multistage

- Multistage multicast
  - switch elements duplicate late
    - 2×2: replicate state
    - $n \times n$ internal
  - alternative: recirculation
  - multiple fabrics
    - copy stages
    - c_{id} translation
    - routing stages



Switches and Routers
Fast Datagram Switches

Fast Datagram Switches
Motivation

- Connection-oriented fast packet switching
  - emerged in ATM standards, but ATM failed
- IP became waist of global network infrastructure
  - increased processing capability enabled fast IP lookups
  - apply fast packet switching to IP datagram forwarding

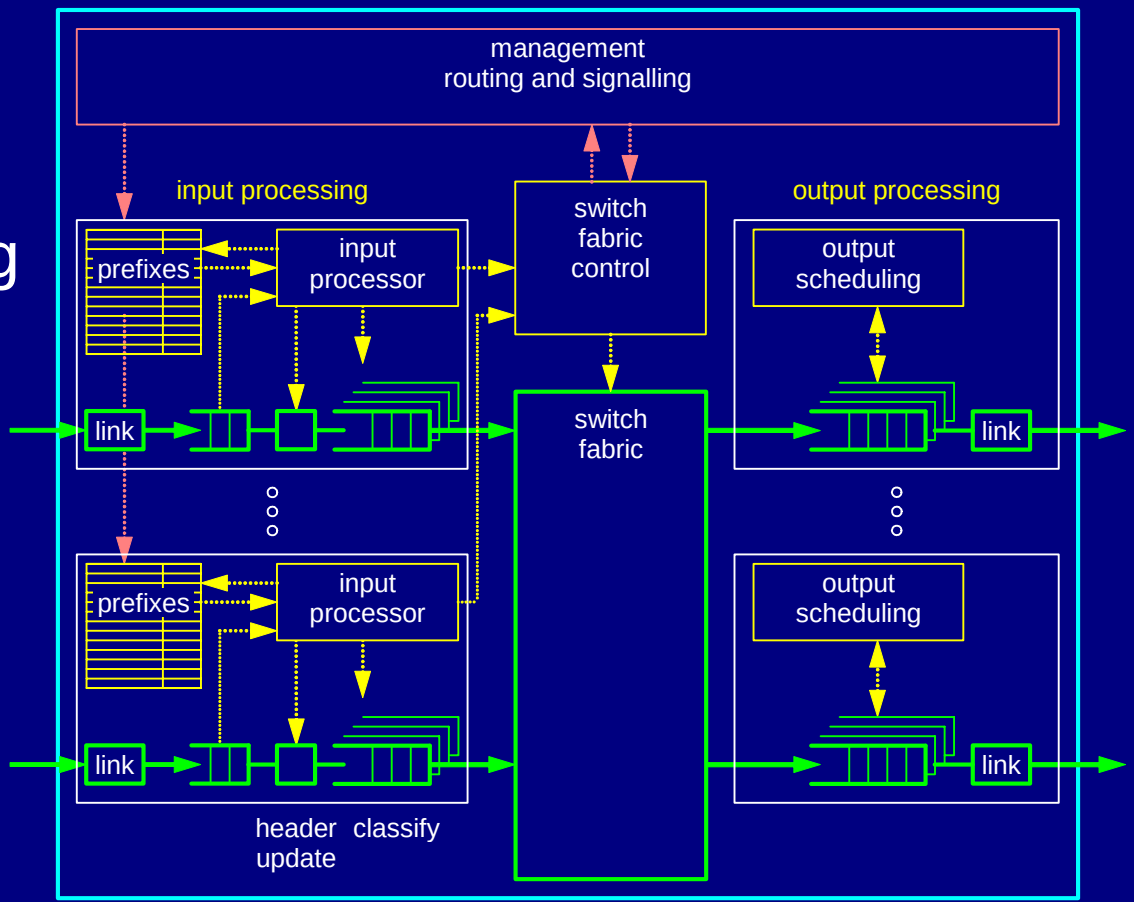
Fast Datagram Switch Principle

S-1B*d*

There is compelling reason to perform high-speed connectionless datagram switching. Apply connection-oriented fast packet switching techniques to fast datagram switching, and exploit technology advances to implement the necessary additional input and output processing in the critical path.

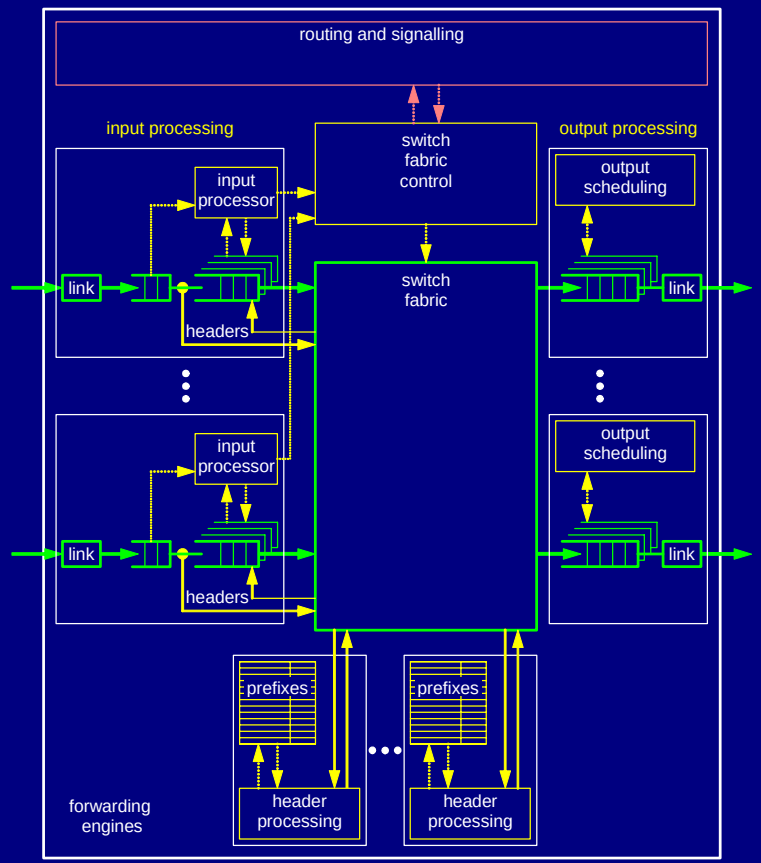
Fast Datagram Switches
Architecture

- Fast packet switch core
- Input processing
  - IP lookup
  - packet classification
- Output processing
  - packet scheduling
    - fair queueing



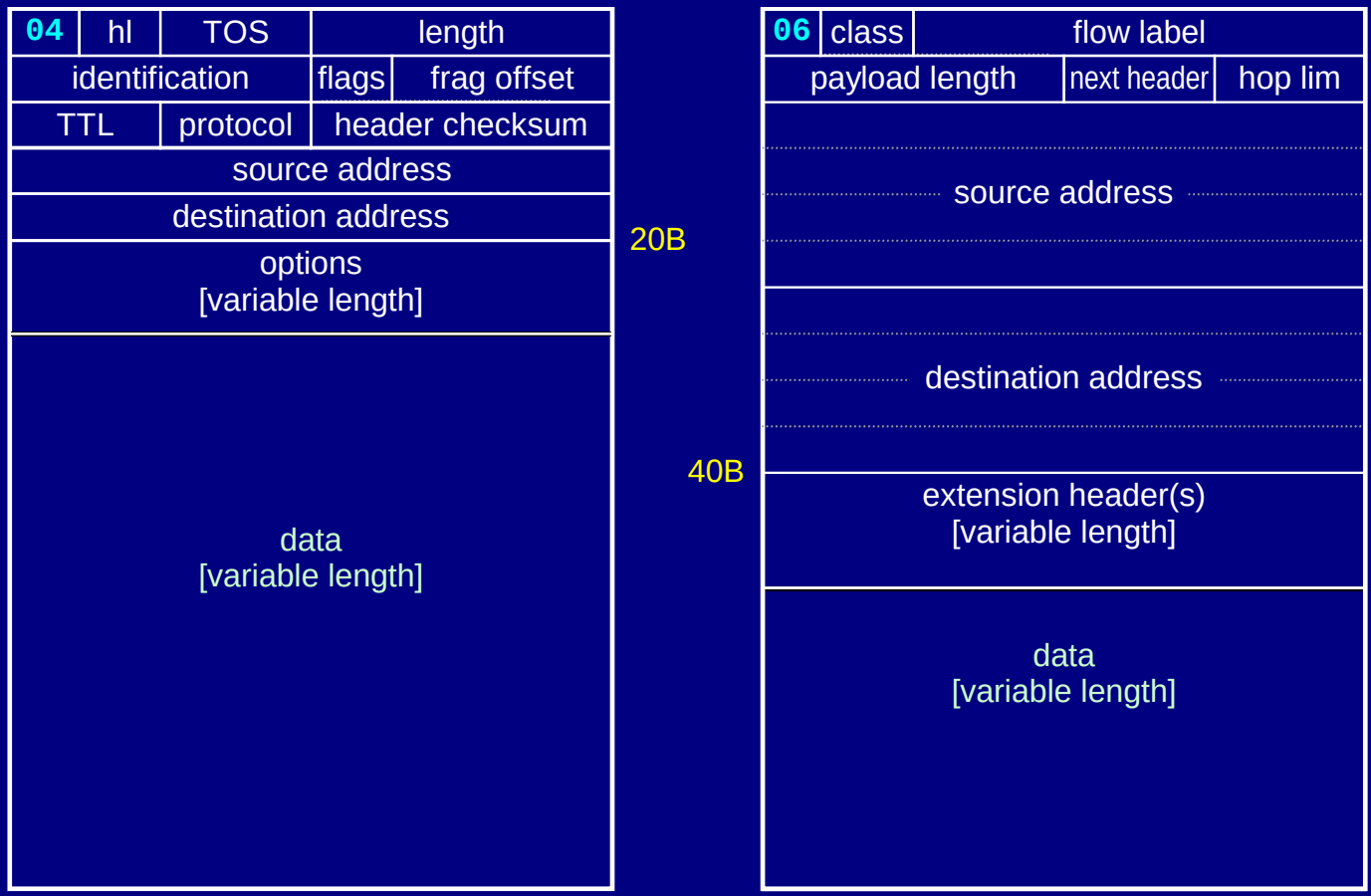
Fast Datagram Switches
Architecture: Shared Forwarding

- Fast packet switch core
- Input processing
  - packet classification *must* be performed at input
- Shared forwarding engines
  - flexible allocation, but
  - uses switch fabric ports and bandwidth
- Output processing
  - packet scheduling
    - fair queueing



Datagram Routing and Switching

Example 5.6 IP Packets



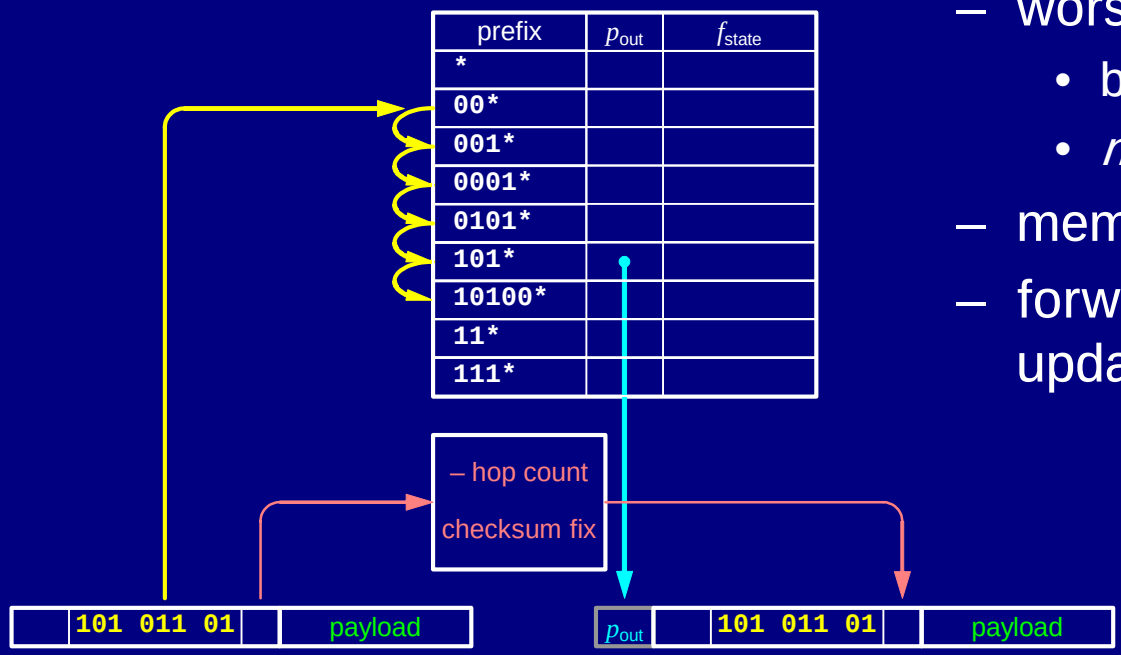
Fast Datagram Switches
Throughput

- Packet processing rate critical
  - packet processing must sustain at least average rate
  - critical path must sustain peak line rate for min size packets

Packet Processing Rate S-II.4*p*

The packet processing rate (packets per second) is a key throughput measure of a switch. Packet processing software and shared parallel hardware resources must be able to sustain the average packet processing rate. Functions in the serial critical path must be designed for the worst case packet processing rate of the path to avoid queueing and blocking of subsequent packets.

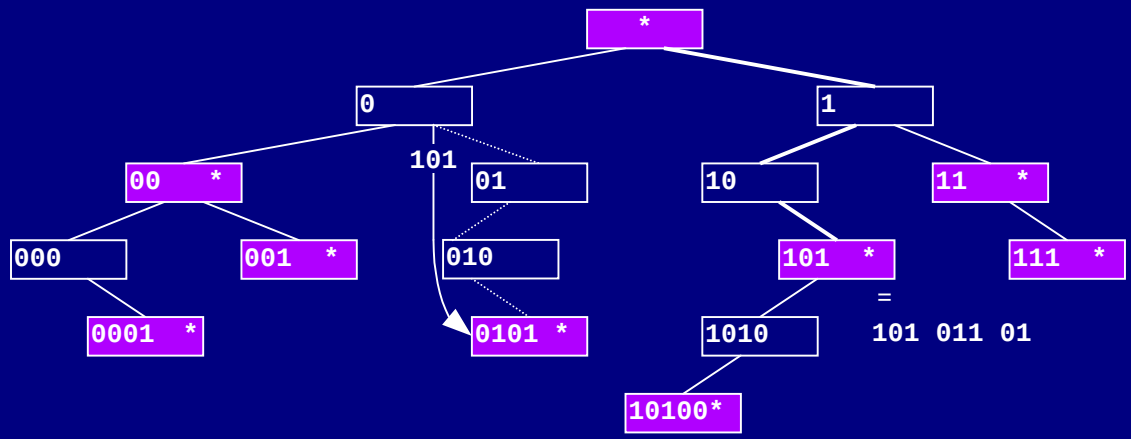
Fast Datagram Switches
Software IP Lookup



- Longest prefix match
- Critical parameters
  - worst case lookup time
    - brute force: $O(\log_2 n)$
    - n tens of thousands
  - memory required
  - forwarding table update time

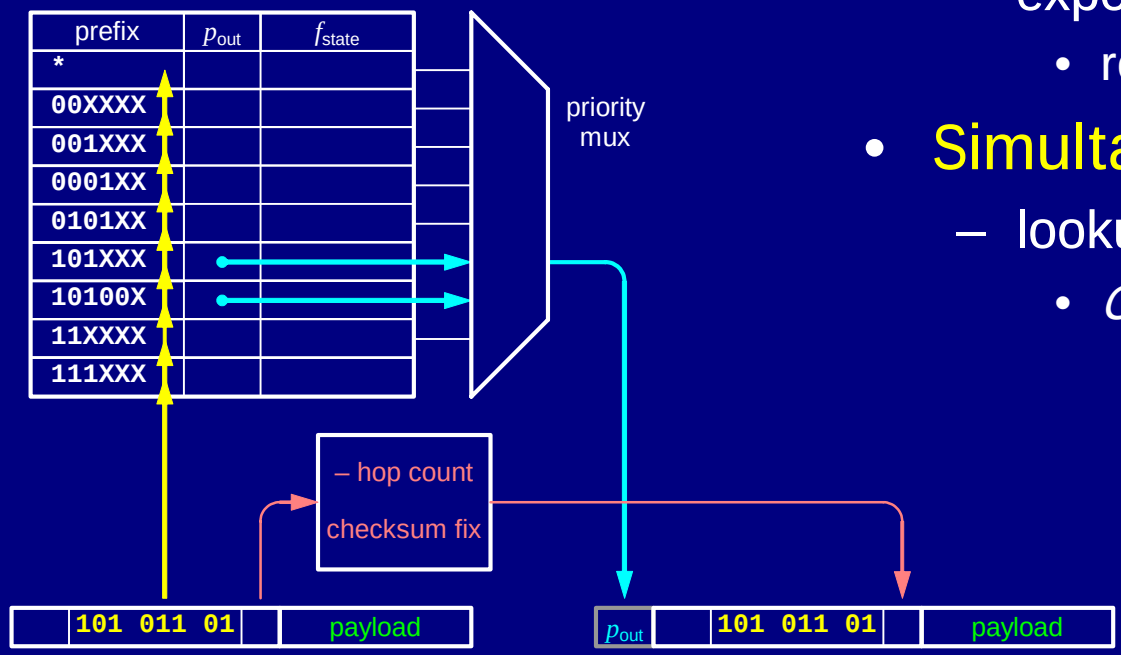
Fast Datagram Switches

Software IP Lookup Example: Trie



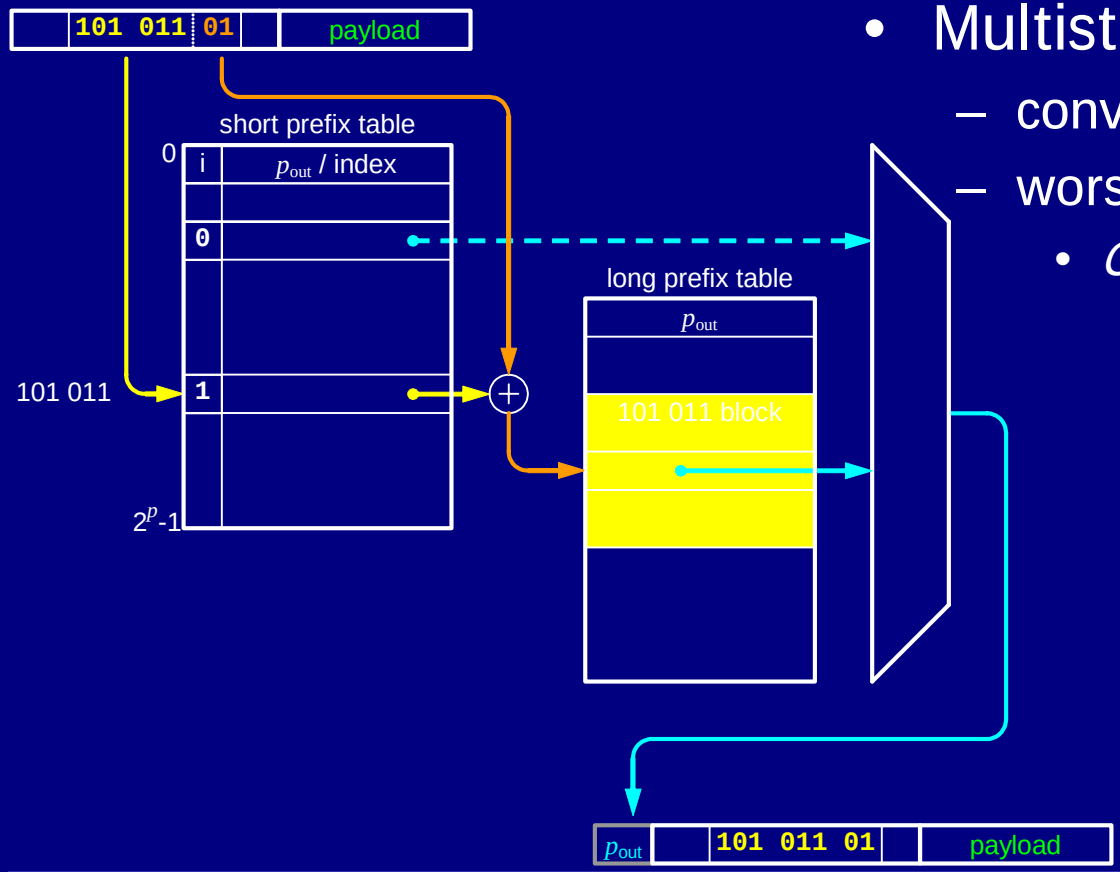
- Many algorithms
- Example: trie
  - sparse binary tree
  - valid prefixes are root
  - lookup time $O(a)$
    - a = number of address bits

Fast Datagram Switches
Hardware IP Lookup



- Ternary CAM
  - **1**, **0**, **X** (don't care)
  - expensive and complex
    - relative to RAM
- Simultaneous match
  - lookup time constant
    - $O(1)$

Fast Datagram Switches
Hardware-Assisted Memory IP Lookup



- Multistage lookup
  - conventional SRAM
  - worst case lookup time
    - $O(s)$ number of stages

Fast Datagram Switches
Hardware vs. Software

- Input and Output processing tradeoff
  - custom hardware generally faster
  - network processor software more flexible

Hardware vs. Software Implementation of Input and Output Processing S-1C*h*

In determining the appropriate implementation of input and output processing, trade the cost and feasibility of hardware against the complexity and feasibility of network processor software.

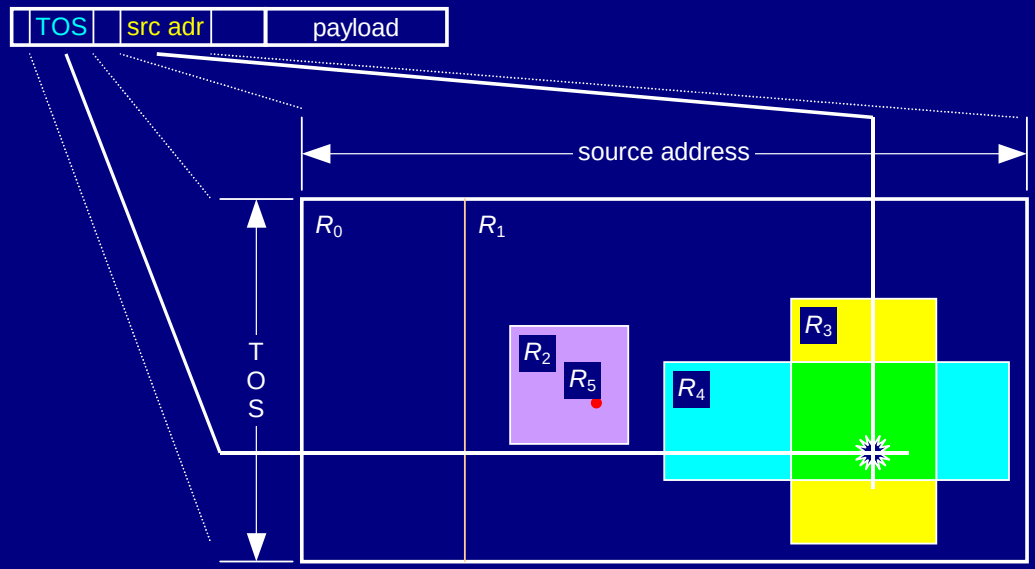
Fast Datagram Switches
Packet Classification

- Packet classification determines how packet treated
  - QOS or diffserv
  - policy based routing
  - security and DOS protection (e.g. firewalls)
  - layer 4 and 7 switching
  - active network processing
- Before queueing to meet most stringent delay class

Bound Packet Classification Time S-II.4*c*

Packets that must be classified to potentially receive delay bounded service must be classified before any queueing at the input. The classification operation must have delay bounds that meet the most stringent service class.

Fast Datagram Switches
Packet Classification



- Multidimensional classification
  - policies may be hierarchal or overlap
  - precedence rules needed
- More complex than longest prefix match
- Hardware and software implementation tradeoffs

Fast Datagram Switches
Output Scheduling

- Output scheduling
  - guaranteed service packets transmitted to meet contract
  - fair service among best effort flows
- Fair queueing
- Per-flow queueing
  - isolates flows from one another

Output Scheduling and Granularity S-II.4*s*

Output scheduling must operate at line rate to support the traffic classes required. Finer-grained queueing and scheduling providers greater isolation of flows and control a the cost of increased queueing complexity.

Fast Datagram Switches
Congestion Control

- Keep queues from building
  - impacts entire port unless per flow queueing
  - bound steady state size of queues
  - throttle transmitter when necessary
    - RED: random early detection
    - ECN: explicit congestion notification

Discard to Keep Queues from Building S-6C*d*

Queue length should reflect transient traffic conditions, and not be allowed to build in the steady state. Perform congestion avoidance and control, including packet discard, to keep queues small locally, and throttle the source for end-to-end congestion avoidance.

Switches and Routers
Programmable and Higher Layer Processing

Switch Implementation
Hardware Alternatives

- Alternative implementation technologies
  - hardware (fast but inflexible)
    - custom VLSI vs. semicustom vs. standard
    - technology
      - DRAM vs. SRAM vs. CAM
      - CMOS vs. GaAs
    - electronic vs. optical

Functional Partitioning and Assignment Principle

Carefully determine what switch functionality is to be implemented in scarce or expensive technology. S-1C

Switch Implementation
Programmable Alternatives

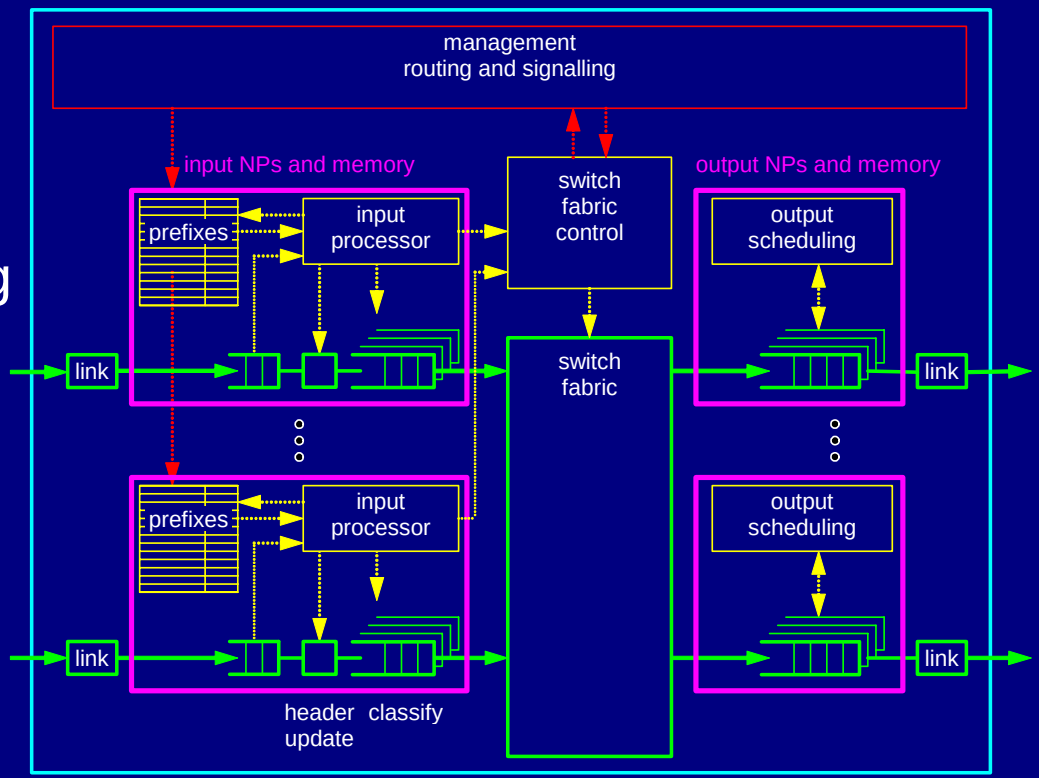
- Alternative implementation technologies
  - fixed hardware (fast but inflexible)
  - programmable hardware (e.g. FPGAs)
  - software: flexible
    - switch control CPU
    - general purpose embedded controller
    - *network processor*:
      specialised embedded controller for packet processing

Switch Implementation
Programmable Input/Output Processing

- Network processors
  - compromise among cost, flexibility, performance
- Advantages of input/output programmability
  - reduce switch design and debug time
  - allow field upgrades
  - allow service providers to deploy new protocols/services
  - enable per port active networking

Switch Implementation
Programmable Input/Output Processing

- NPs replace
  - input processing
    - lookup
    - classification
  - output processing
    - scheduling
  - additional
    - encryption
    - policing
  - large buffers
    - off NP chip



Network Processors

Examples

	Interface Capacity	Packet Processors	Co-processors	Control Processor
Intel		μengines		ARM
IXP 1200	OC-12	6 @ 166 MHz 4K × 40b I mem	hash	StrongARM 232 MHz 8MB + 256MB+ external
IXP 2400	OC-48 14Mpps 40B	8 @ 600 MHz 4K × 40b I mem pipeline interconnection	hash	Xscale 600MHz 32KI$ [32+2]KD$ 2GB+32MB external
IXP 2850	OC-192 60Mpps 40B	16 @ 1.4 GHz 4K×40b I+996w D mem pipeline interconnection	hash crypto	Xscale 700MHz 32KI$ [32+2K]D$ Ext 6GB+64MB ext
C-Port Motorola C-5	4 × OC-12 15Mpps	16 @ 266 MHz 64KI + 12KD mem pipeline/parallel	queue, buffer, fabric, table lookup	XP 266 MHz 32KI + 32KD mem
IBM PowerNP 4GS3	OC-48 4.5Mpps	8 × 2 @ 133 MHz picoprocessors	8×10 e.g. tree search, checksum	PowerPC 405 133 MHz 16KI$ + 16KD$

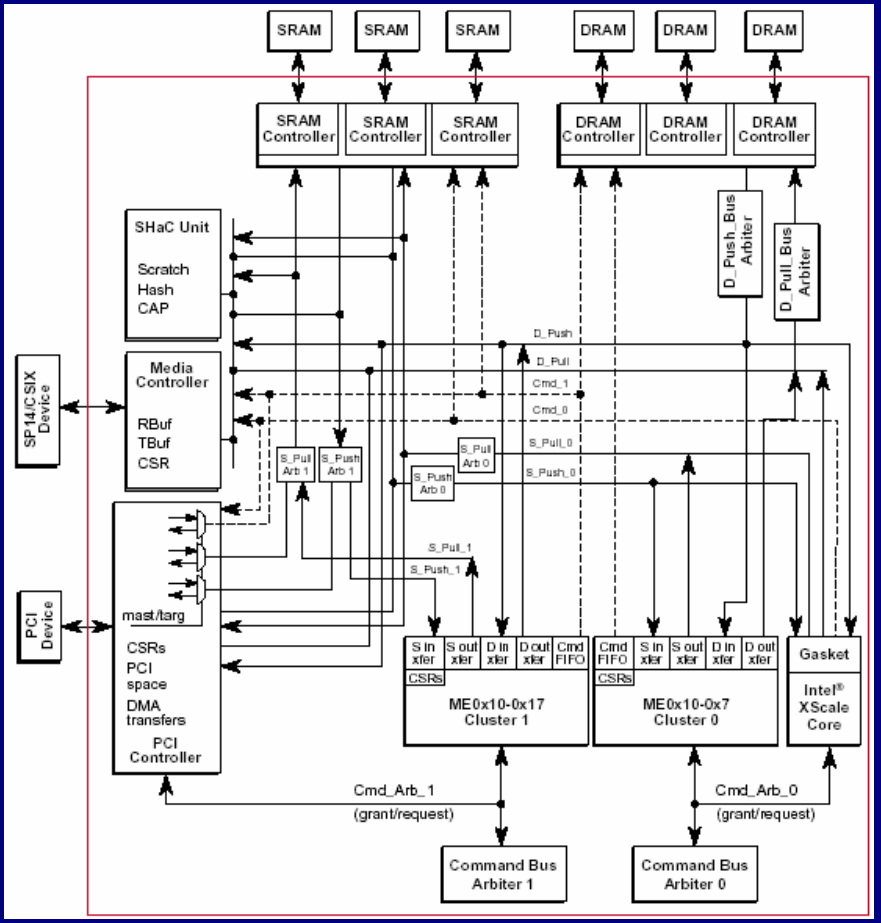
others are proprietary and require NDA for details

Network Processors
Example: Intel IXP 2800/2850

- Moderately specialised embedded controller
  - organised for packet processing
  - XScale ARM core controller + 16 microengines
  - some functional units
    - hash, crypto (AES, DES, SHA-1 on 2850)
    - CRC, pseudorandom generator per microengine
- Programming
  - requires *significant* knowledge of hardware organisation

Network Processors
Example: Intel IXP 2800/2850

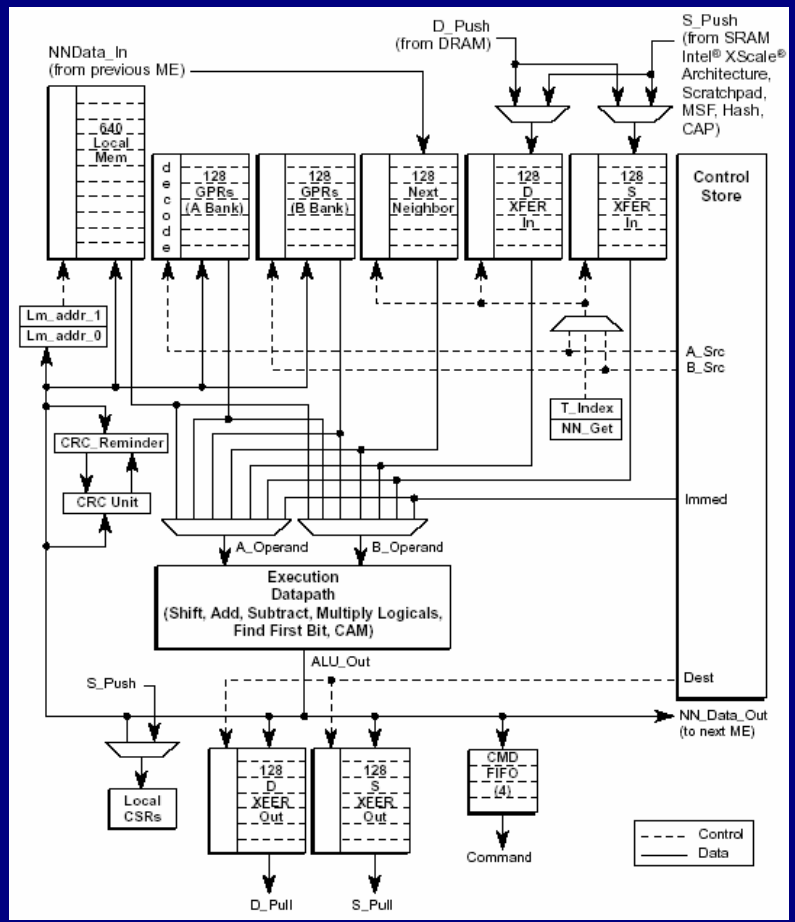
- Processors
  - XScale core @ 700MHz
  - 16 microengines
- 16KB scratchpad mem
- Controllers
  - PCI
  - SRAM, DRAM
- Media/switch interface
- Hash unit
- Crypto (2850 only)



Network Processors

Example: Intel IXP 2800/2850 Microengine

- 16 @ 1.4 GHz
- 2 × 128 GPRs (A,B)
- ALU (A • B)
- 640 longword local mem
- {S|D}RAM regs
- 16 × 32+4b CAM
- CRC-{16,32} unit

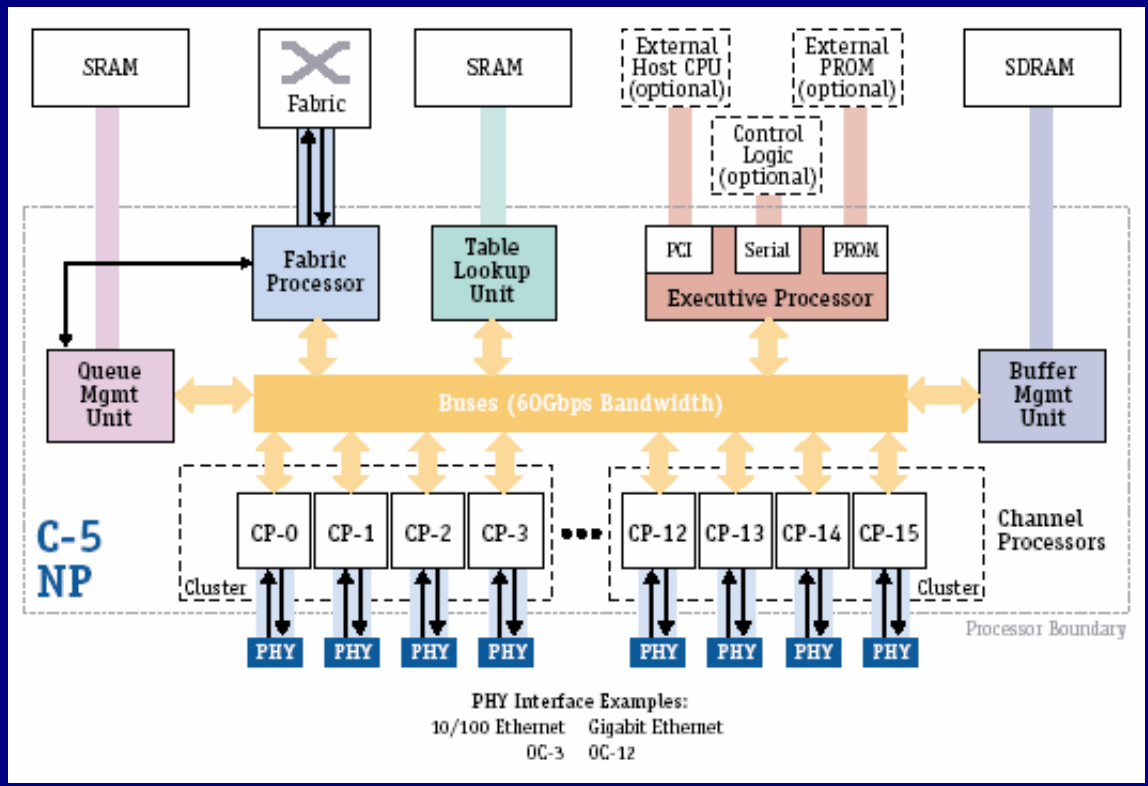


Network Processors
Example: C-Port Motorola C-5

- Moderately specialised embedded controller
  - organised for packet processing
  - XP (executive processor) MIPS-1 core
  - 16 CPs (channel processors) MIPS-1
  - some functional units
    - table lookup and classification engine
    - fabric processor and interface
    - queue management unit interface to 128 Kword SRAM
    - buffer management unit interface to 128 MB SDRAM
    - external interface to traffic management coprocessor
- Programming
  - requires *significant* knowledge of hardware organisation

Network Processors
Example: C-Port Motorola C-5

- Processors
  - XP MIPS-1
    - 266 MHz
  - 16 CPs
    - 266 MHz
- Coprocessors
- Interfaces
  - PCI
  - S{D}RAM
  - fabric



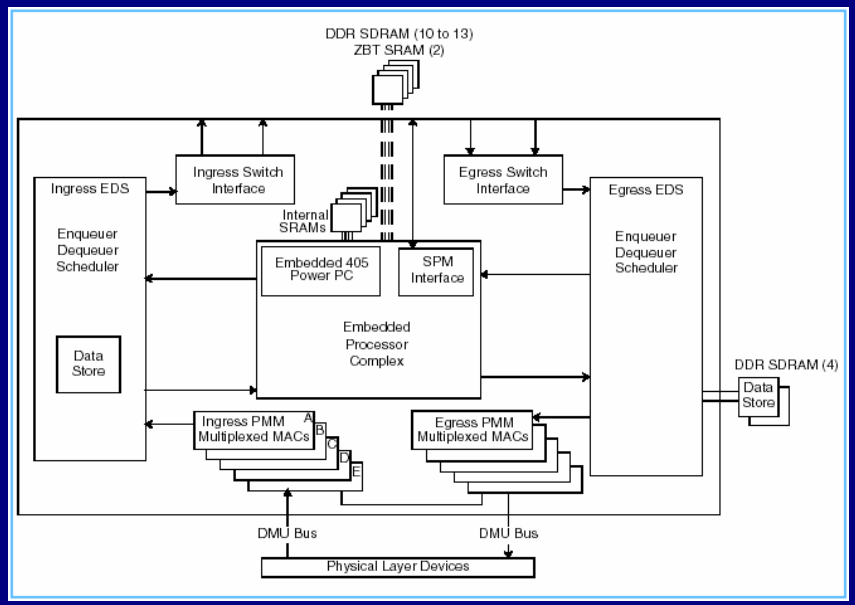
Network Processors
Example: IBM PowerNP 4GS3

- Highly specialised embedded controller
  - organised for packet processing; many functional units
    - PowerPC 405 core @ 133 MHz
    - completion unit, dispatch unit, control store arbiter enqueue/dequeue unit, traffic scheduler/shaper
    - 8 DPPUs (dyadic protocol processor units)
      - 2 picoprocessors each @ 133 MHz
      - 10 coprocessors: checksum, bus control, ext coprocessor control, counter, data store, enqueue, policer, string copy, tree search, semaphore manager
- Programming
  - requires *significant* knowledge of hardware organisation

IBM has sold the NP to Hifn – datasheets and manuals no longer online

Network Processors
Example: IBM PowerNP 4GS3

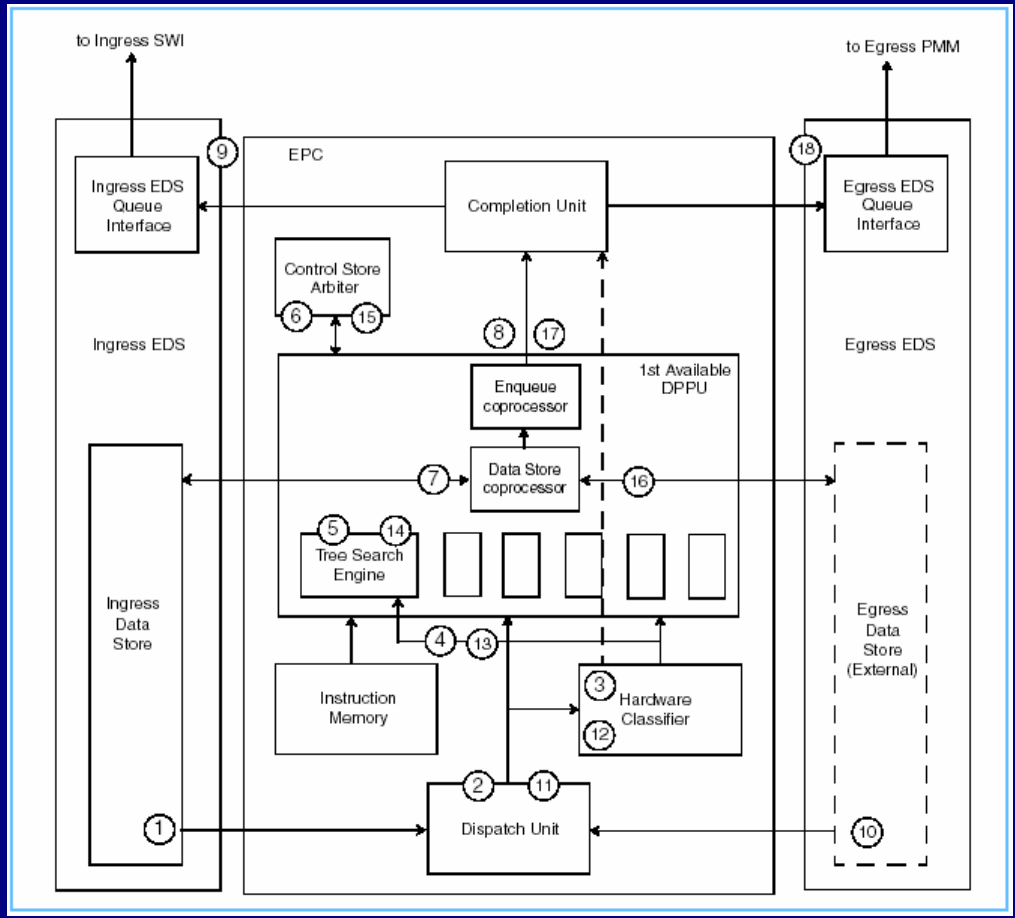
- Processors
  - PowerPC @ 133 MHz 16K I + 16K D cache
  - 8 DPPUs
- Memory
  - 32 KB instruction
  - 128 KB SRAM buffer
  - 113 KB SRAM ctl store
- Interface controllers
  - switch ingress+egress
  - SRAM, SDRAM
  - PCI



- Coprocessors/accelerators
  - traffic scheduler/shaper
  - enqueue/dequeue
  - completion, dispatch, ctl store

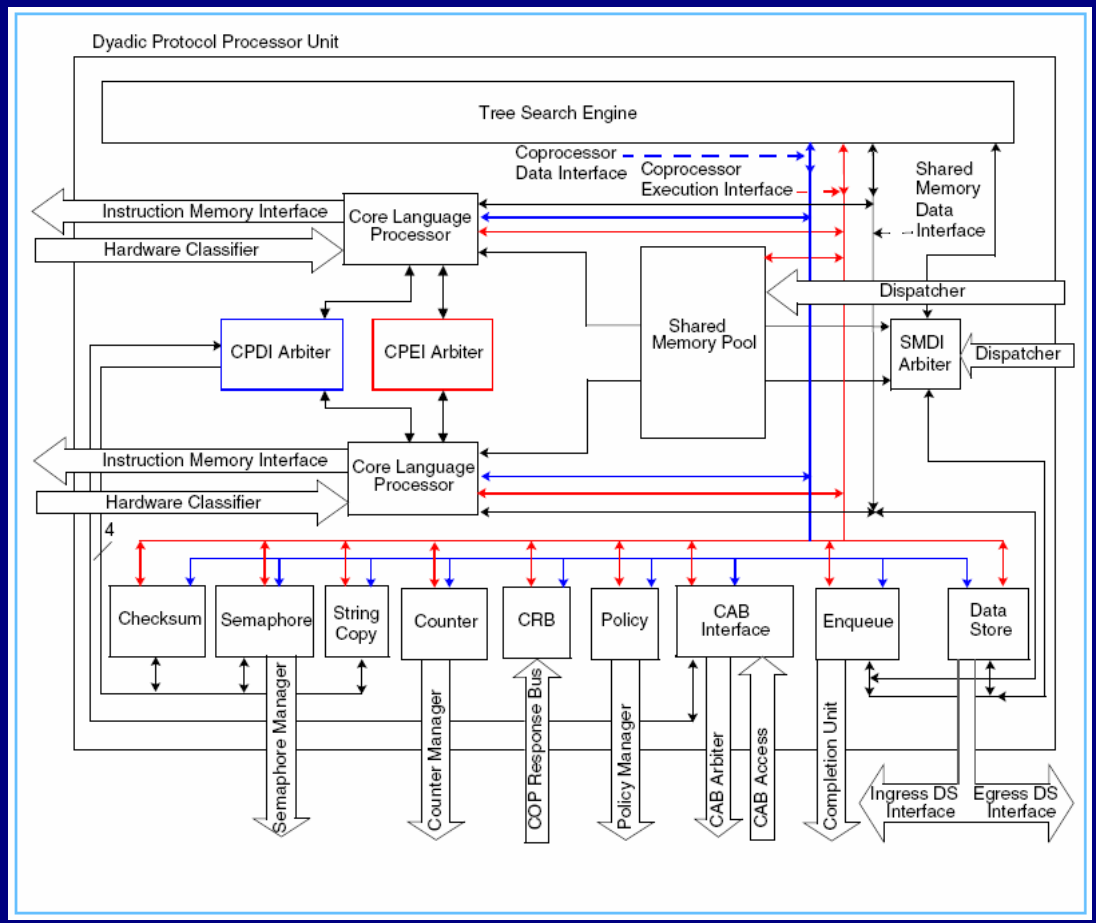
Network Processors
Example: IBM PowerNP 4GS3 EPC

- 8 DPPUs
  - 2 picoprocessors
    - 133 MHz
    - 16|32 GPRs
    - ALU
  - 10 coprocessors
- 8 × 4KB memory



Network Processors

Example: IBM PowerNP 4GS3 DPPU



Higher Layer and Active Processing
Motivation and Strategies

- Example motivation for processing in network nodes
  - firewalls
  - Web caches
  - active transcoders
  - multimedia stream mixing and merging
- Alternative strategies
  - application layer proxies
  - active network nodes
  - layer 4 and layer 7 switches
  - integration of function into layer 3 forwarding code

Higher Layer and Active Processing
Performance

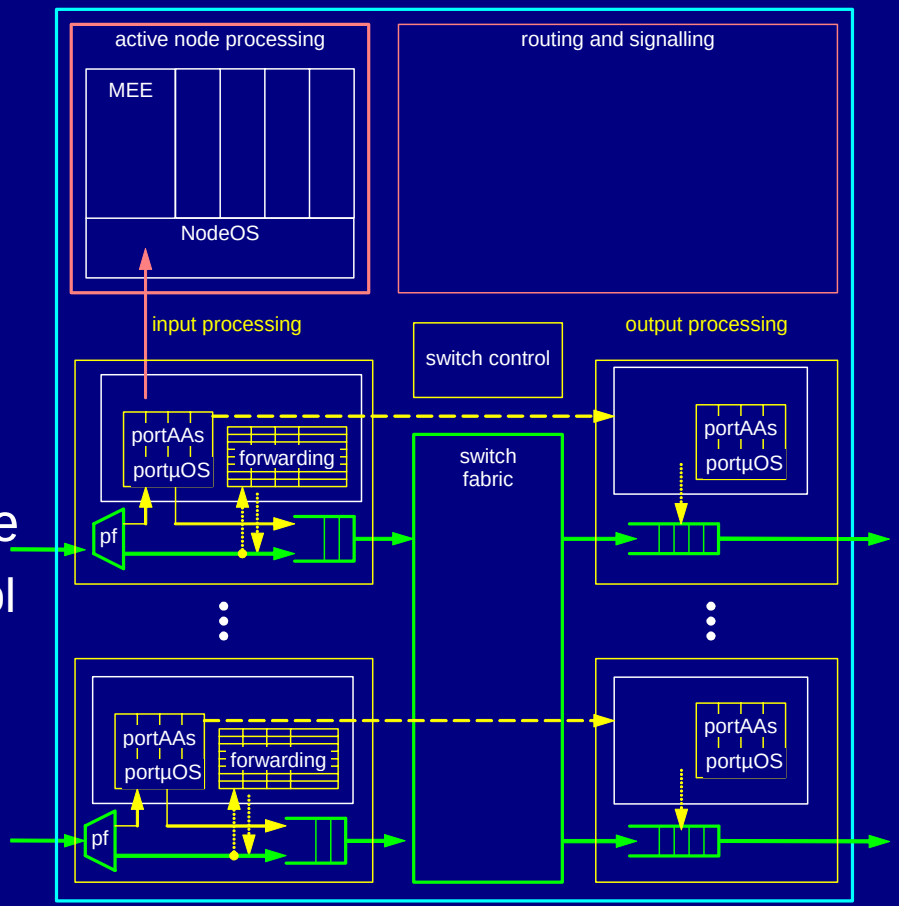
- Active processing reasonable tradeoff
  - performance vs. flexibility
- Active processing must
  - not impede the normal fast forwarding path
  - occur with enough throughput so active queues don't build

Active Network Processing S-II.4*a*

Active network processing should not impede the non-active fastpath; packet filters in the critical path must operate at line rate to pass non-active packets. The ability to perform per packet active processing requires sufficient processing power to sustain the require active packet throughput

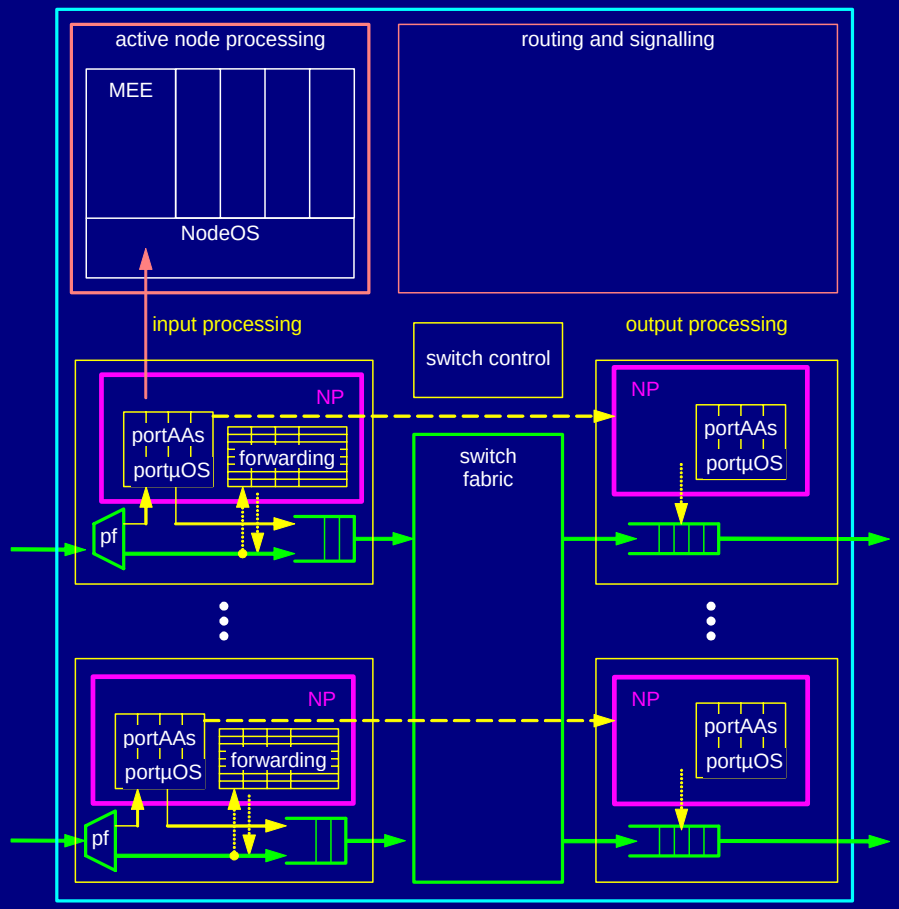
Active Processing
Granularity and Node Architecture

- Processing granularity
  - global control plane
    - e.g. routing algorithm
  - per flow control plane
    - e.g. modify RSVP control
  - per packet control plane
    - e.g. congestion control
  - per packet data plane
    - e.g. transcoding



Active Processing
Network Processors

- Network processors
  - enable per port dynamic programmability



Network Processors
Future Prospects

- Future prospects of open COTS NPs in doubt
  - IBM exited and Intel may exit market
  - high end market controlled by single switch vendor: Cisco
- Programmable network prospects
  - tied directly to deployment of open COTS NPs
  - therefore programmable net future seriously threatened