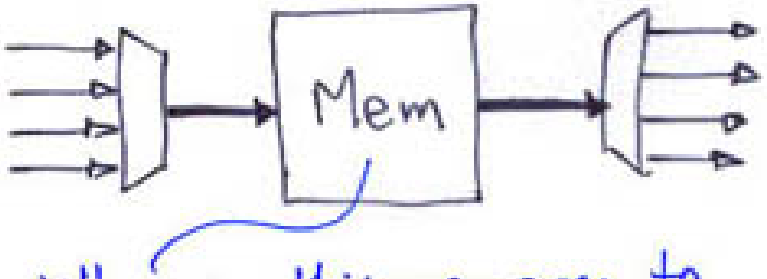
3.1 The need for multiple queues within a same buffer memory: **Multi-Queue Data Strucutres**

- Buffers memories for time switching: what data structures?
- Single queue feeding multiple destinations/classes

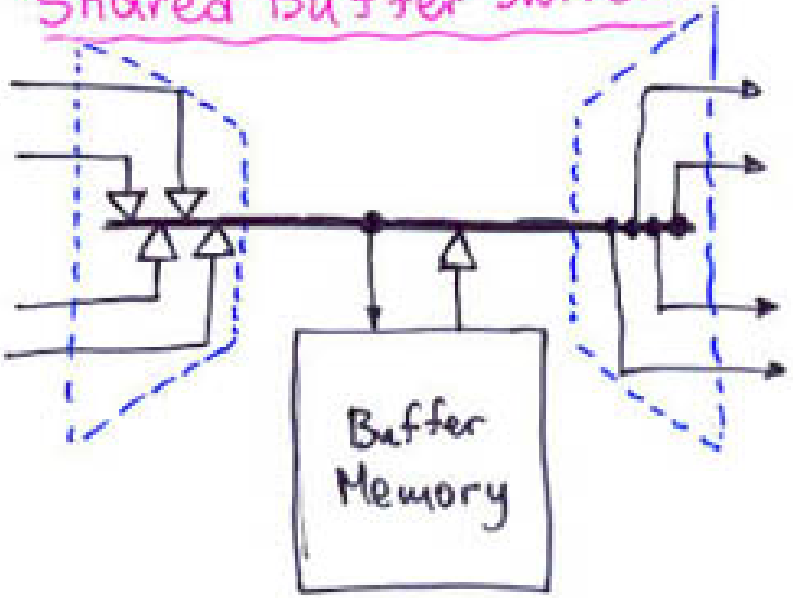
  $\Rightarrow$ Head-of-Line (HOL) Blocking $\Rightarrow$ poor performance
- Multiple Queues in one buffer:
  - Partitioned Space (underutilized) $\Rightarrow$ circular queues
  - Shared Space (efficient) $\Rightarrow$ linked-list queues

Time Switching:

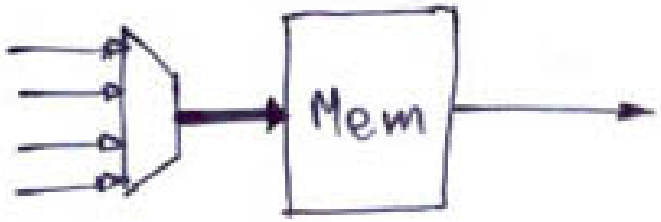


Where in this memory to store packets? (which address?)

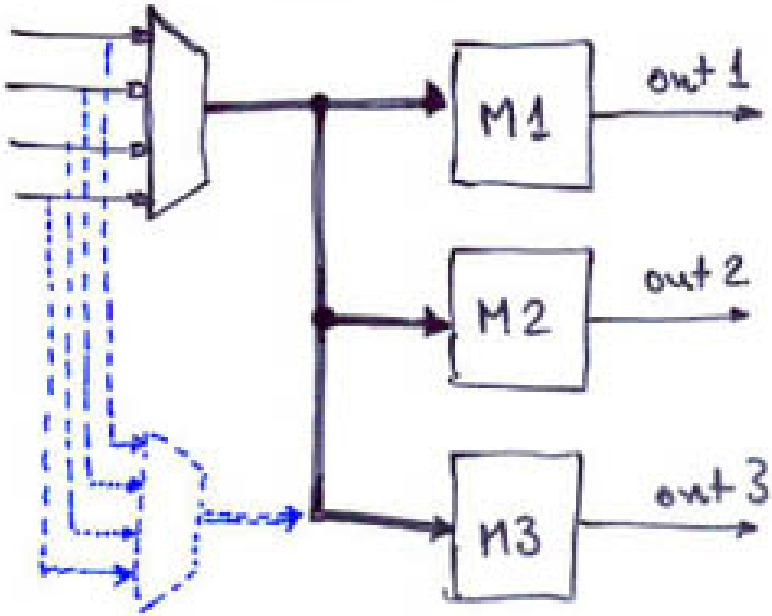
"Shared Buffer Switch":



Simplification: n x 1 Switch "Output Queueing"

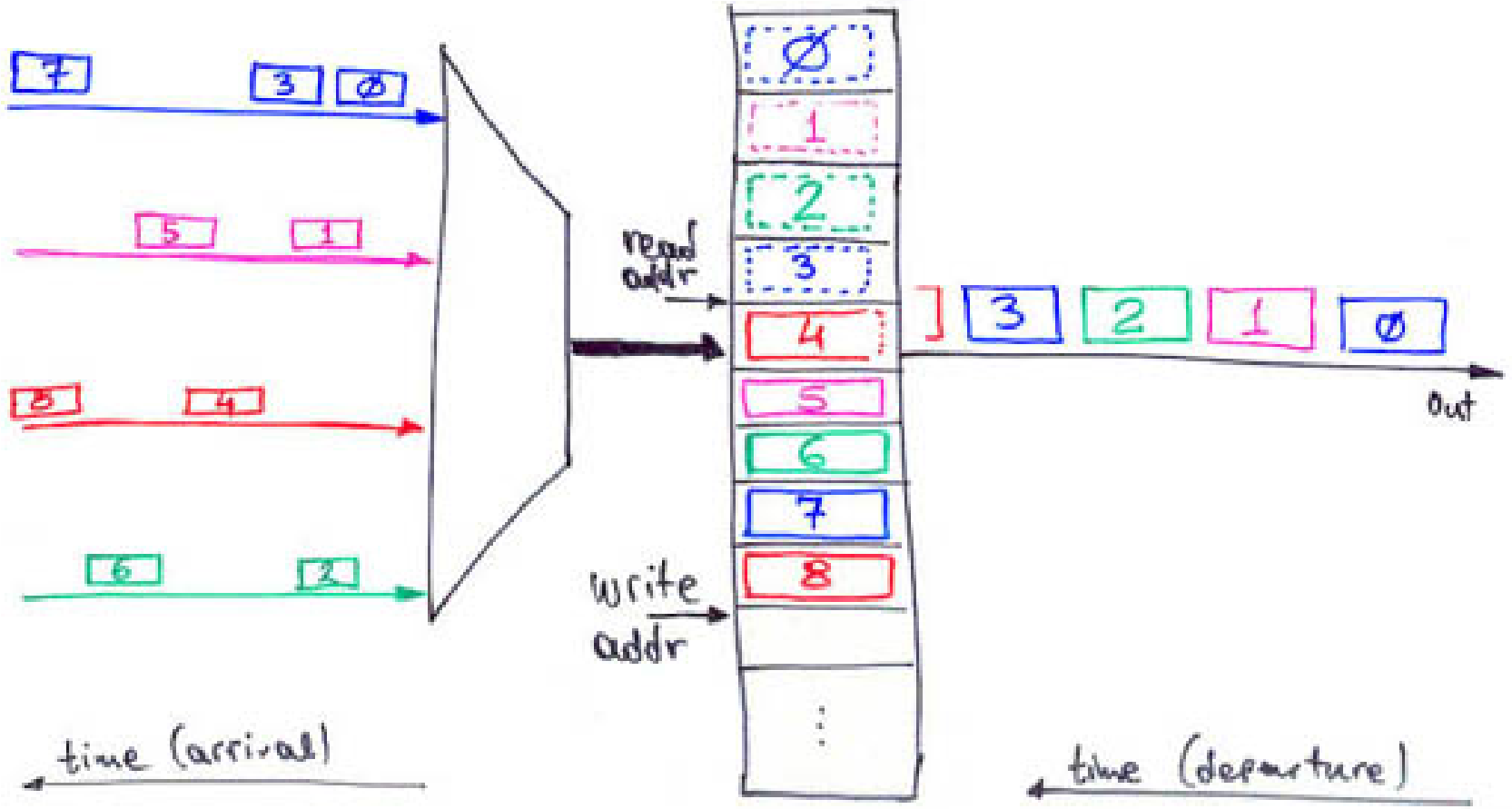


Note: can build the general case out of this:



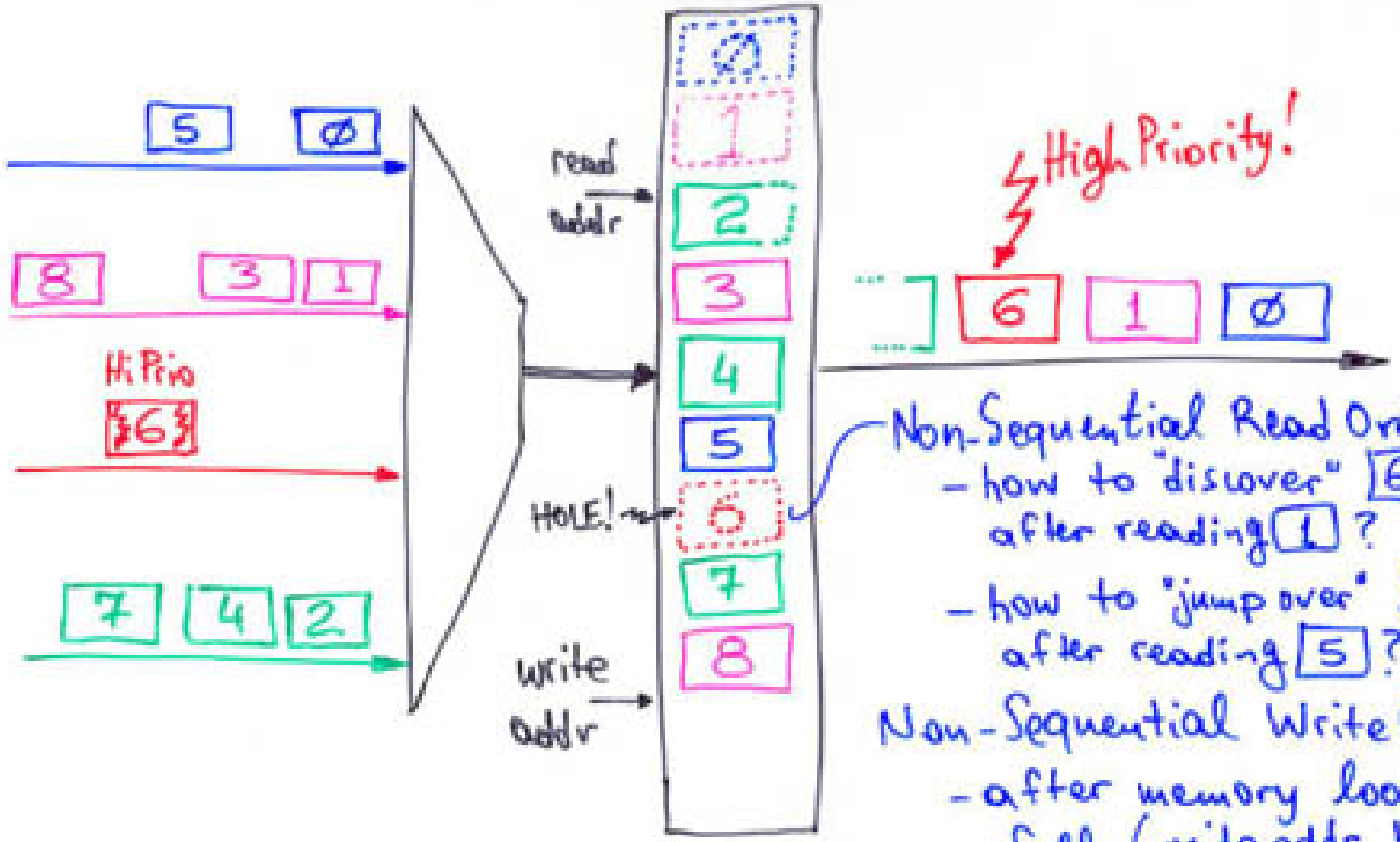
Simple Case:

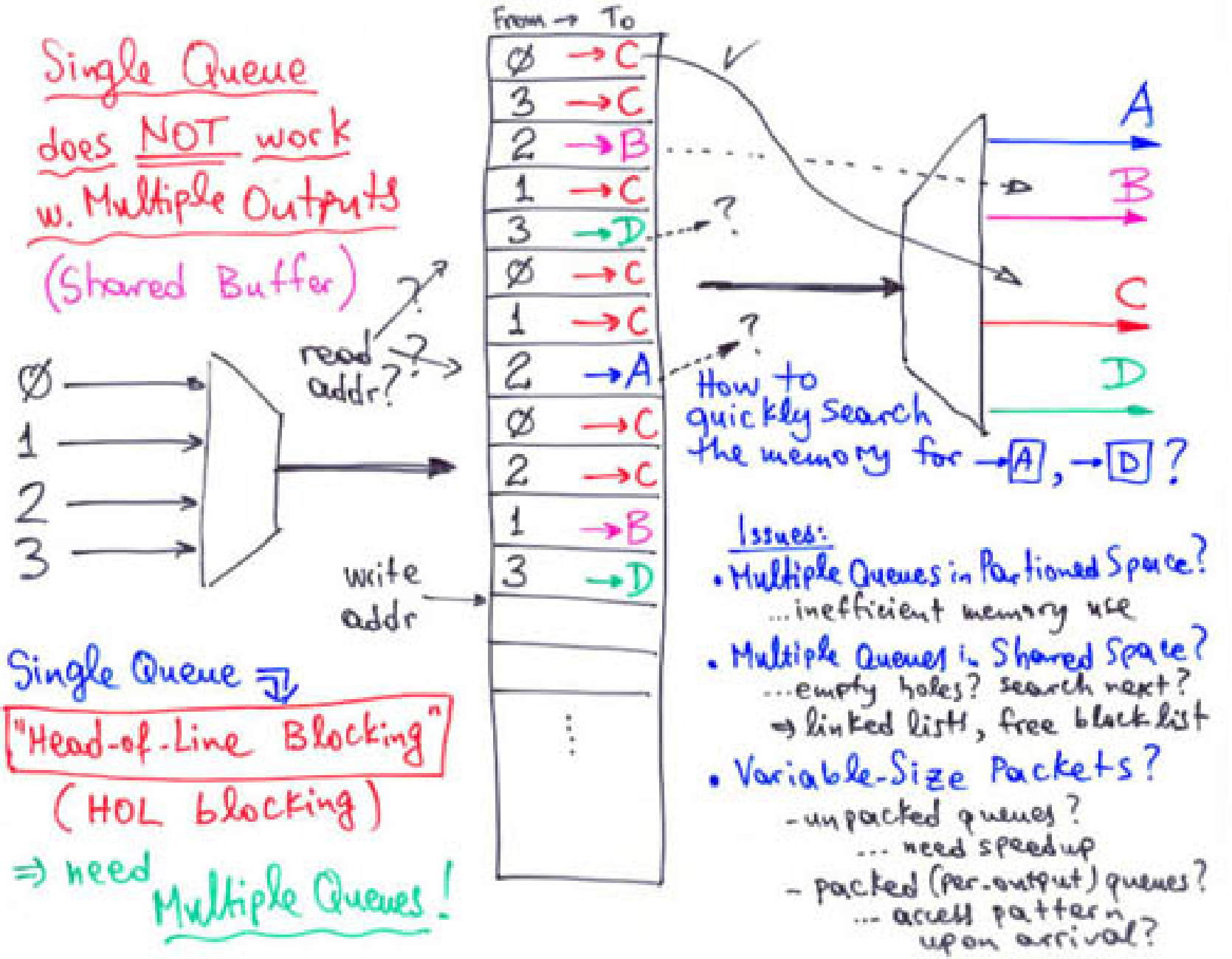
- Single Output (output queueing)
- Single Queue
- Strict First-In First-Out (FIFO) service discipline



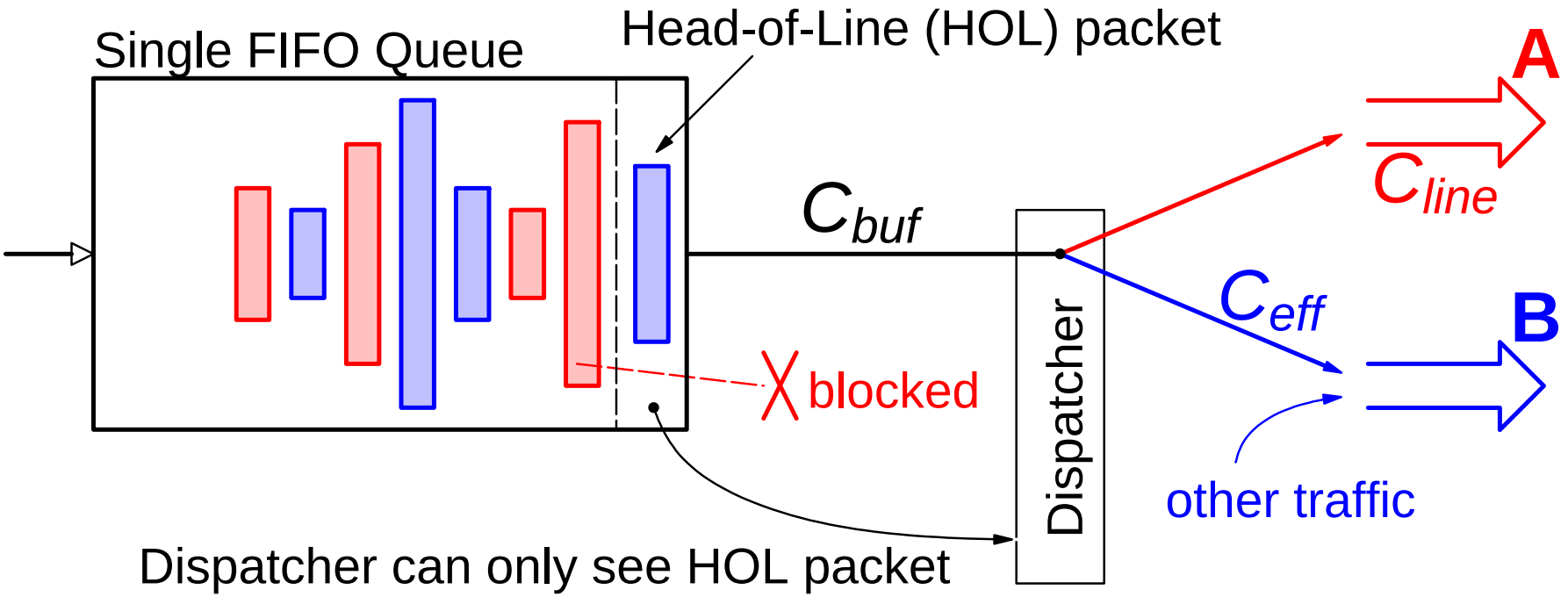
Single Queue does NOT work w. Multiple Priorities

Different QoS classes ⇒ non-FIFO service order




Single Queue
does NOT work
w. Multiple Outputs
(Shared Buffer)
From → To
Ø →C
3 →C
2 →B
1 →C
3 →D
Ø →C
1 →C
2 →A
Ø →C
2 →C
1 →B
3 →D
read addr?
write addr
How to quickly search the memory for →A, →D ?
A
B
C
D
Issues:
• Multiple Queues in Partioned Space?
...inefficient memory use
• Multiple Queues in Shared Space?
...empty holes? search next?
⇒ linked list, free blacklist
• Variable-Size Packets?
- unpacked queues?
... need speedup
- packed (per-output) queues?
... access pattern upon arrival?
Single Queue ⇒
"Head-of-Line Blocking"
(HOL blocking)
⇒ need Multiple Queues!

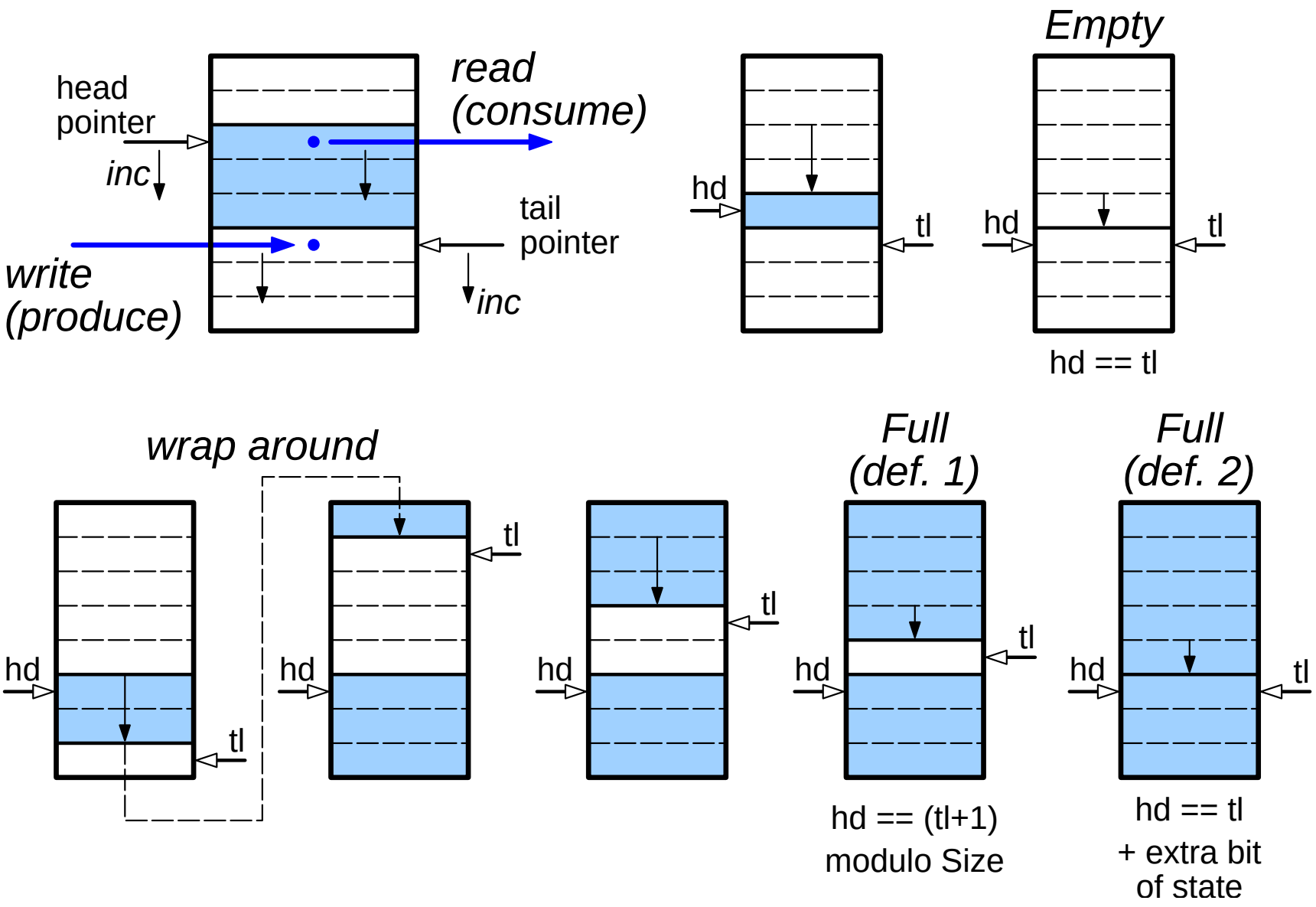
Shared Queue ⇒ Head-of-Line (HOL) Blocking



- Queue shared among multiple destinations/classes (A and B)
- $C_{line} < C_{buf}$ or $C_{eff} < C_{buf}$ due to other traffic

⇒ HOL packet can block packets behind it to other dest./classes

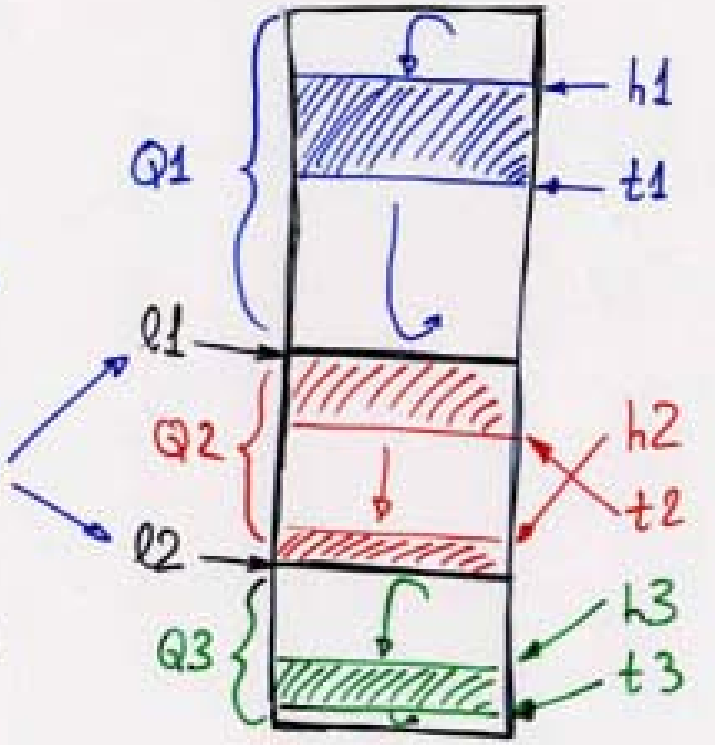
Reminder: Circular Array Implementation of FIFO Queue



B.2 Multiple FIFO Queues with Statically Partitioned Space

for each of them:

- One RAM for all queues;
  Two counters/pointers per queue.
- Limits between queue areas (partitions)
  can be "hardwired", or "configurable"
  off-line (when queues are empty);
  limit registers needed in the latter case
- Under utilization of memory space
  (one queue may overflow, while space
  exists on others)
- Variable-size packets OK.



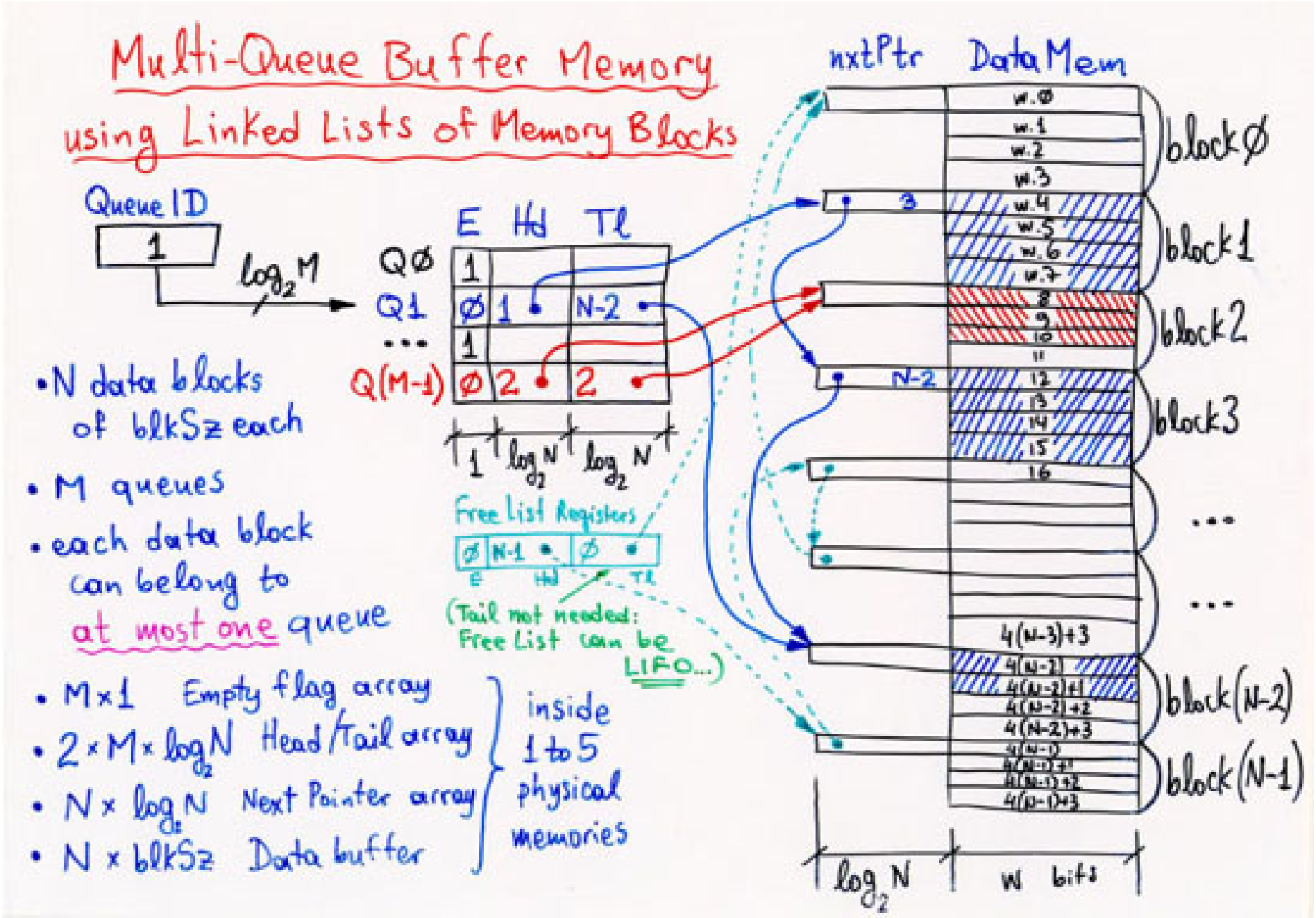
534
12.6

Multiple Queues Sharing a Buffer Memory Space

- Often need many queues where most have low occupancy (or empty) but few have high occupancy
- Departures create holes $\Rightarrow$ empty space is fragmented
  $\Rightarrow$ neighboring packets physically non-contiguous
  $\Rightarrow$ linked-list data structure

  Reference: Y. Tamir, G. Frazier: "High Performance Multi-Queue Buffers for VLSI Communication Switches", ISCA 1988
- "*malloc( )*" works on fixed-size blocks
  - block size is a tradeoff between fragmentation cost (for very large blocks) and memory address rate (for very small blocks): see exercise set 4
  - … unless multiple packet fragments in a same block – ex. 6.1

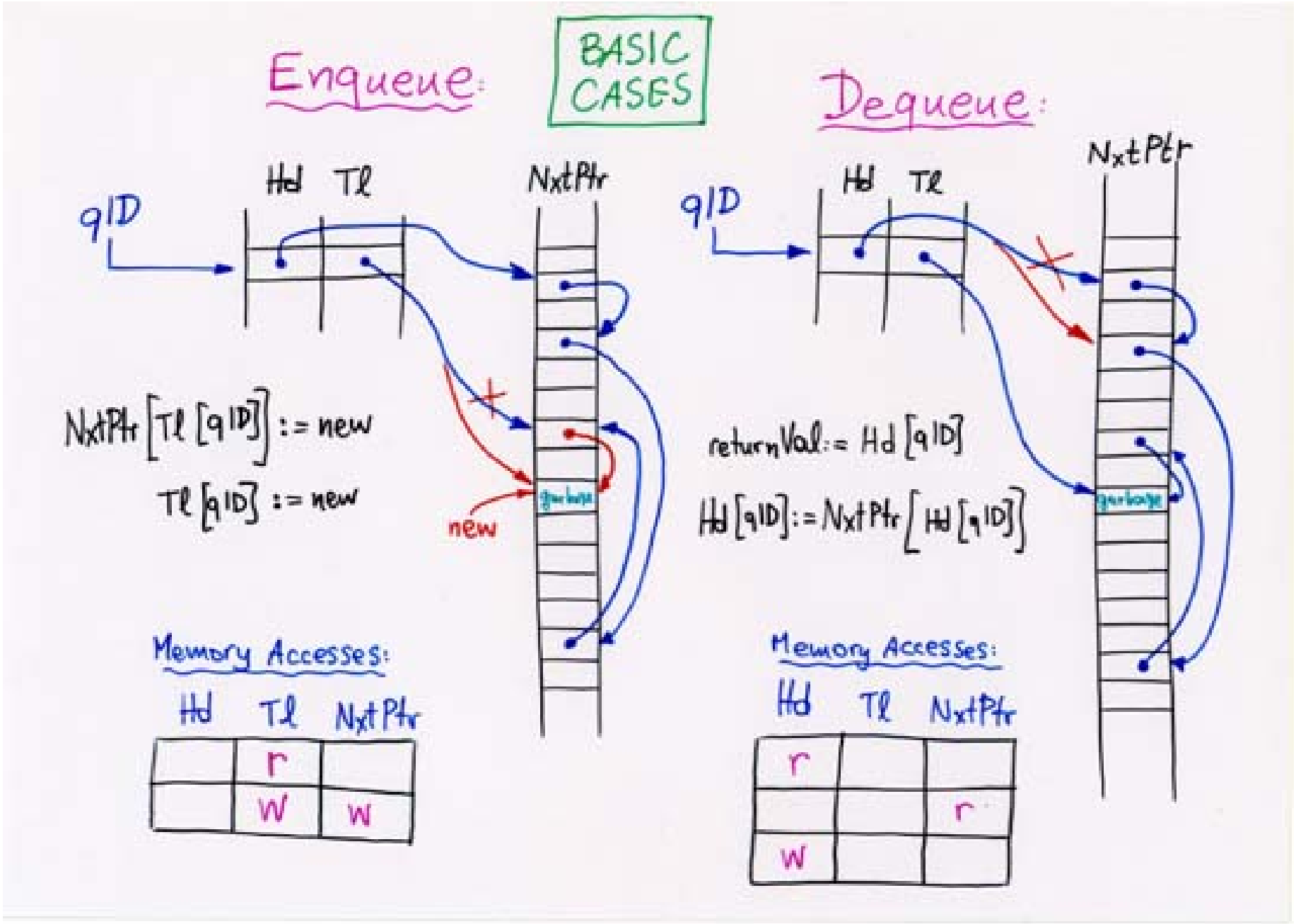
Multi-Queue Buffer Memory using Linked Lists of Memory Blocks

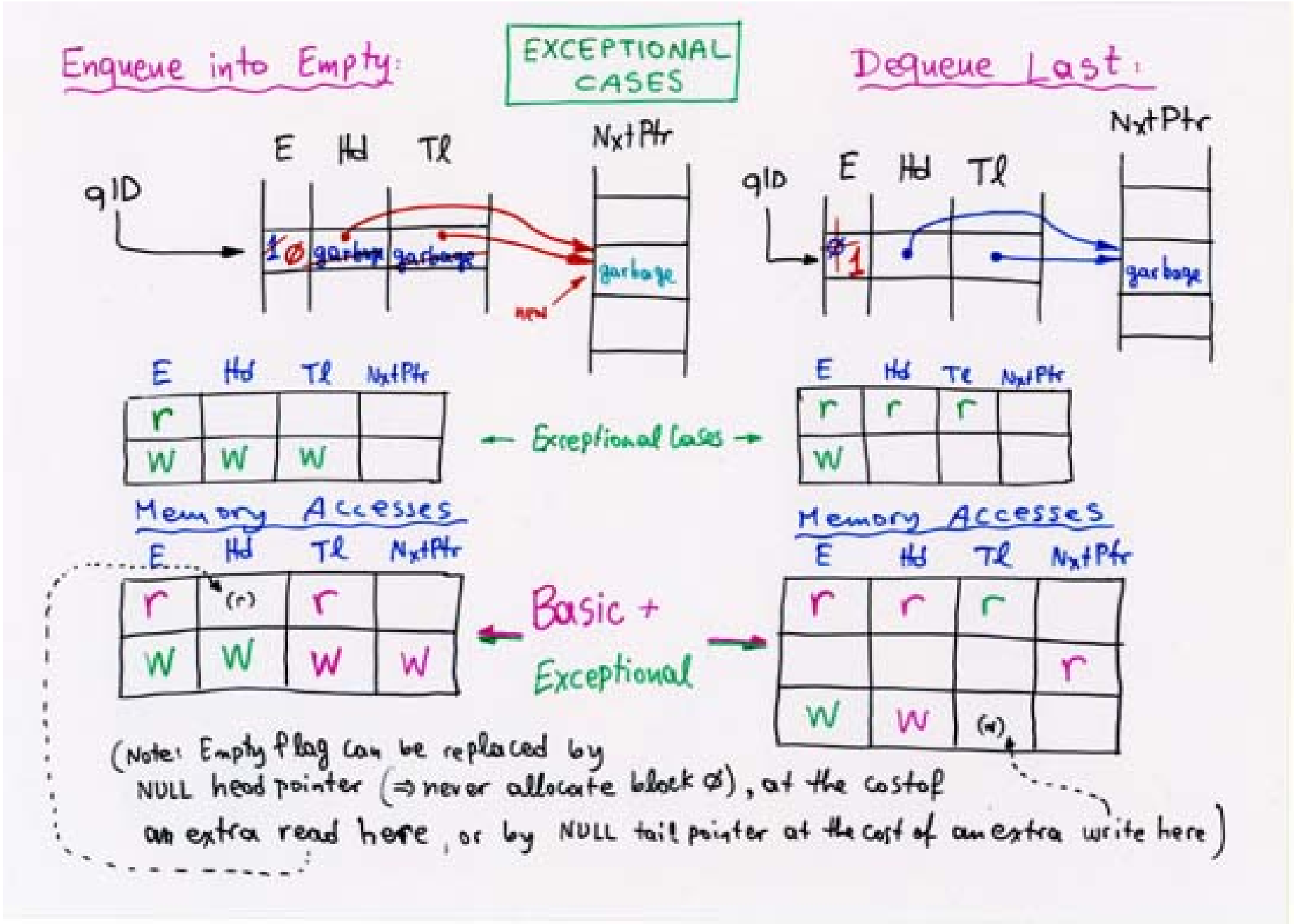


- N data blocks of blkSz each
- M queues
- each data block can belong to at most one queue

- M x 1 Empty flag array
- 2 x M x log2 N Head/Tail array
- N x log2 N Next Pointer array
- N x blkSz Data buffer

} inside 1 to 5 physical memories


Enqueue:
BASIC CASES
Dequeue:
Hd
Tl
NxtPtr
qID
NxtPtr[Tl[qID]] := new
Tl[qID] := new
new
garbage
returnVal := Hd[qID]
Hd[qID] := NxtPtr[Hd[qID]]
Memory Accesses:
Hd Tl NxtPtr
r
w
w
Memory Accesses:
Hd Tl NxtPtr
r
r
w


Enqueue into Empty:
EXCEPTIONAL CASES
Dequeue Last:
qID
E
Hd
Tl
NxtPtr
garbage
new
Exceptional cases
Memory Accesses
Basic +
Exceptional
(Note: Empty flag can be replaced by NULL head pointer (⇒ never allocate block ∅), at the cost of an extra read here, or by NULL tail pointer at the cost of an extra write here)

Cost-Performance Tradeoffs:

Assumptions:

- "off-chip" means enq/deq controller FSM & memories are located on separate chips
- "on-chip" means all together
- 1 memacc./cycle/port
- off-chip dependent accesses (read, then use data as next address) cost one extra cycle of latency between them

E	Hd	Tl	NxtPtr	Enqueue Latency (clock cycles)	Enqueue Thruput ($\frac{op's}{cycle}$)	Dequeue Latency (clock cycles)	Dequeue Thruput ($\frac{op's}{cycle}$)
in a single, 1-port off-chip memory				4 or 5	$\frac{1}{4 \text{ or } 5}$	5	$\frac{1}{5 \text{ or } 4}$
1-port on-chip	in a single, 1-port off-chip memory			4	1/4	5	1/4
1-port on-chip	1-port, off-chip $\log_2 N$ wide		1-port off-chip	3*	1/3	5	1/3
1-port on-chip	1-port off-chip	1-port off-chip	1-port off-chip	3*	1/2	5	1/2
1-port on-chip	1-port on-chip	1-port on-chip	1-port off-chip	2*	1/2	3	1/2
2-port on-chip	2-port on-chip	2-port on-chip	1-port off-chip	2*	1	3	1

Note:

* enqueue latency may increase when mixing enqueue and dequeue operations in the same pipeline.

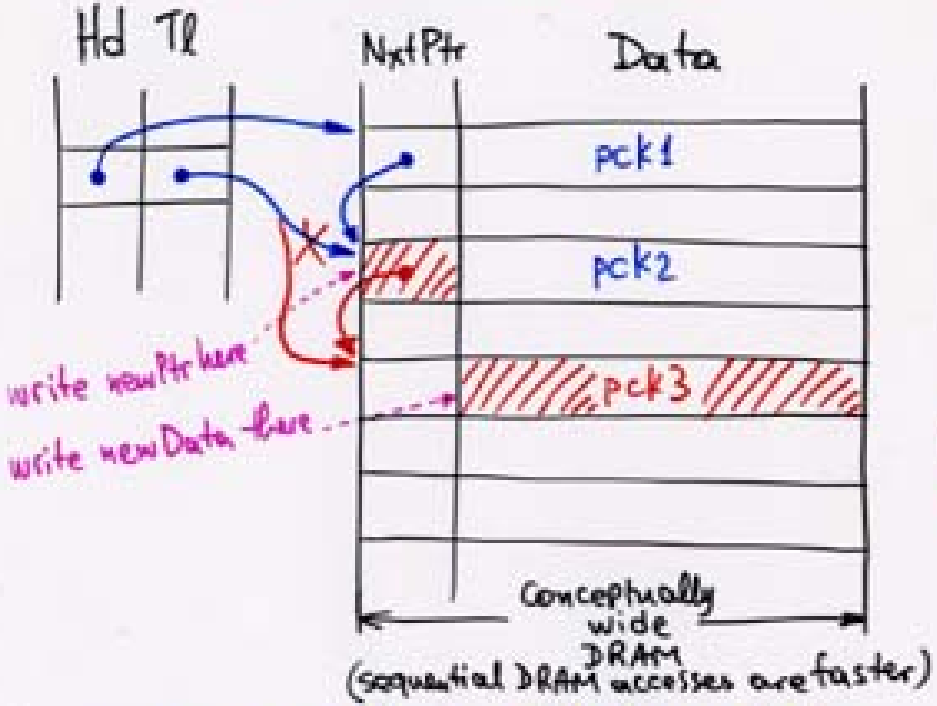
Notes on Enqueue/Dequeue Performance Table

- Table assumes we always need to perform both normal and exceptional accesses – see ex. 5.1
- Table assumes fall-through timing for off-chip SRAM: one-cycle latency per access, plus one extra cycle between dependent off-chip accesses – see ex. 5.2
- For peak throughput: overlap successive operations (latency of individual operations increases)

  For a highly pipelined implementation, refer to: Kornaros, Kozyrakis, Vatsolaki, Katevenis: ``Pipelined Multi-Queue Management in a VLSI ATM Switch Chip with Credit-Based Flow Control", 17th Conf. on Advanced Research in VLSI, 1997

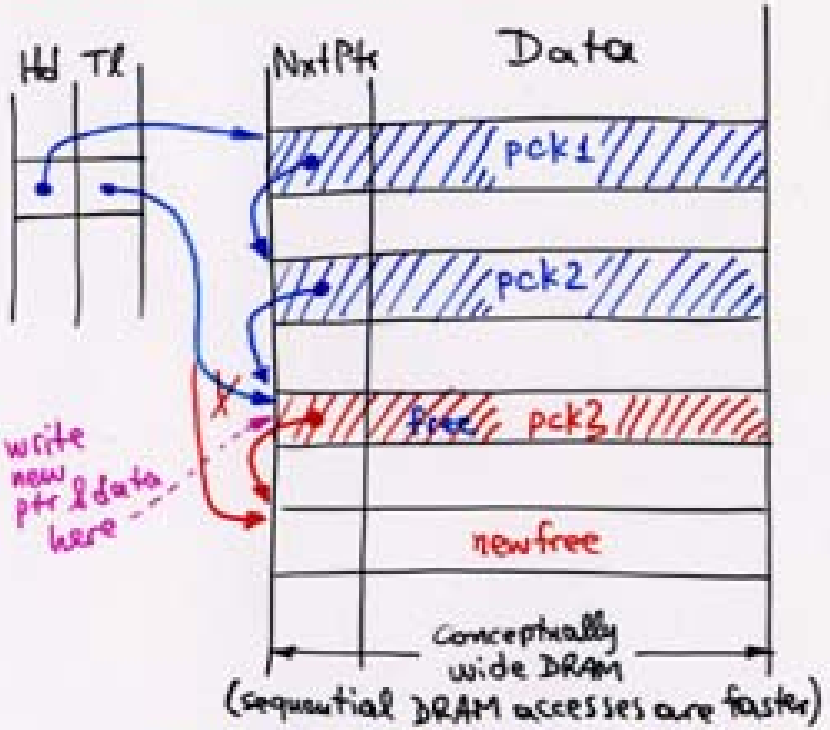
NxtPtr inside Data Buffer Memory:
Optimization for DRAM: Free Block Preallocation

Traditional Enqueue:



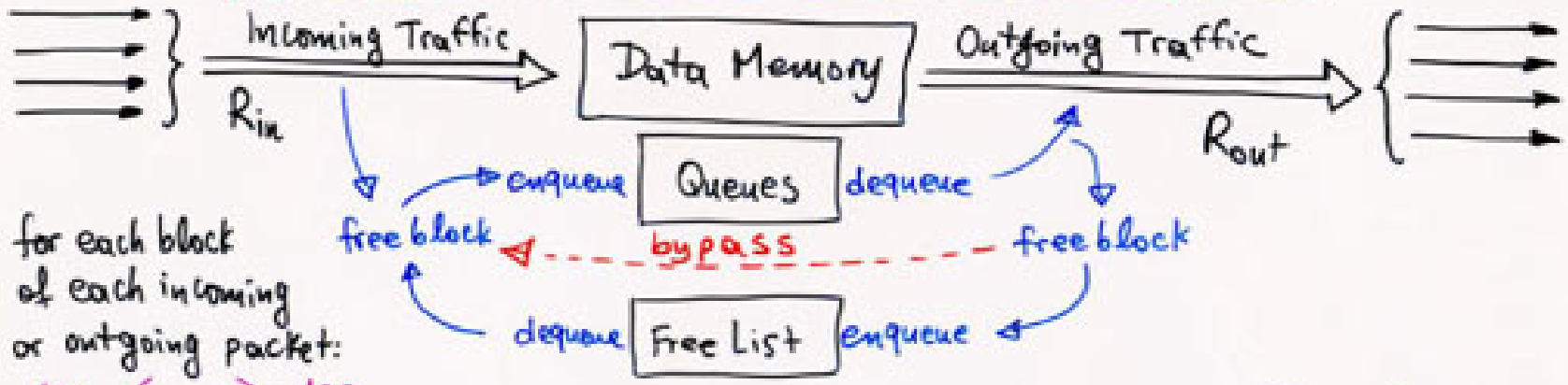
- needs to touch two blocks, at two independent, random addresses

Enqueue with freeBlock preallocation:



- needs only touch one block
- cost: M extra blocks for M queues

Queue Op Rate, Free List Rate, Free List Bypass:



for each block of each incoming or outgoing packet:

$\Rightarrow$ Hd/Tl mem. (Queues) Op Rate $\sim R_{in} + R_{out}$

but NxtPtr mem. Op Rate $\sim 2 \times (R_{in} + R_{out})$

However, FreeList Bypass can reduce this to 1.

[2] P. Andersson, C. Svensson (Lund Univ., Sweden): "A VLSI Architecture for an 80 Gb/s ATM Switch Core", IEEE Innovative Systems in Silicon Conference, Oct. 96.

Assumptions:

- each incoming block/packet is enqueued into a single queue
  (e.g. multicast packets not enqueued into multiple queues)
- for freelist bypass, each outgoing (transmitted) block is also freed
  (e.g. multicast packets depart a single time, simultaneously to all links)

Also consider a Free Block Cache: keep in a small on-chip memory the pointers to the top few blocks of the Free List.

Dropping a Segment (= move to free list)
or Moving a Segment to a Different Queue

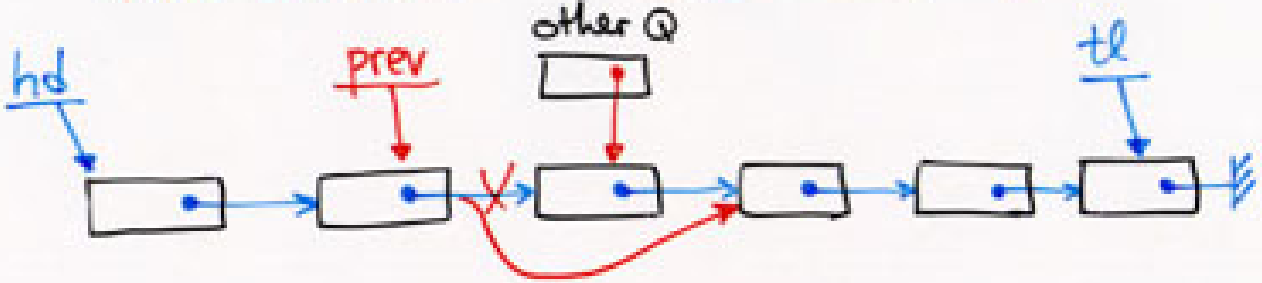
① From the Head of the Source Queue:

(a) dequeue from source queue,

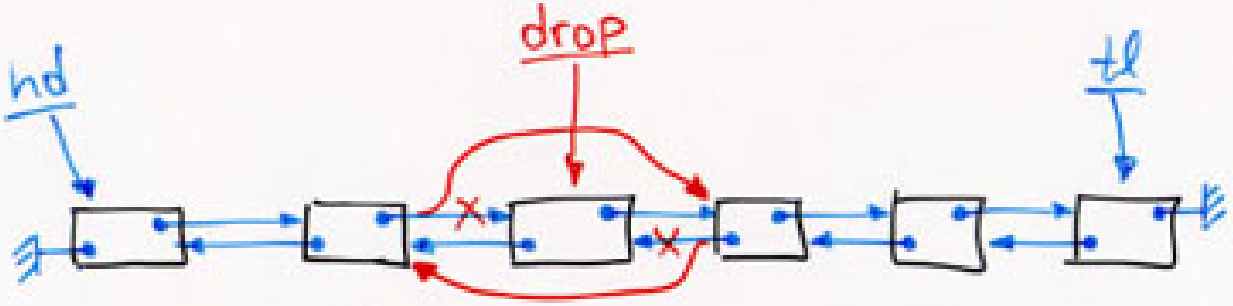
(b) enqueue into destination Q.

② From the "Middle" of the Source Queue

• Need a Pointer to the Previous Segment:



else • Need Doubly Linked List:



Dropping Entire Packet (multi-segment) or moving to other Queue:

(1) In O(pckSize) Time:

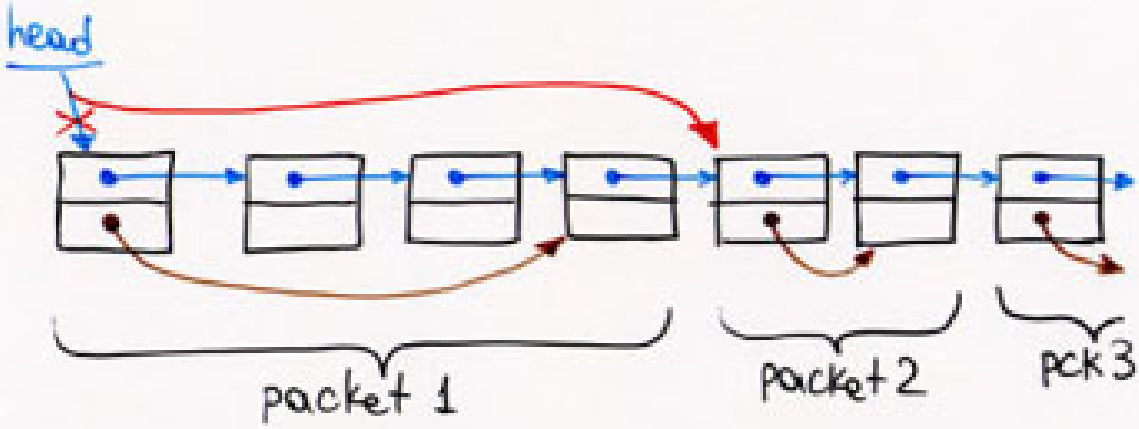
- Drop or move one segment at a time ...

⇒ takes time proportional to the size of the packet

Usually, this consumes the same resources (time, memory ports) as if the packet were transmitted to the output port

(2) In O(1) Time:

- Need two next-pointers per segment:



(the utilization of the second set of pointers is only: $\left(\frac{1}{\text{average number of segments per packet}}\right)$

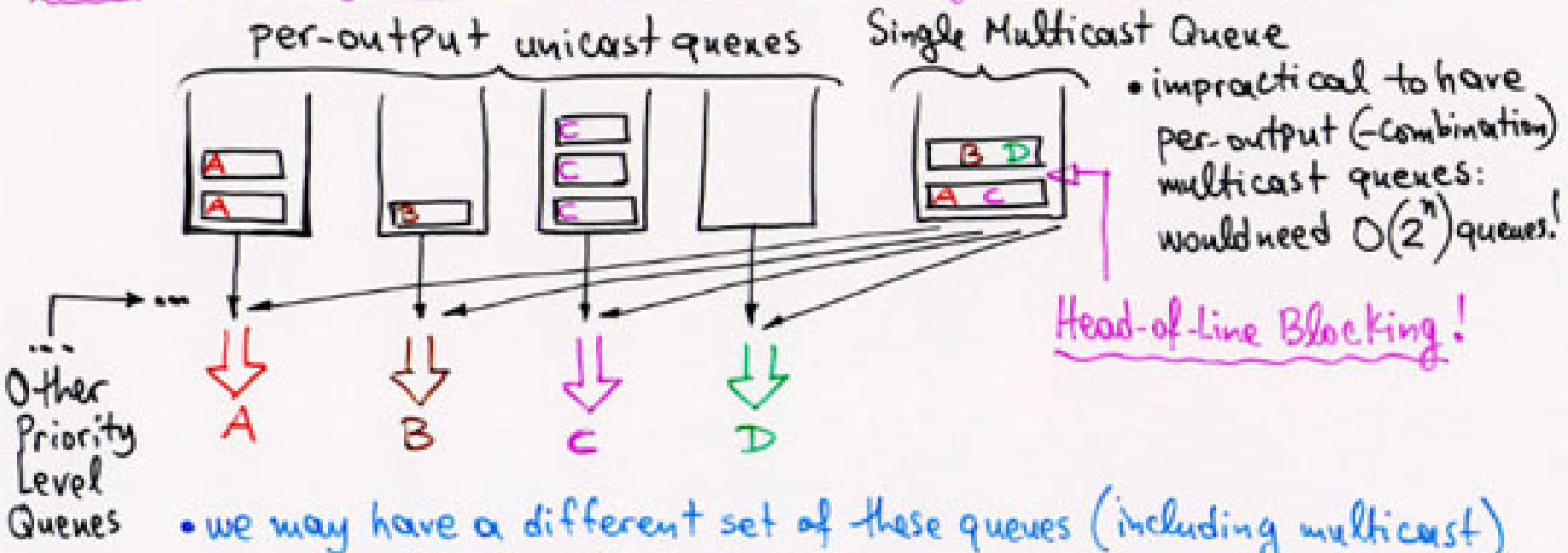
→ alternative: separate "packet descriptor" data structures, like for multicast segment descriptors — see below).

Queueing for Multicast Traffic

- Multicast traffic is expected to become very important in the future –but so has it been for many years in the past…
- Supporting multicast traffic usually increases complexity and cost
- Queueing for Multicast Traffic:
  - Each segment (block) allowed in only one queue $\Rightarrow$ HOL blocking
  - Each segment allowed in multiple queues $\Rightarrow$ need many nxtPtr's
  - Enqueue throughput and nxtPtr space: static vs. dynamic sharing
- References:
  - F. Chiussi, Y. Xia, V. Kumar: "Performance of Shared-Memory Switches under Multicast Bursty Traffic", IEEE Jour. Sel. Areas in Communications (JSAC), vol. 15, no. 3, April 1997, pp. 473-487.
  - D. Stiliadis: "Efficient Multicast Algorithms for High-Speed Routers", Proc. IEEE Workshop on High Performance Switching and Routing (HPSR 2003), Torino, Italy, June 2003, pp. 117-122.

Multicast Traffic: Same or Different Queues with Unicast Traffic?

Case (1): Each segment is only allowed to belong to a single queue:

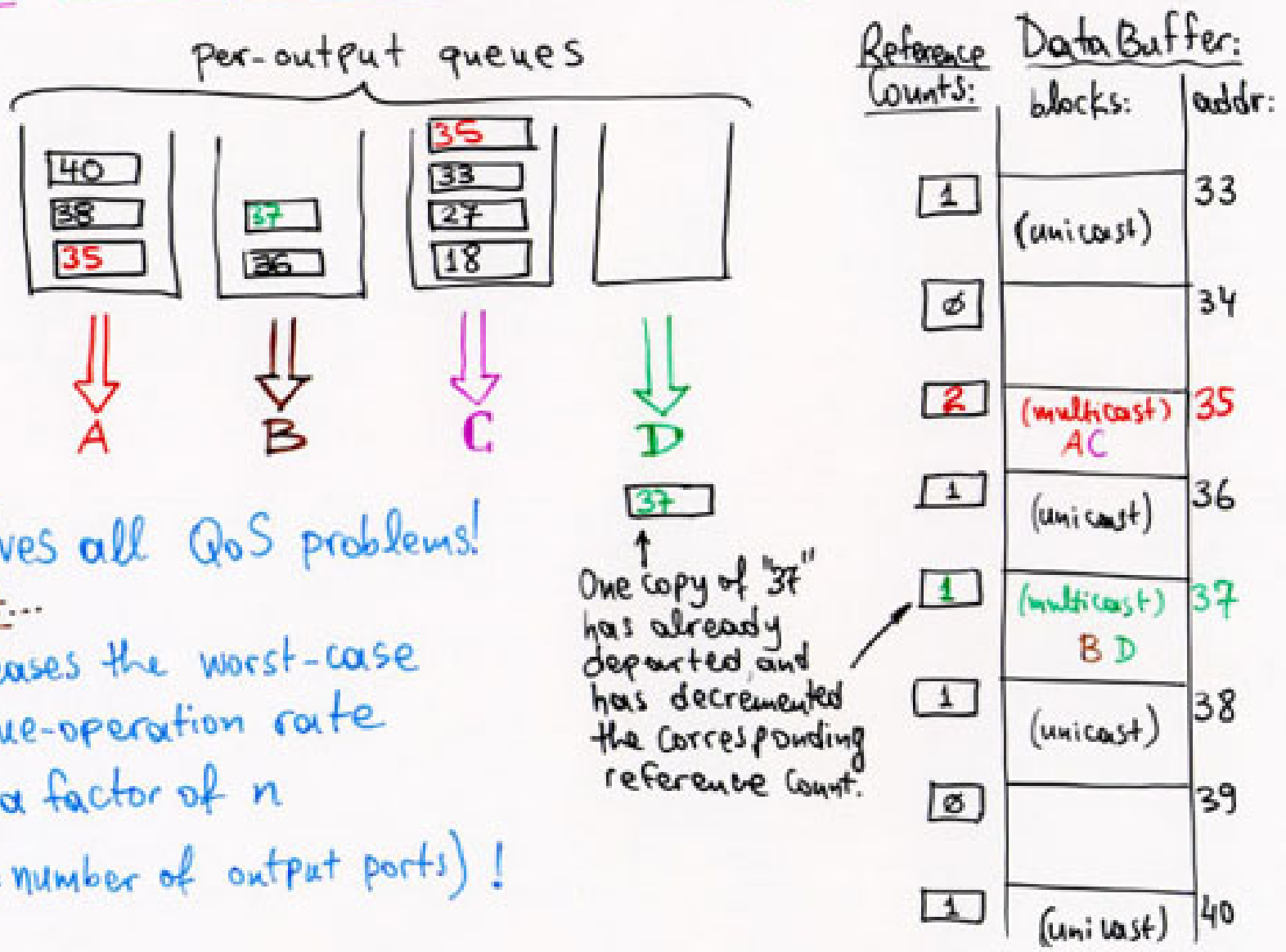


- impractical to have per-output (-combination) multicast queues: would need $O(2^n)$ queues!

Head-of-Line Blocking!

- we may have a different set of these queues (including multicast) per priority level, but it may still happen that traffic destined to outputs A and C currently exists at priority levels higher than "our" cell [A C] while all queues destined to B and D at priority levels above "our" cell [B D] are empty...

Case (2): Each segment is allowed to belong to multiple queues:



- Solves all QoS problems!

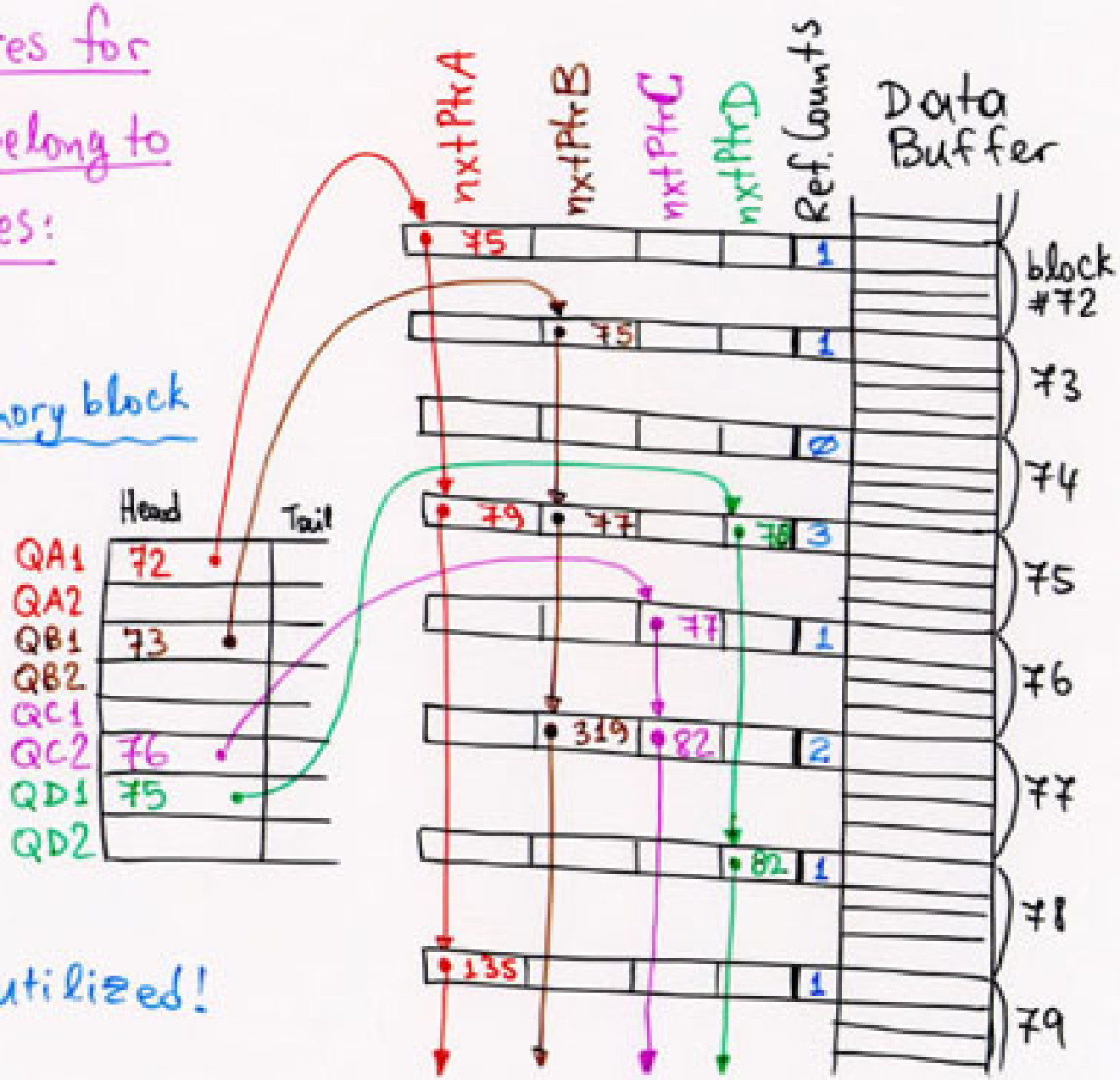
  but...

- Increases the worst-case queue-operation rate by a factor of n (n = number of output ports)!

Data Structures for a Segment to belong to up to N Queues:

Case (1):

N nxtPtr's per memory block



- most segments are unicast

  ⇓

  nxtPtr's are grossly underutilized!

Case (2): Decouple Linked List Nodes from Data Buffer Addresses

- twice the cost per nxtPtr (need a segPtr as well now)

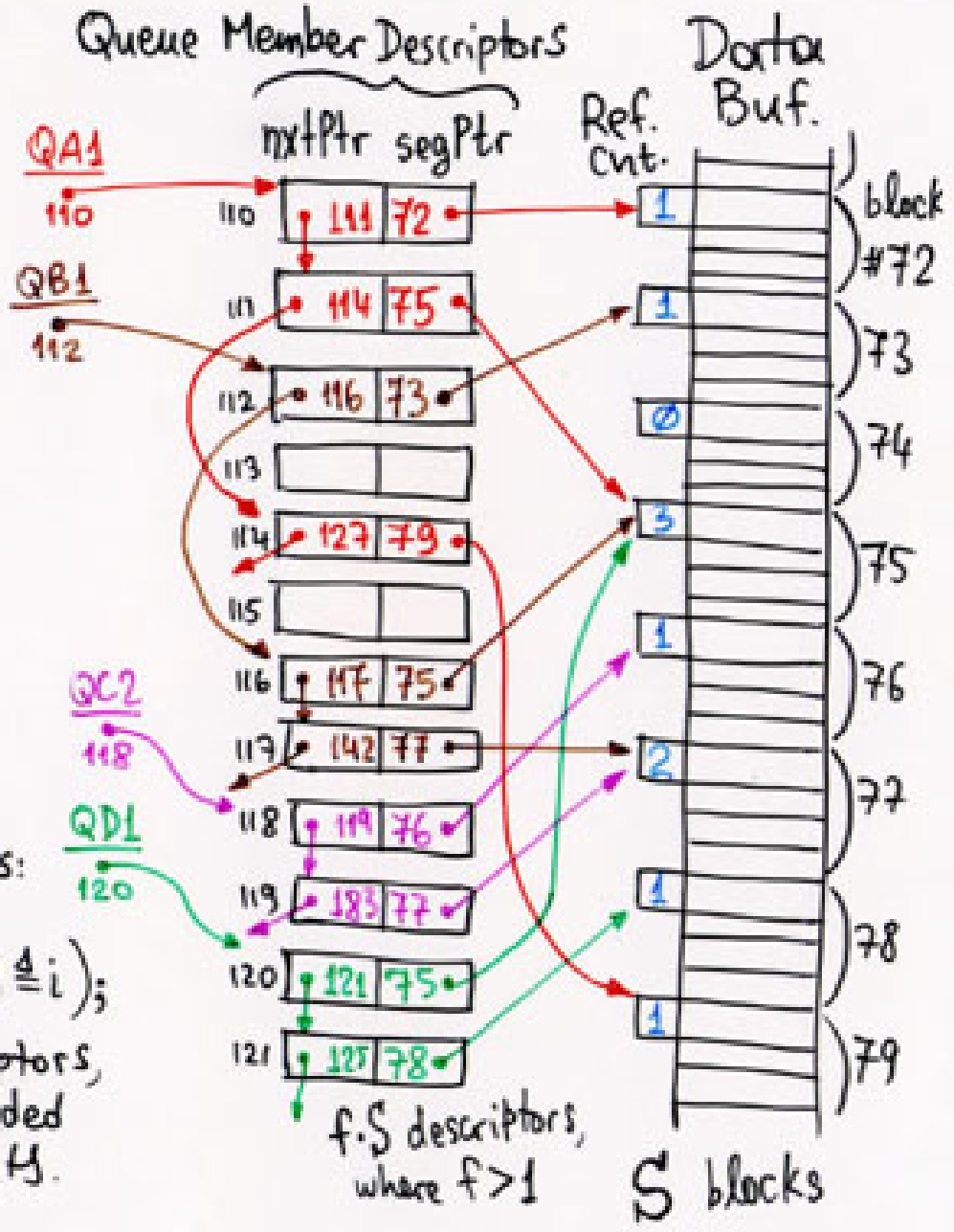
but...

- much fewer than N·S descriptors (based on average ratio of unicast-to-multicast segments, and average fan-out of multicast segments - e.g. f=2)

Optimization:

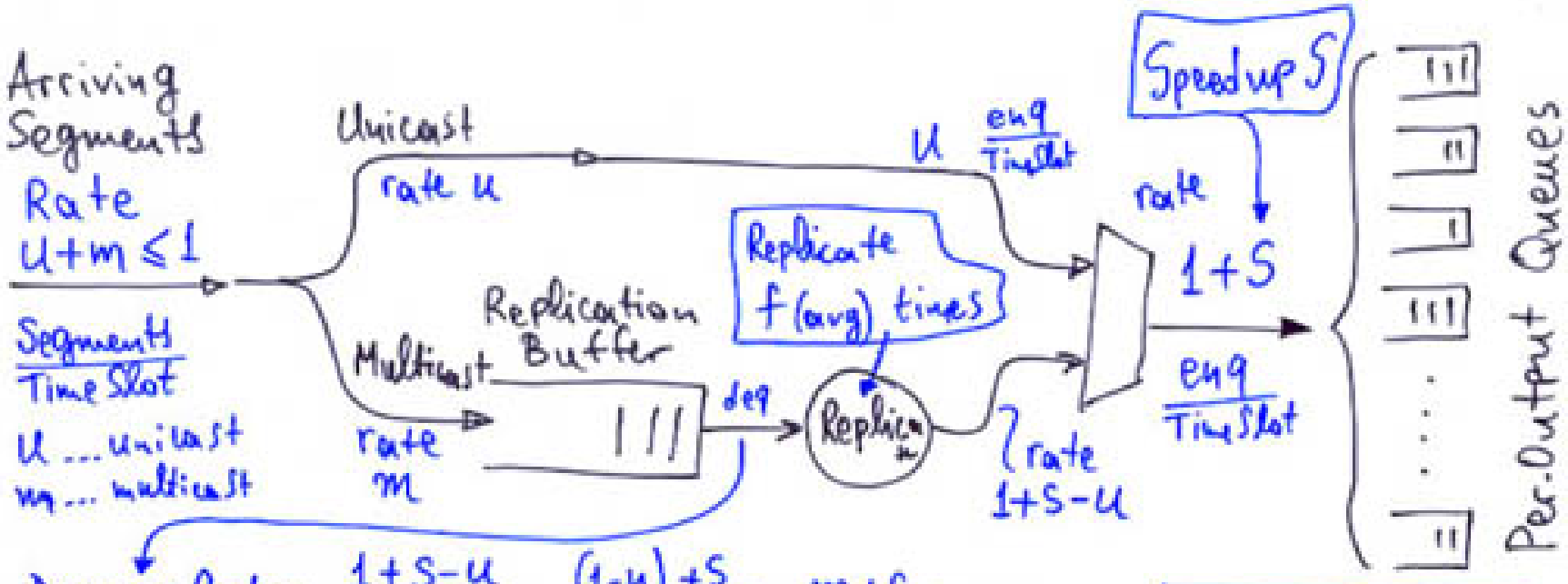
Partition the address space of queue member descriptors into two parts:

- Ø to S-1: unicast-only segments, no segPtr needed (segPtr[i] ≜ i);
- S to fS-1: full queue member descriptors, with nxtPtr and segPtr, intended for use by multicast segments.



Enqueue Operation Rate

for multicast segments onto multiple per-output queues



Dequeue Rate: $\frac{1+S-u}{f} = \frac{(1-u)+S}{f} \geq \frac{m+S}{f} \geq m \Rightarrow \boxed{S \geq m \cdot (f-1)}$

for the replication buffer not to overflow

References:

- F. Chiussi, Y. Xia, V. Kumar : IEEE JSAC, April 1997, pp. 473-487
- D. Stiliadis : HPSR 2003, pp. 117-122