

3.1 The need for multiple queues within a same buffer memory:

Multi-Queue Data Structures

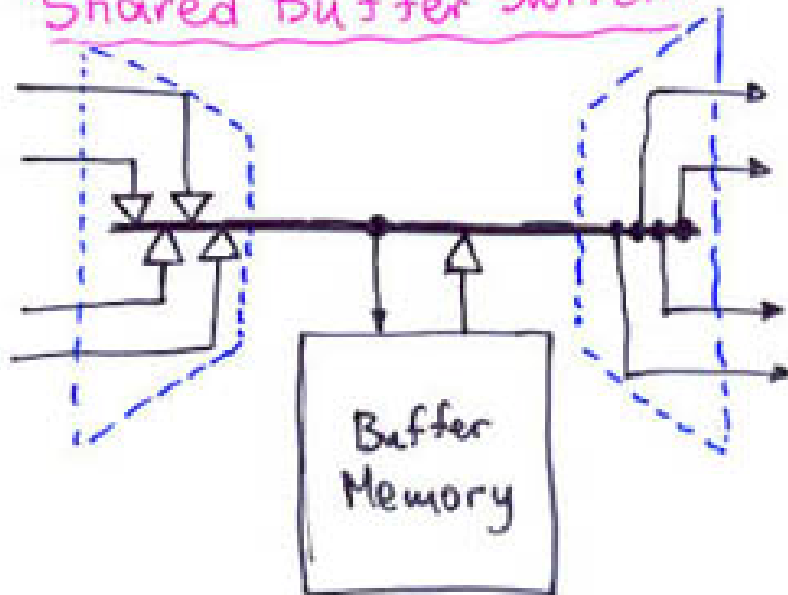
- Buffers memories for time switching: what data structures?
- Single queue feeding multiple destinations/classes
 - ⇒ Head-of-Line (HOL) Blocking ⇒ poor performance
- Multiple Queues in one buffer:
 - Partitioned Space (underutilized) ⇒ circular queues
 - Shared Space (efficient) ⇒ linked-list queues

Time Switching:

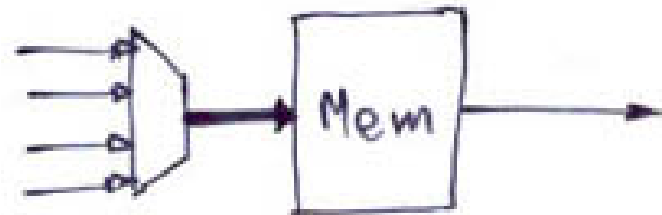


Where in this memory to store packets? (which address?)

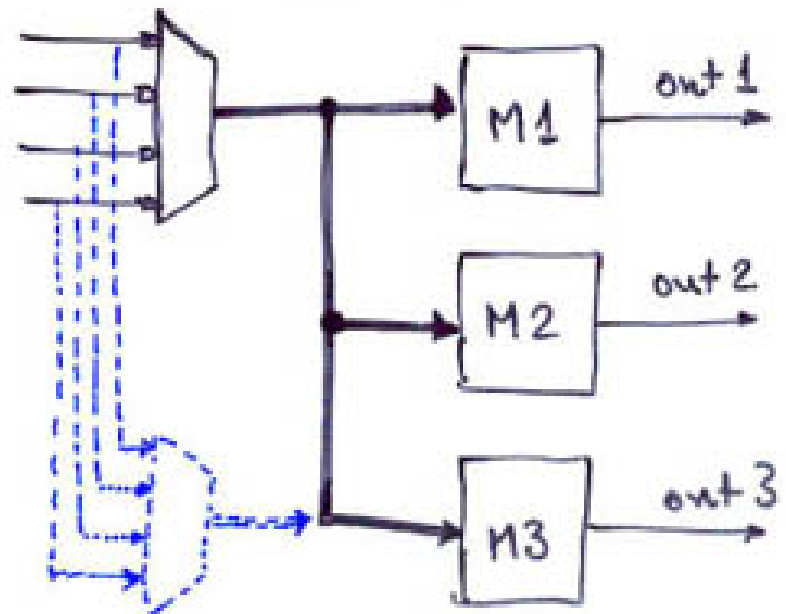
"Shared Buffer Switch":



Simplification: $n \times 1$ Switch "Output Queuing"

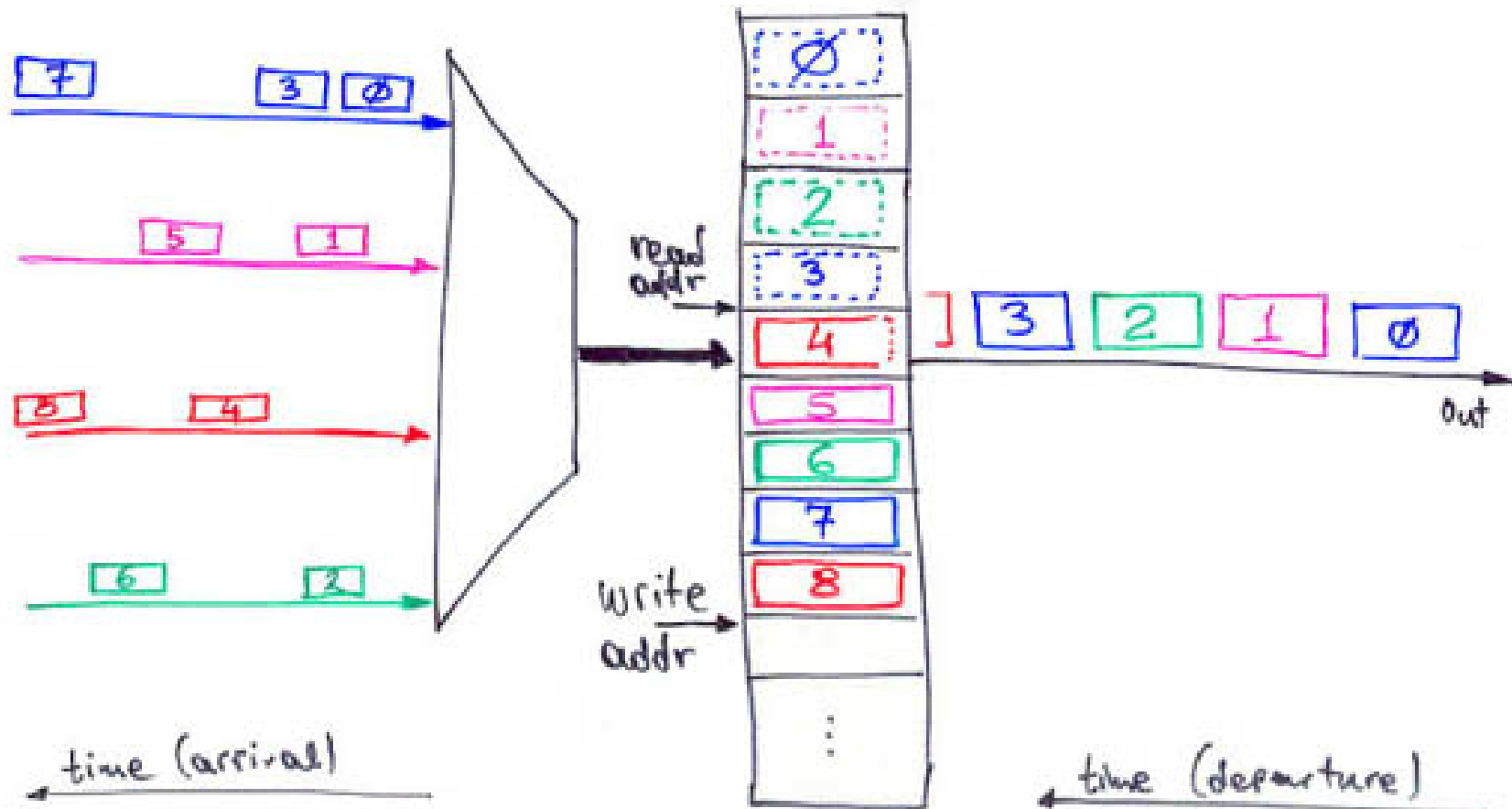


Note: Can build the general case out of this:



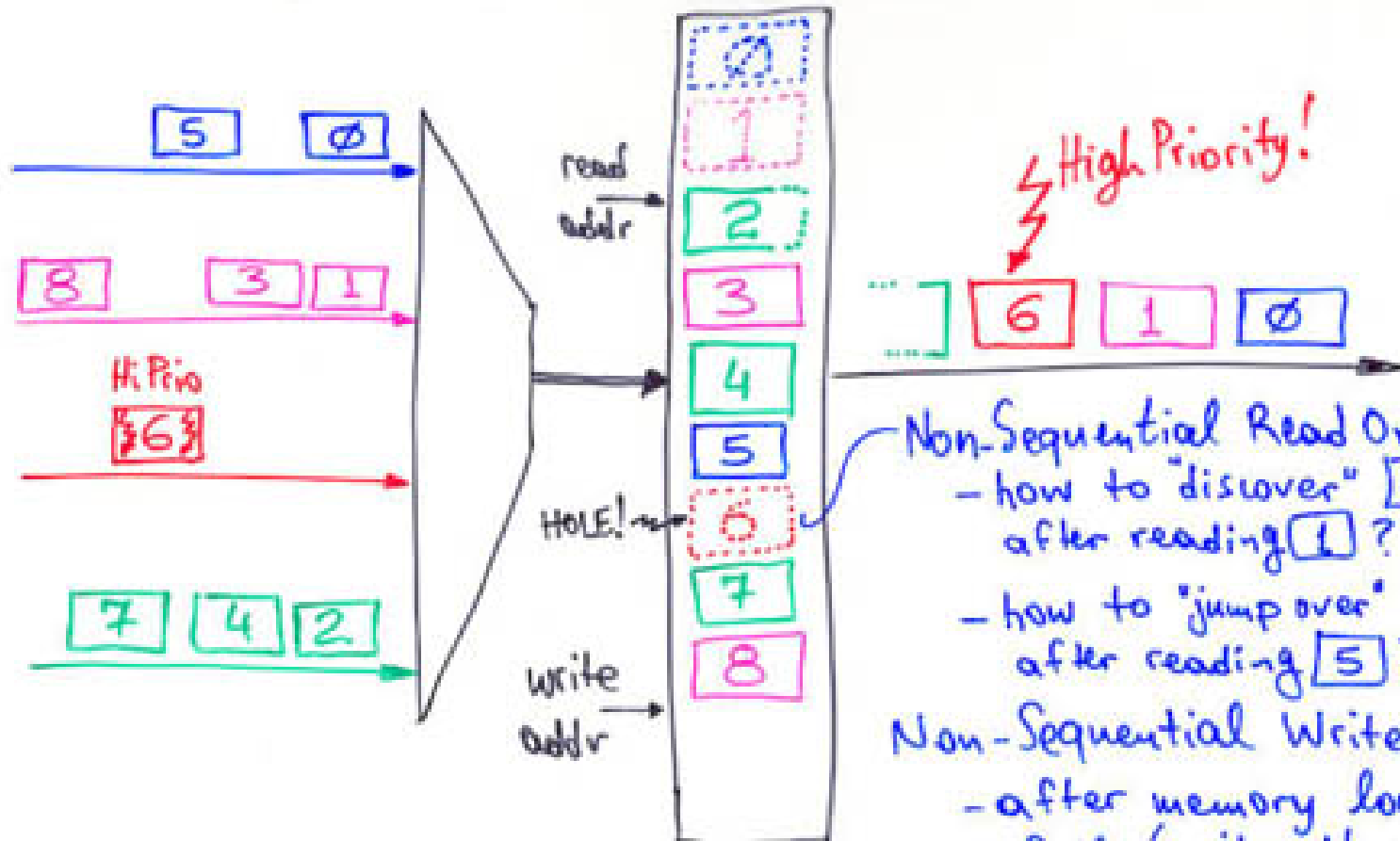
Simple Case:

- 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 \Rightarrow non-FIFO service order



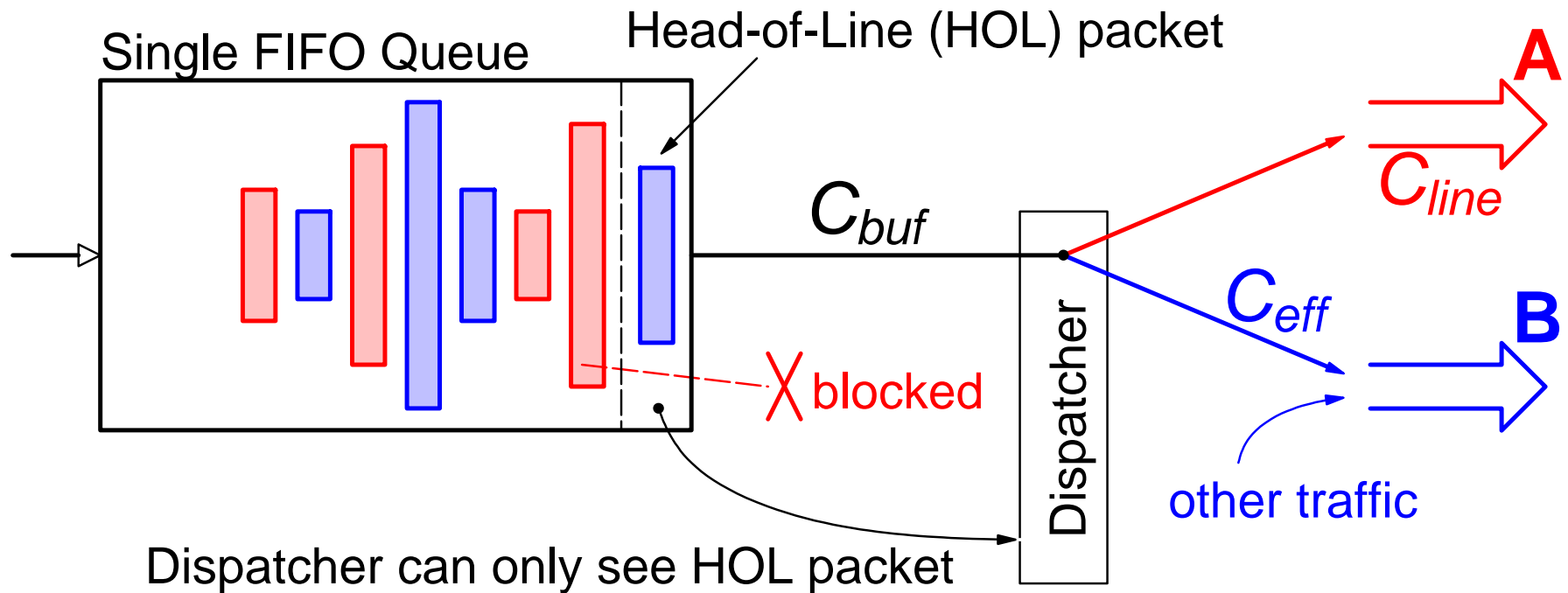
Non-Sequential Read Order:

- how to "discover" 6 after reading 1?
- how to "jump over" 6 after reading 5?

Non-Sequential Write Order:

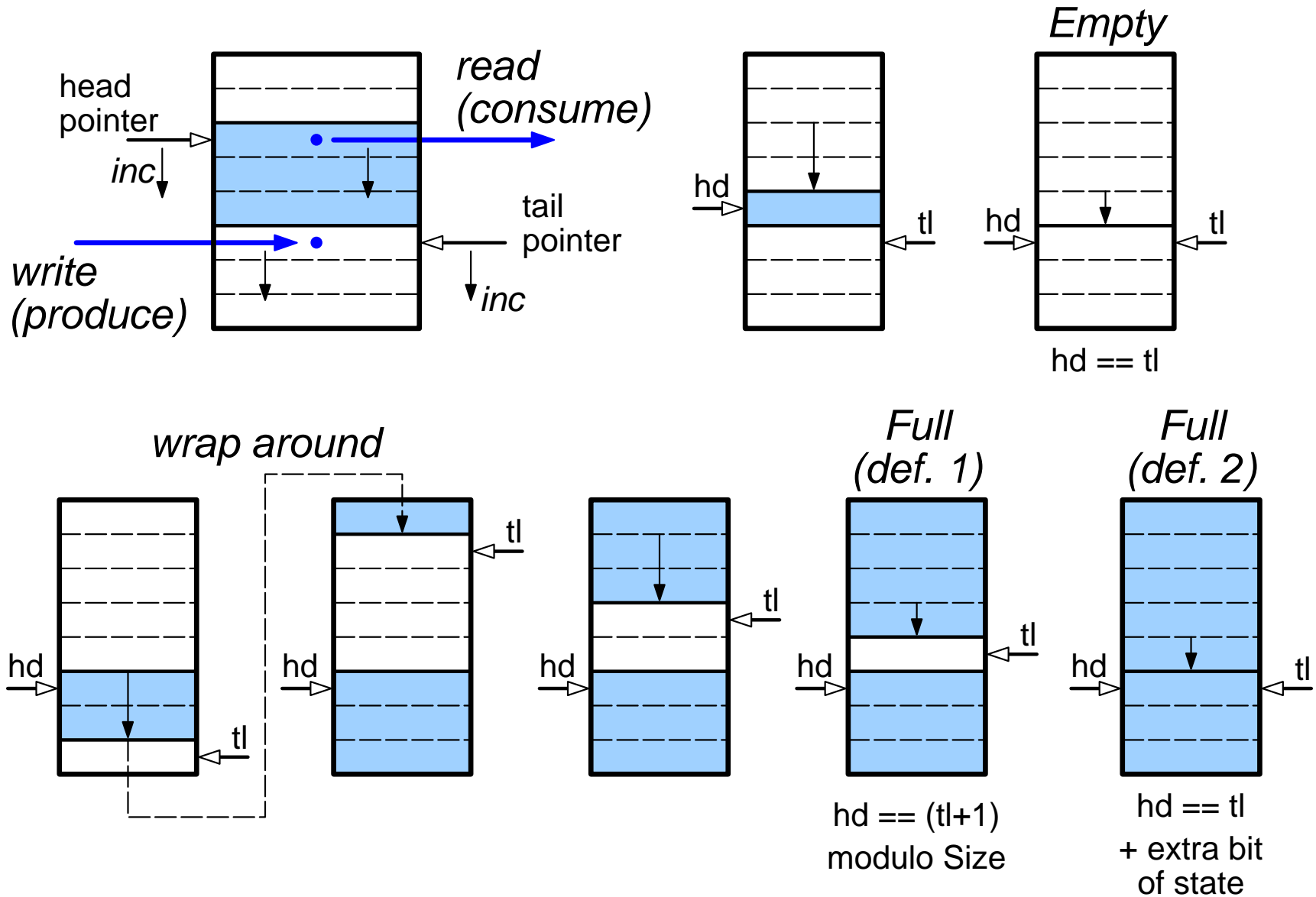
- after memory looks full (write addr. has wrapped around), how to locate the holes and write in them?

Shared Queue \Rightarrow 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
- \Rightarrow HOL packet can block packets behind it to other dest./classes

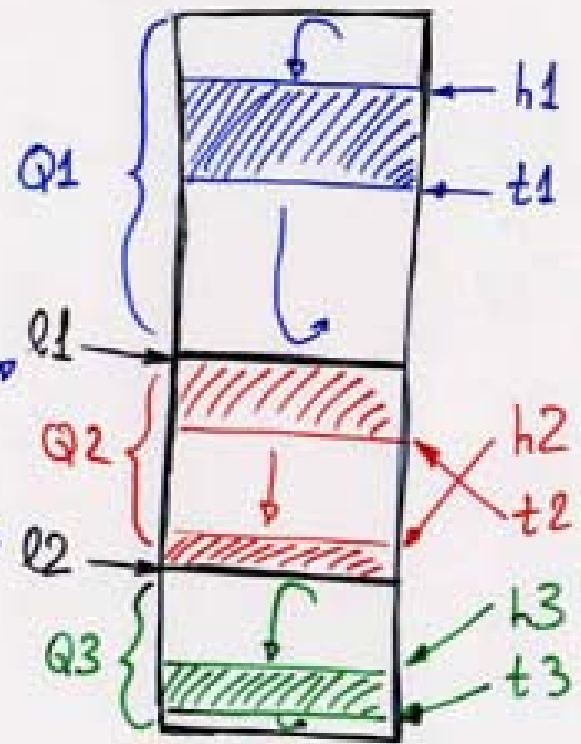
Reminder: Circular Array Implementation of FIFO Queue



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
- Underutilization of memory space (one queue may overflow, while space exists on others)
- Variable-size packets OK.



534
12.6

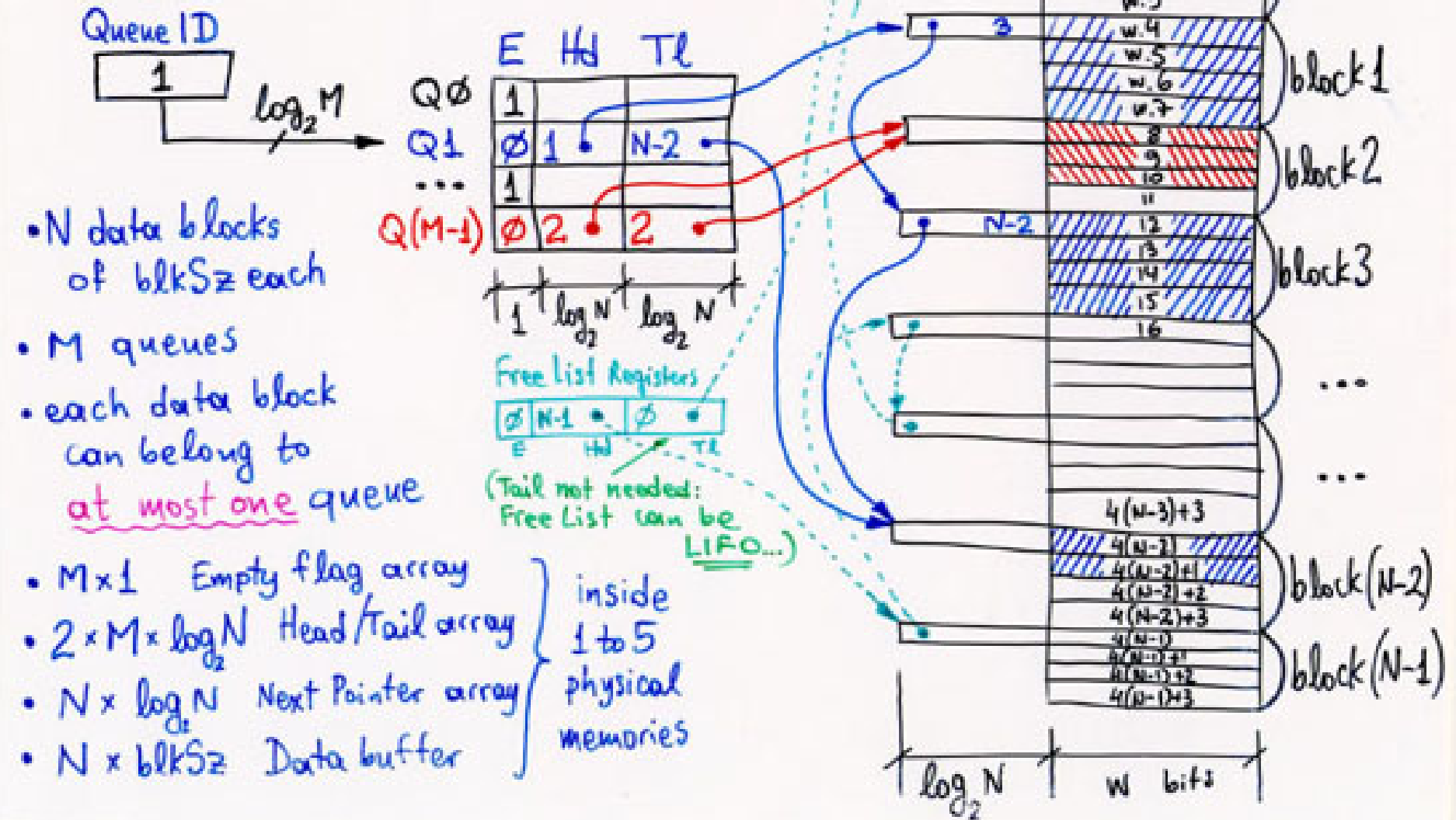
Multiple Queues Sharing a Buffer Memory Space

- 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

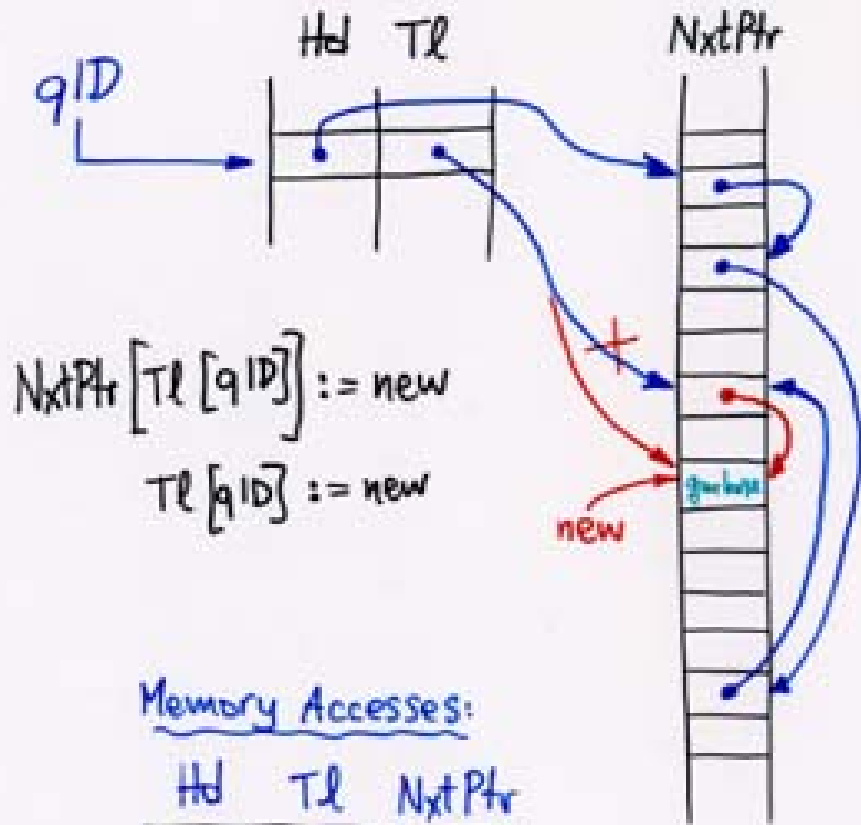


- N data blocks of blkSz each
- M queues
- each data block can belong to at most one queue

- 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:

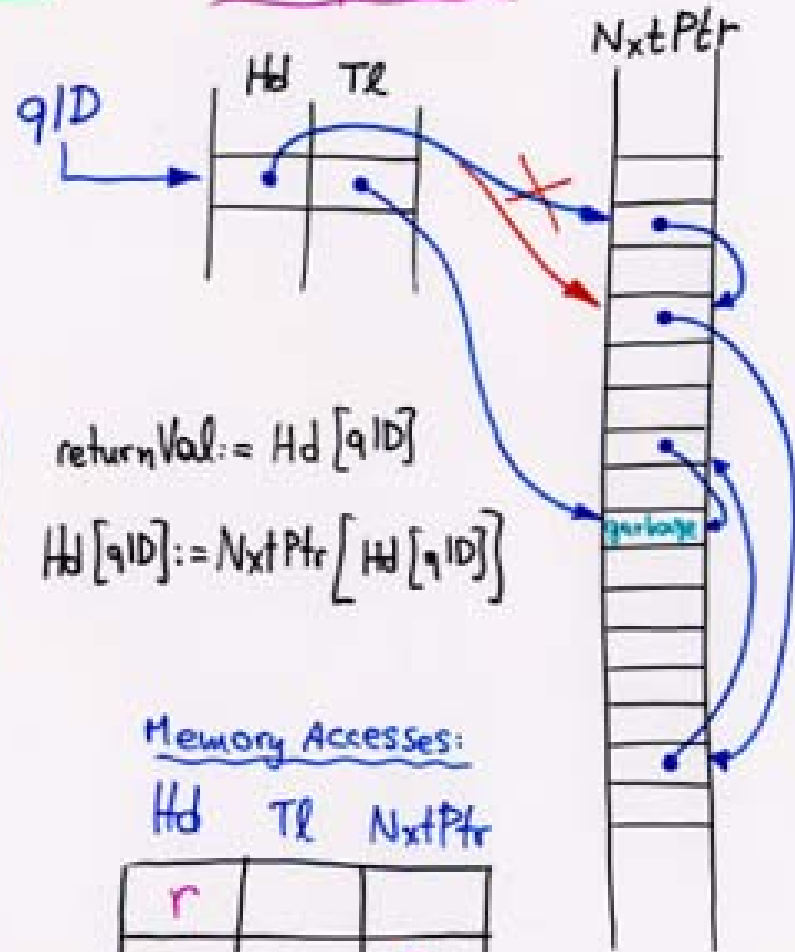
BASIC CASES



Memory Accesses:

Hd	Tr	NextPtr
	r	
	w	w

Dequeue:



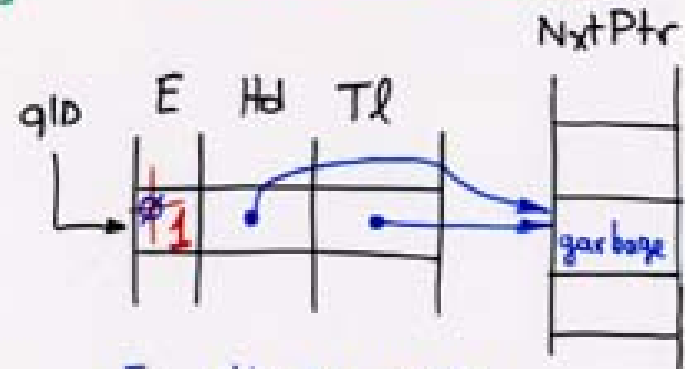
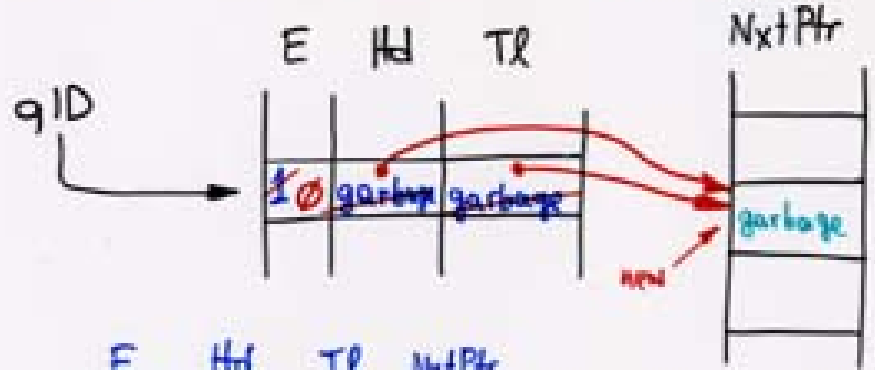
Memory Accesses:

Hd	Tr	NextPtr
r		
		r
w		

Enqueue into Empty:

EXCEPTIONAL CASES

Dequeue Last:



E	Hd	TL	NextPtr
r			
W	W	W	

← Exceptional Cases →

E	Hd	TL	NextPtr
r	r	r	
W			

Memory Accesses

E	Hd	TL	NextPtr
r	(r)	r	
W	W	W	W

← Basic + Exceptional →

Memory Accesses

E	Hd	TL	NextPtr
r	r	r	
W	W	(r)	r

(Note: Empty flag can be replaced by NULL head pointer (⇒ never allocate block 0), 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

Note:

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

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

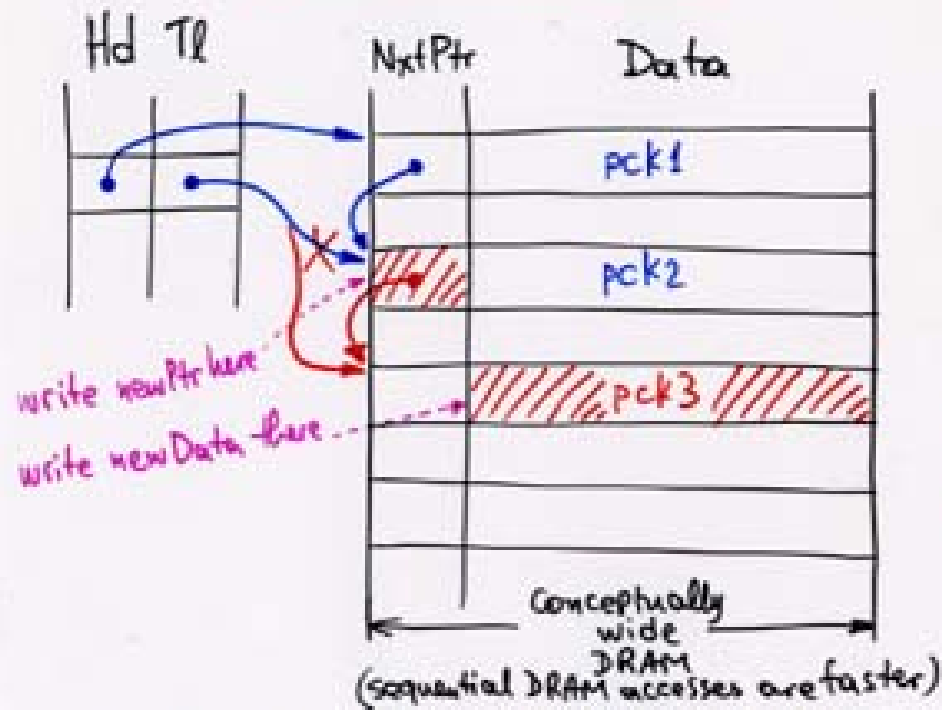
Notes on Enqueue/Dequeue Performance Table

- 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

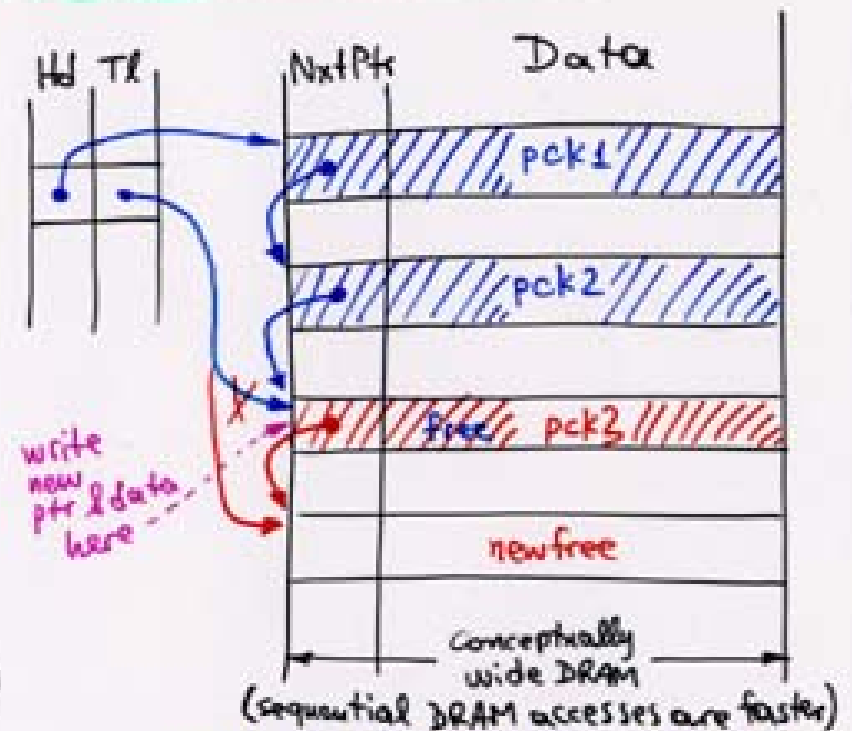
NextPtr inside Data Buffer Memory:
Optimization for DRAM: Free Block Preallocation

Traditional Enqueue:



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

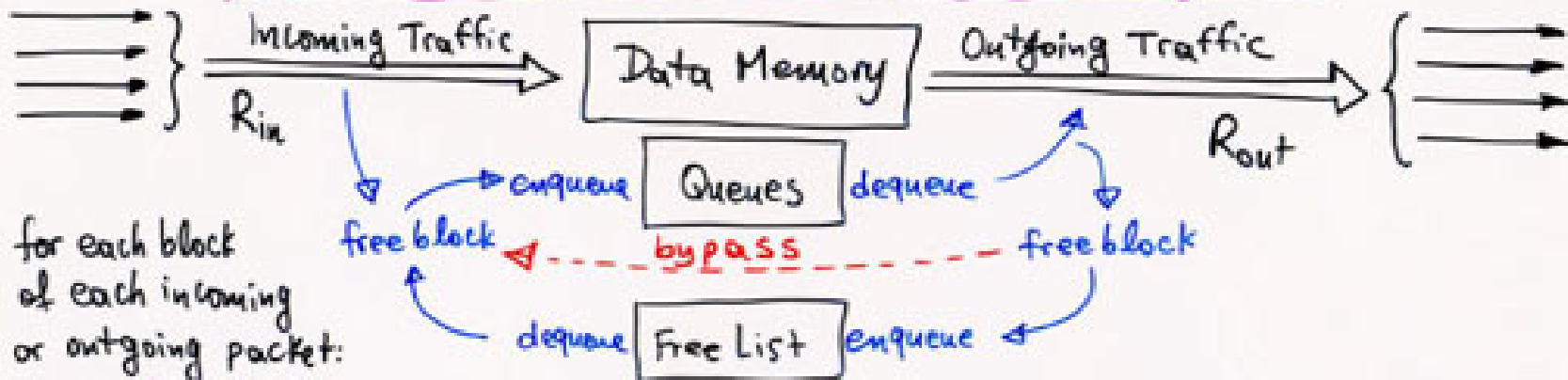
Enqueue with freeBlock preallocation:



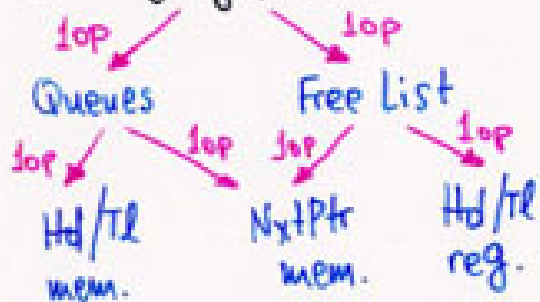
- needs only touch one block
- cost: M extra blocks for M queues

Nikologiannis, Katevenis: "Efficient Per-Flow Queueing in DRAM at OC-192 Line Rate using Out-of-Order Execution Techniques", IEEE ICC, Helsinki, Jun. 2001

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



for each block of each incoming or outgoing packet:



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

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

However, Free List Bypass can reduce this to 1.

↳ P. Andersson, C. Svensson (Lund Univ, Sweden):
 "AVLS Architecture for an 80 Gb/s ATM Switch Core",
 IEEE International 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.

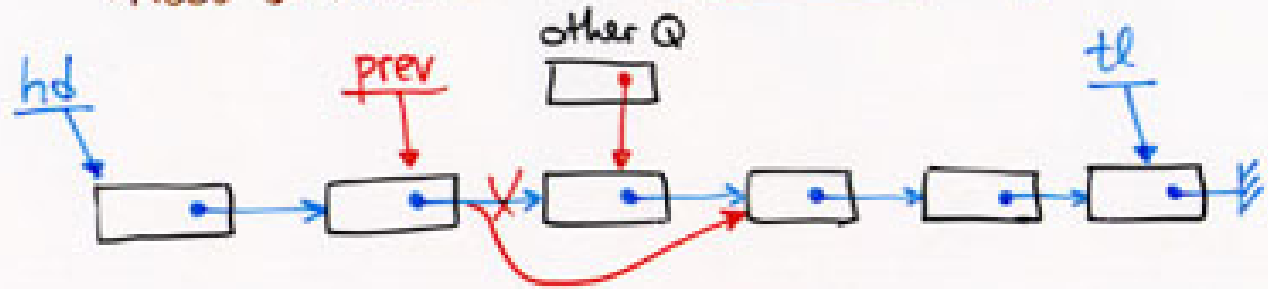
Dropping a Segment (= move to free list) or Moving a Segment to a Different Queue

① From the Head of the Source Queue:

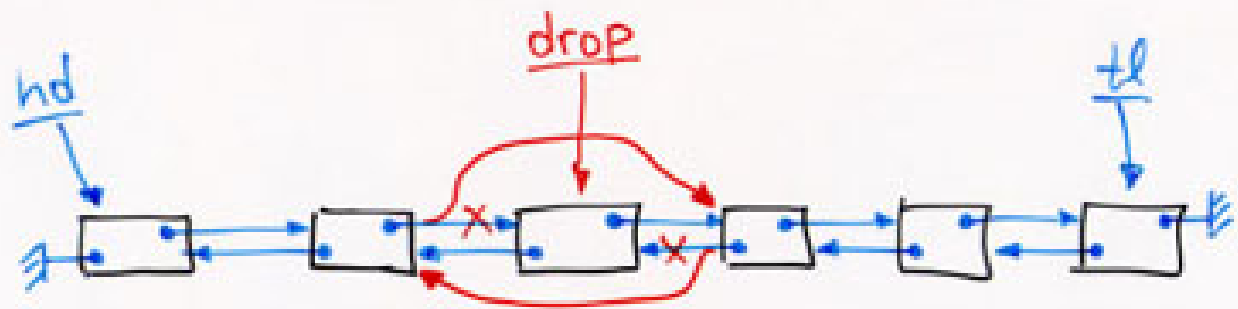
- (a) dequeue from source queue,
- (b) enqueue into destination Q.

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 $O(\text{pktSize})$ Time:

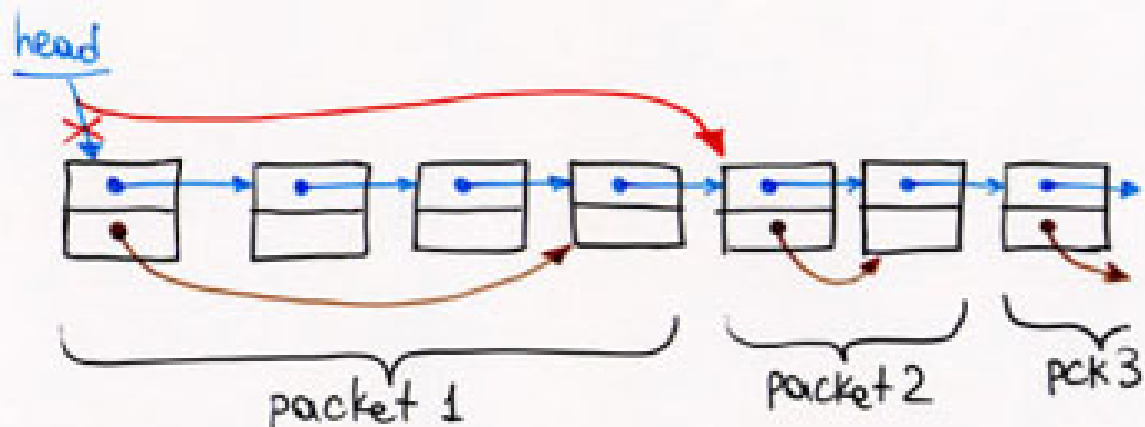
- Drop or move one segment at a time...

⇒ 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: $\frac{1}{\text{average number of segments per packet}}$)

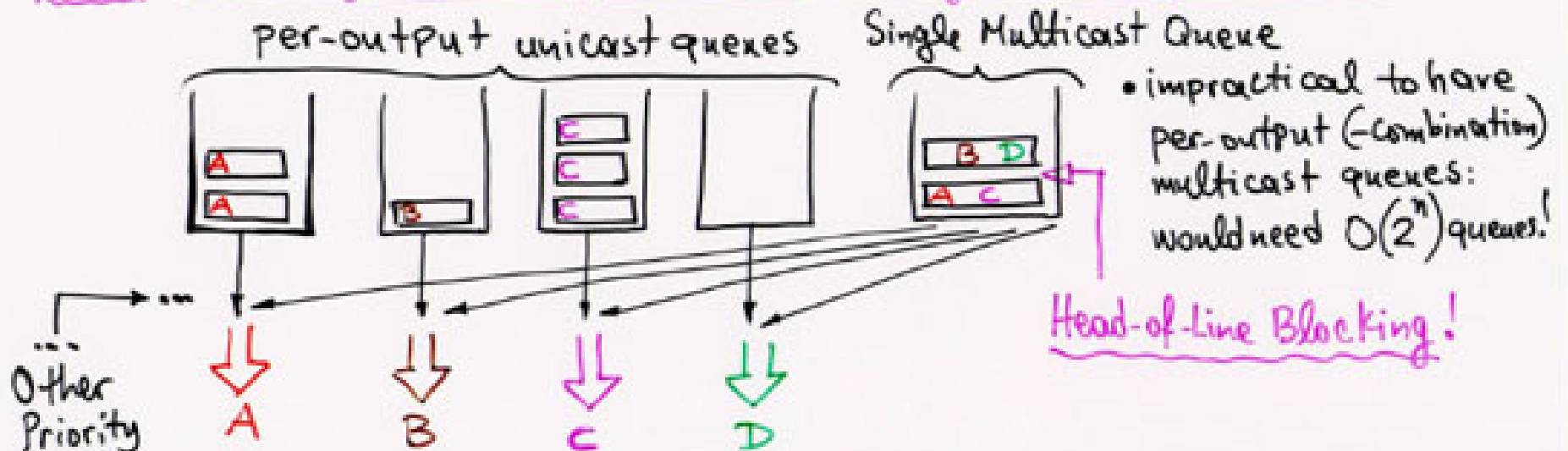
→ alternative: separate "packet descriptor" data structures, like for multicast segment descriptors - see below).

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 nextPtr's
 - Enqueue throughput and nextPtr 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?

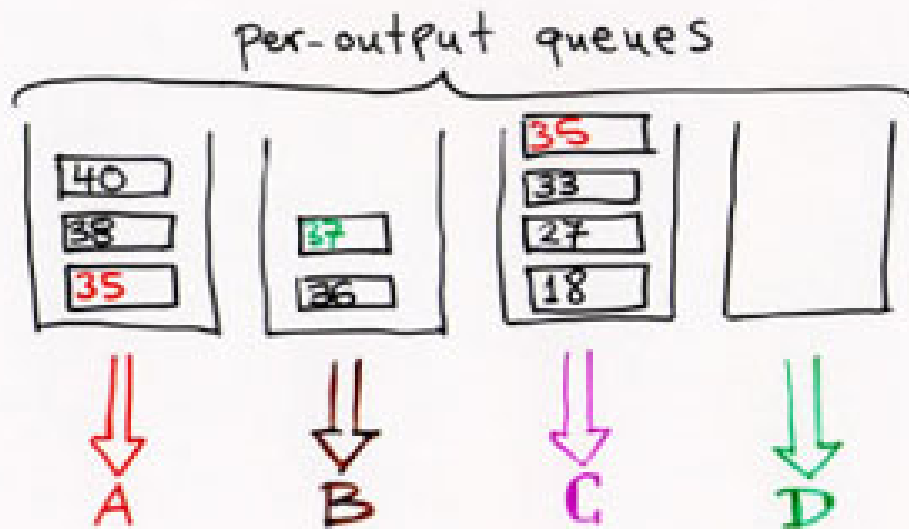
Case 1: Each segment is only allowed to belong to a single queue:



Other
Priority
Level
Queues

- 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:



Reference Counts:	Data Buffer blocks:	addr:
1	(unicast)	33
0		34
2	(multicast) AC	35
1	(unicast)	36
1	(multicast) BD	37
1	(unicast)	38
0		39
1	(unicast)	40

- Solves all QoS problems!
but...
- Increases the worst-case queue-operation rate by a factor of n
(n = number of output ports) !

↑
One copy of "37"
has already
departed, and
has decremented
the corresponding
reference count.

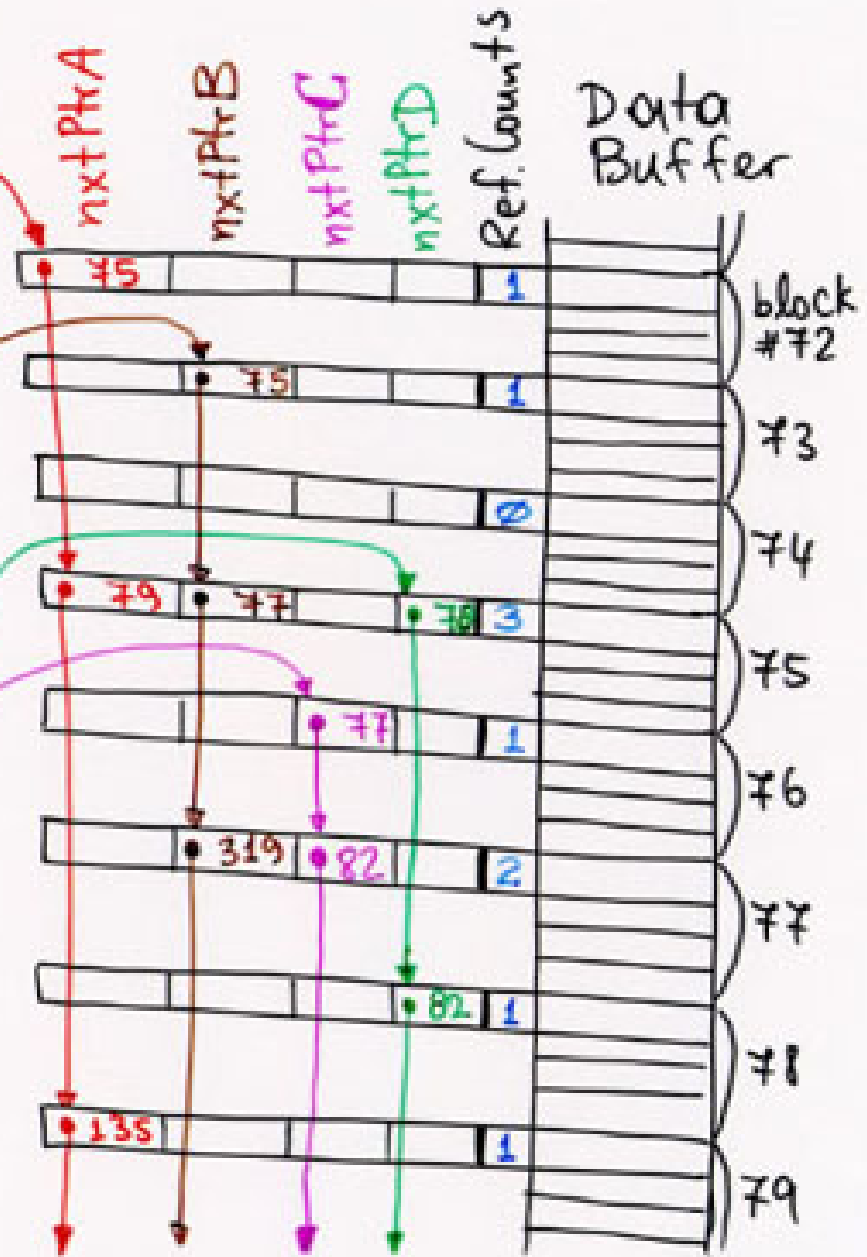
Data Structures for
a Segment to belong to
up to N Queues:

Case 1:

N nextPtr's per memory block

- most segments are unicast
 ↓
 nextPtr's are grossly underutilized!

	Head	Tail
QA1	72	
QA2		
QB1	73	
QB2		
QC1		
QC2	76	
QD1	75	
QD2		



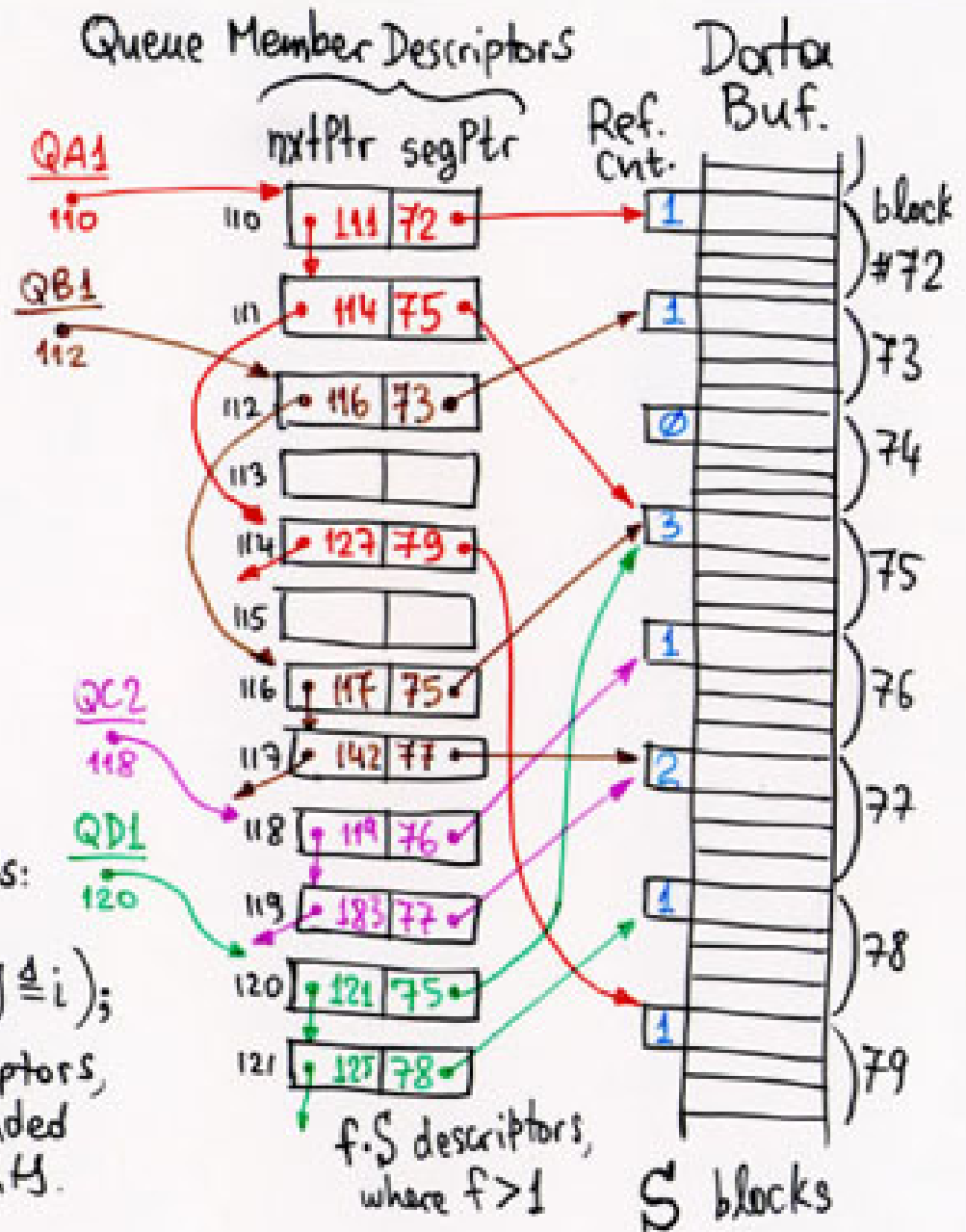
Case 2: Decouple Linked List Nodes from Data Buffer Addresses

- twice the cost per nextPtr (need a segPtr as well now) but...
- much fewer than $N \cdot S$ descriptors (based on average ratio of unicast-to-multicast segments, and average fan-out of multicast segments - e.g. $f=2$)

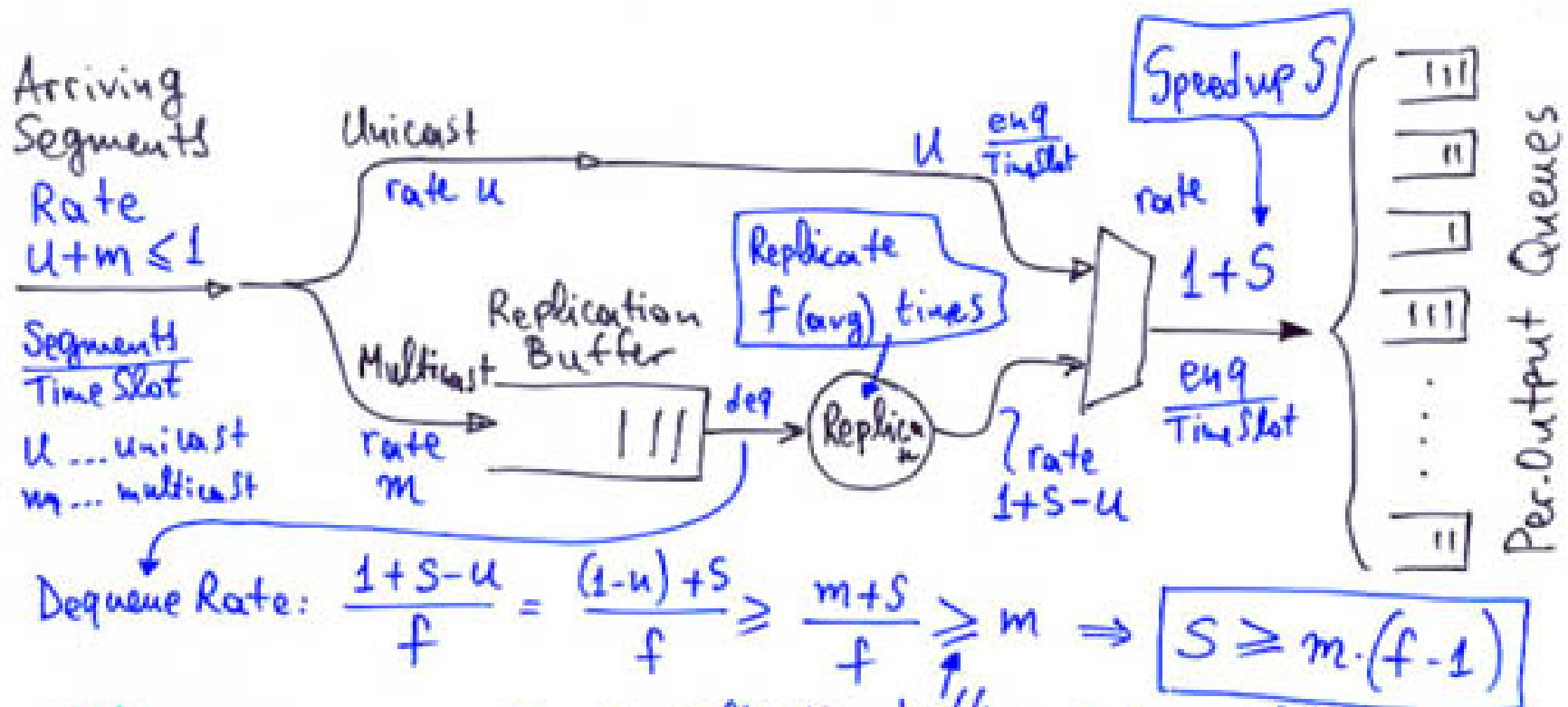
Optimization:

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

- 0 to $S-1$: unicast-only segments, no segPtr needed (segPtr $[i] \triangleq i$);
- S to $fS-1$: full queue member descriptors, with nextPtr and segPtr, intended for use by multicast segments.



Enqueue Operation Rate for multicast segments onto multiple per-output queues



References:

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