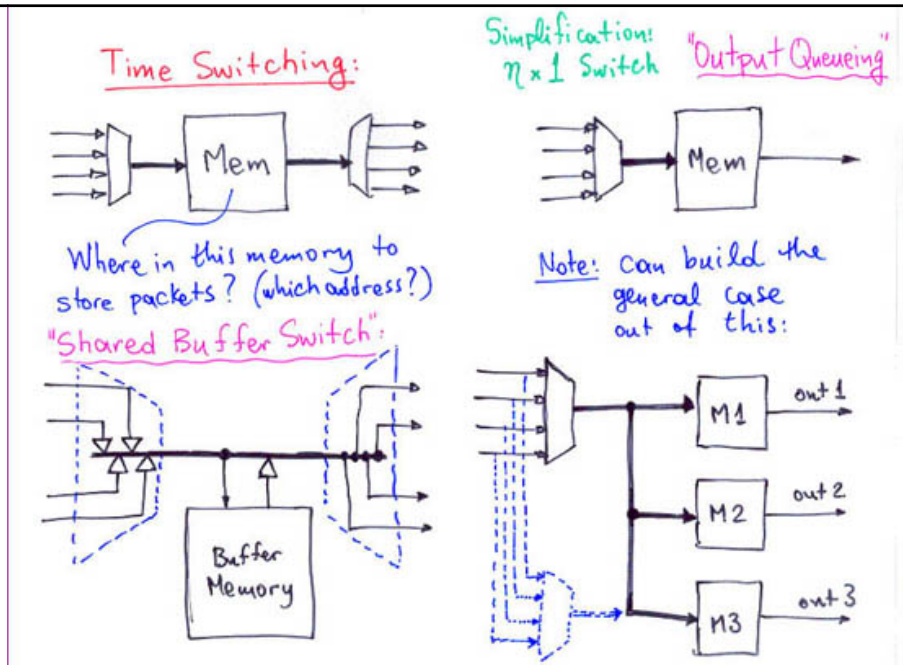
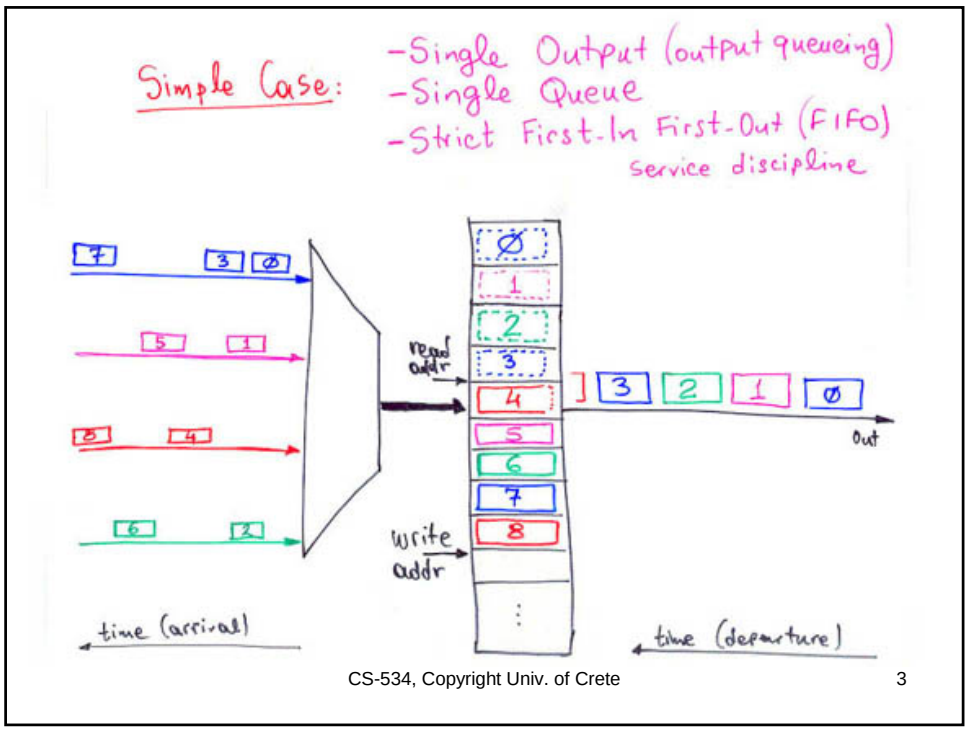
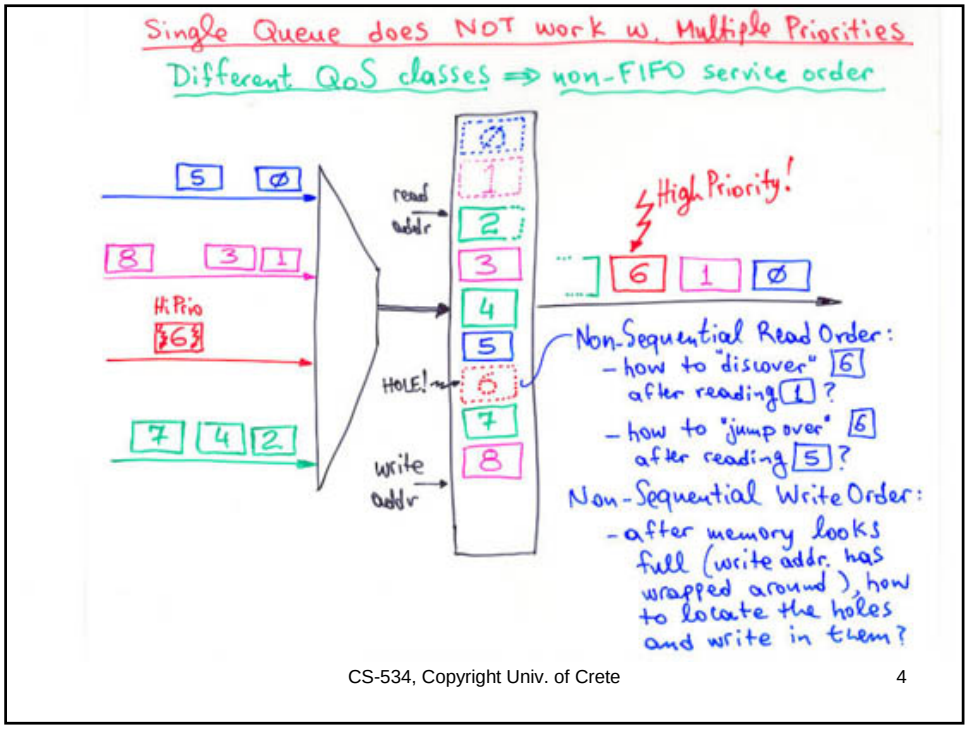
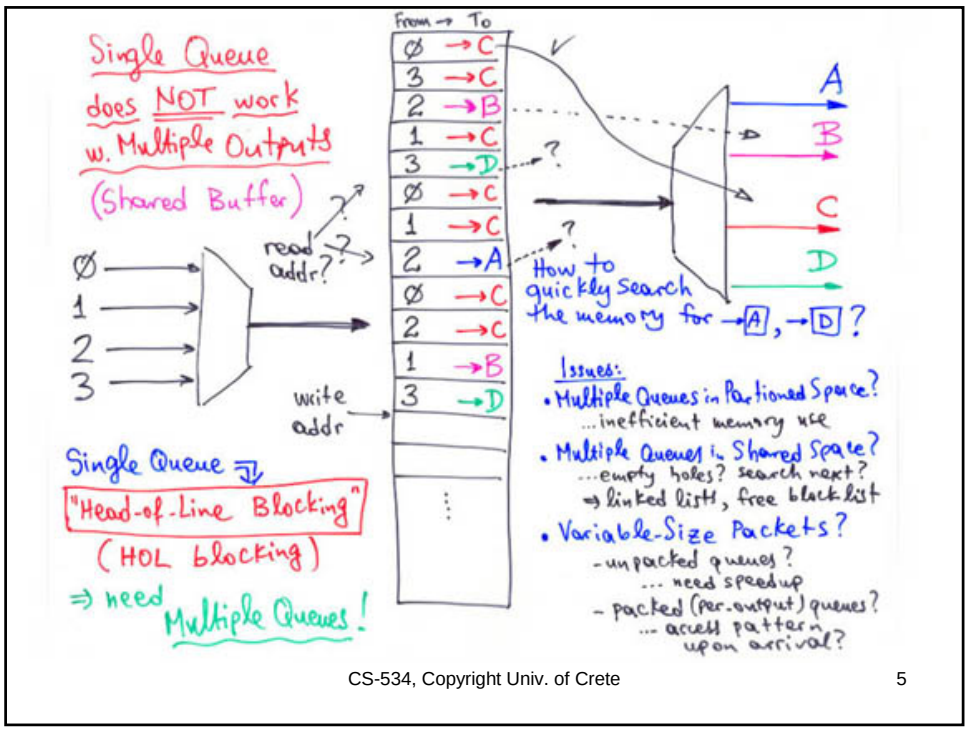
3.1 The need for multiple queues within a same buffer memory: Multi-Queue Data Strucutres

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

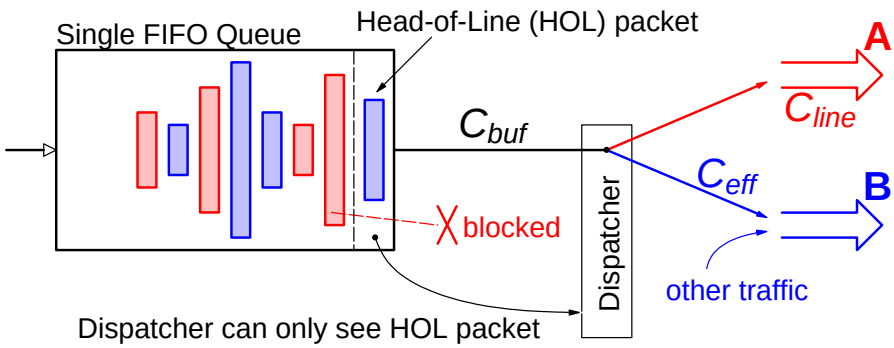



Simple Case:
-Single Output (output queueing)
-Single Queue
-Strict First-In First-Out (FIFO) service discipline
read addr
write addr
out
time (arrival)
time (departure)
CS-534, Copyright Univ. of Crete
3


Single Queue does NOT work w. Multiple Priorities
Different QoS classes ⇒ non-FIFO service order
High Priority!
read addr
write addr
HOLE!
Hi Prio
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?
CS-534, Copyright Univ. of Crete
4



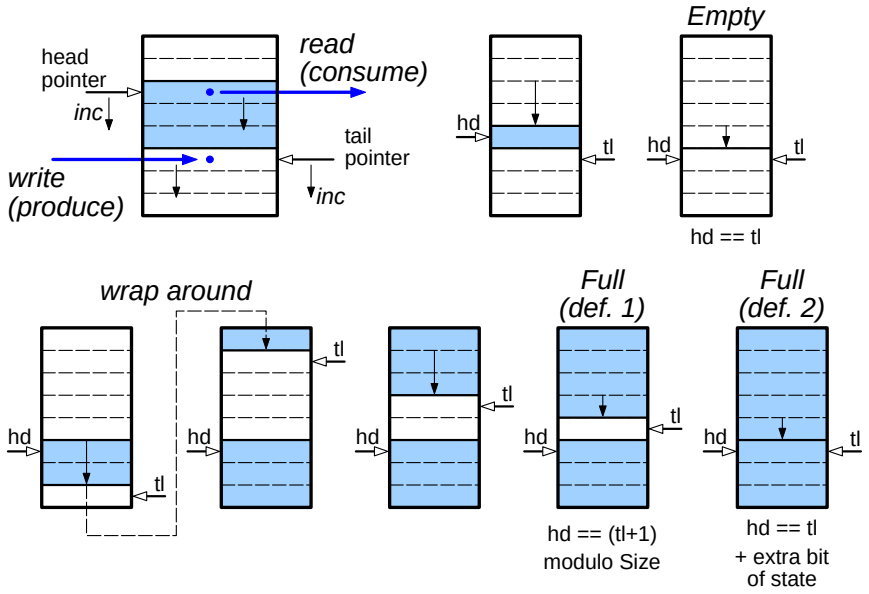
Shared Queue ⇒ Head-of-Line (HOL) Blocking



- 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 Queu

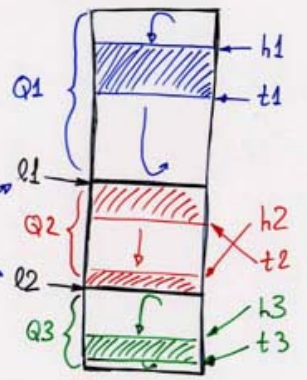


CS-534, Copyright Univ. of Crete 7

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.



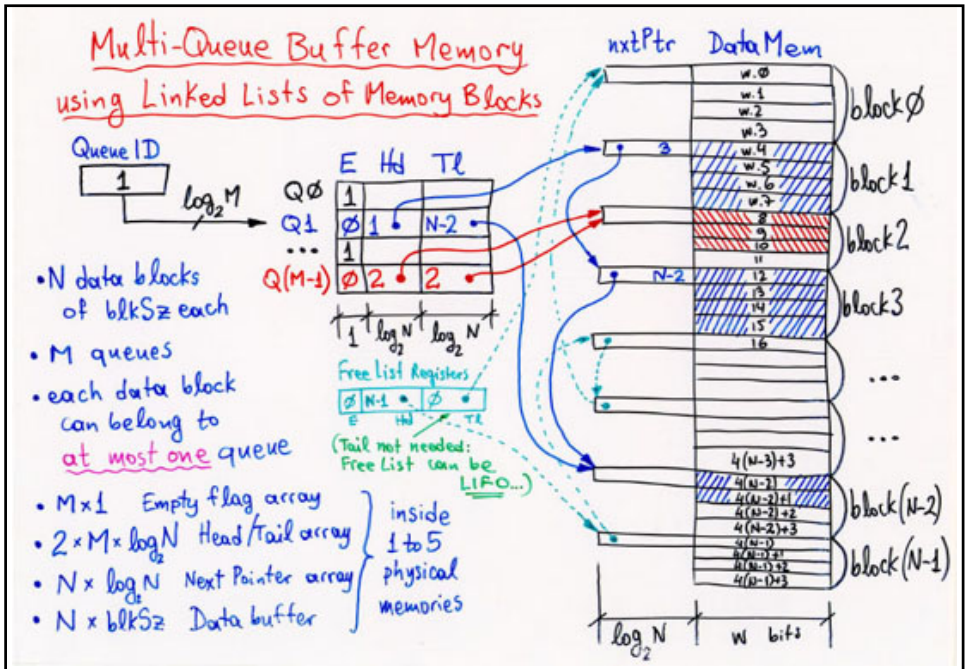
534
12.6

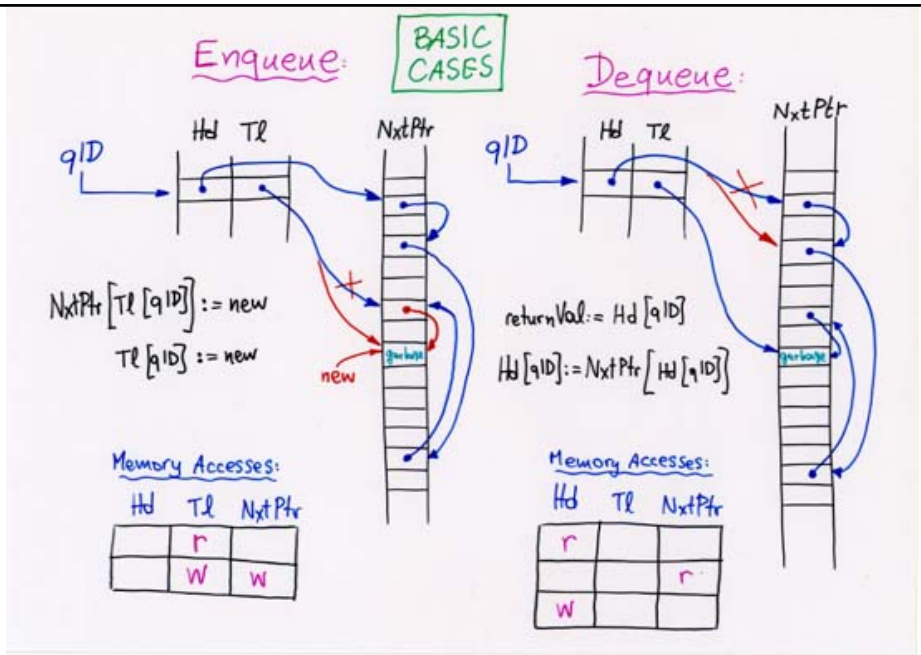
CS-534, Copyright Univ. of Crete 8

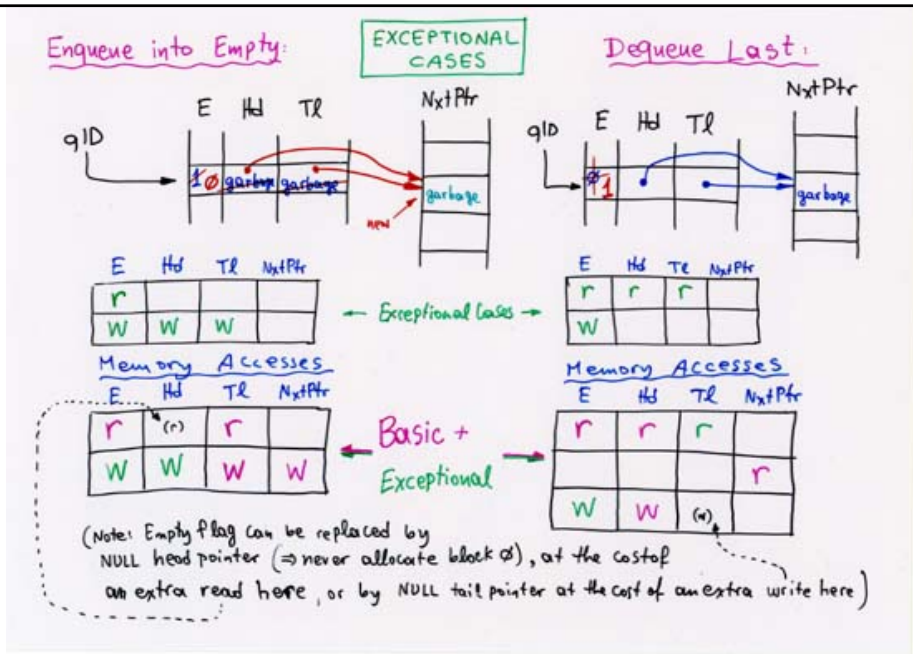
Multiple Queues Sharing a Buffer Memory Space

- Often need many queues where most have low occupancy (or empty) but few have high occupancy
- Departures create holes ⇒ empty space is fragmented
  ⇒ neighboring packets physically non-contiguous
  ⇒ 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




Enqueue:
BASIC CASES
Dequeue:
qID
Hd Tl
NxtPtr
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
E Hd Tl NxtPtr
r
w w w
Exceptional Cases
E Hd Tl NxtPtr
r r r
w
Memory Accesses
E Hd Tl NxtPtr
r (r) r
w w w w
Basic + Exceptional
Memory Accesses
E Hd Tl NxtPtr
r r r
r
w w (w)
(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 (op's/cycle)	Dequeue Latency (clock cycles)	Dequeue Thruput (op's/cycle)
	in a single, 1-port off-chip memory			4 or 5	1/(4 or 5)	5	1/(5 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

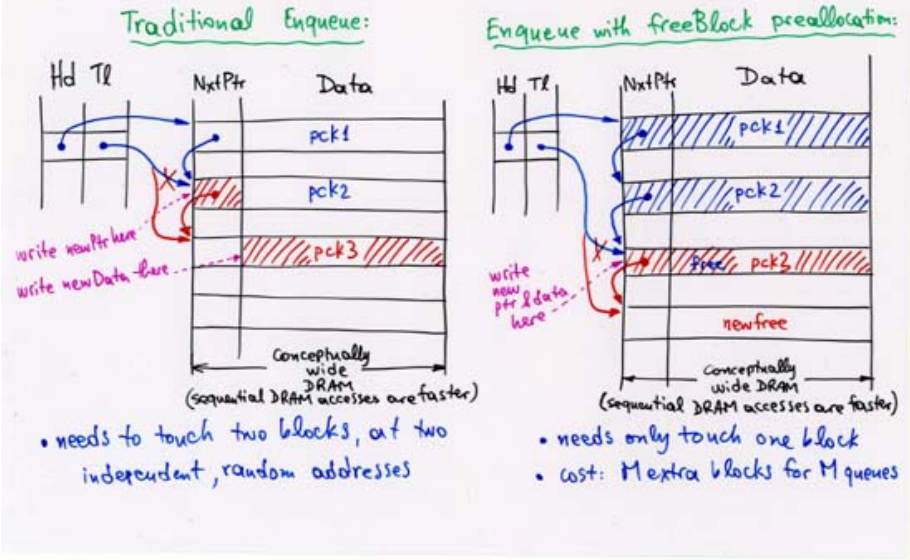
Note:
* enqueue latency may increase when mixing enqueue and dequeue operations in the same pipeline.

Notes on Enqueue/Dequeue Performance Table

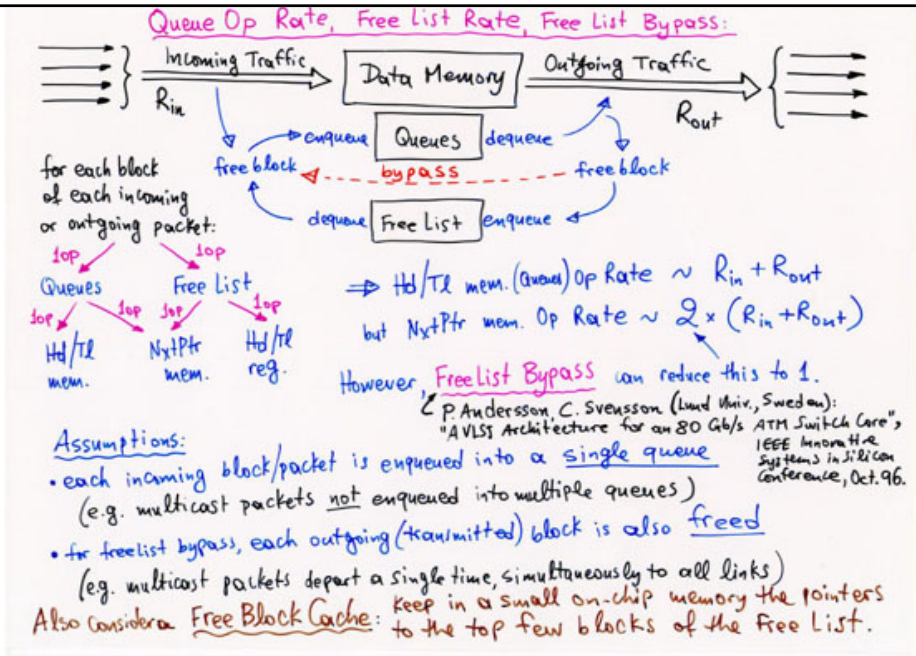
- Table assumes we always need to perform both normal and exceptional accesses – see ex. 5.1
- Table assumes fall-through timing for off-chip SRAM: one-cycle latency per access, plus one extra cycle between dependent off-chip accesses – see ex. 5.2
- For peak throughput: overlap successive operations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



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



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

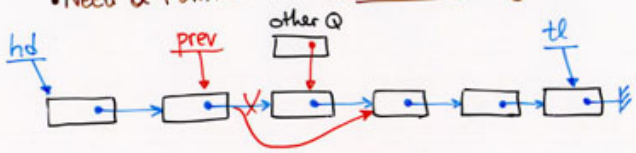
① From the Head of the Source Queue:

(a) dequeue from source queue,

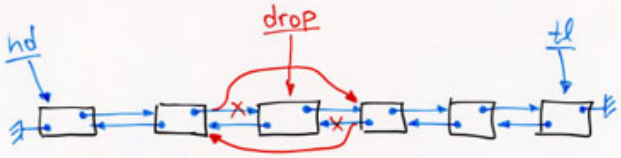
(b) enqueue into destination Q.

② From the "Middle" of the Source Queue

• Need a Pointer to the Previous Segment:



else • Need Doubly Linked List:



CS-534, Copyright Univ. of Crete 17

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

① In O(pckSize) Time:

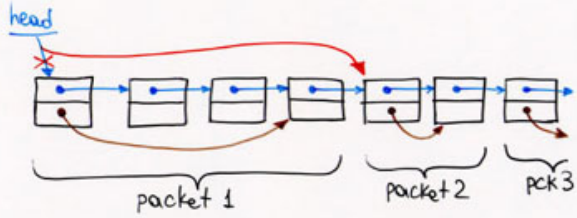
• Drop or move one segment at a time...

⇒ 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:



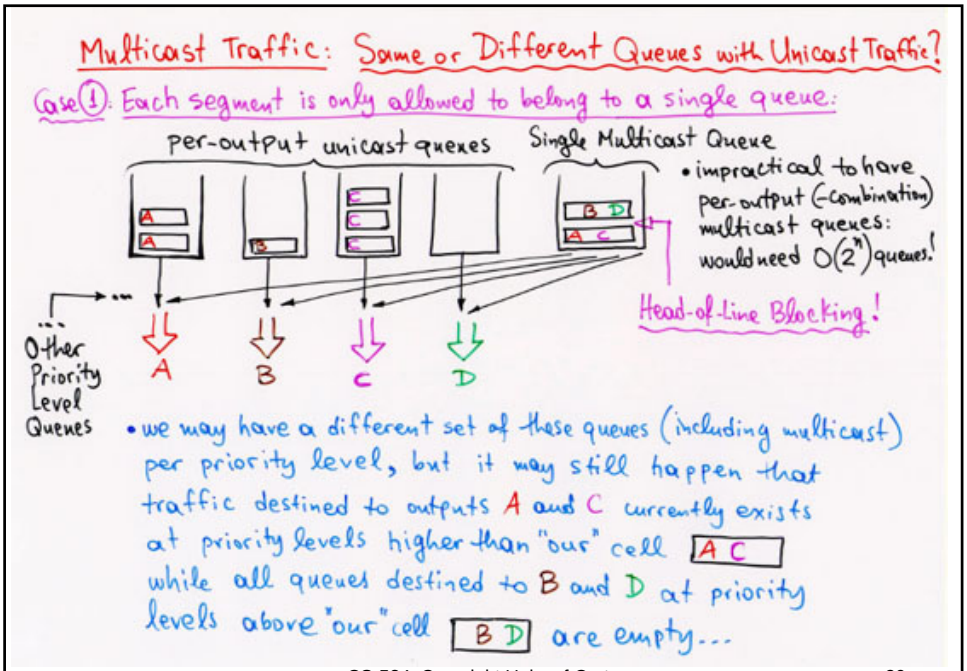
(the utilization of the second set of pointers is only: $\left(\frac{1}{\text{average number of segments per packet}}\right)$

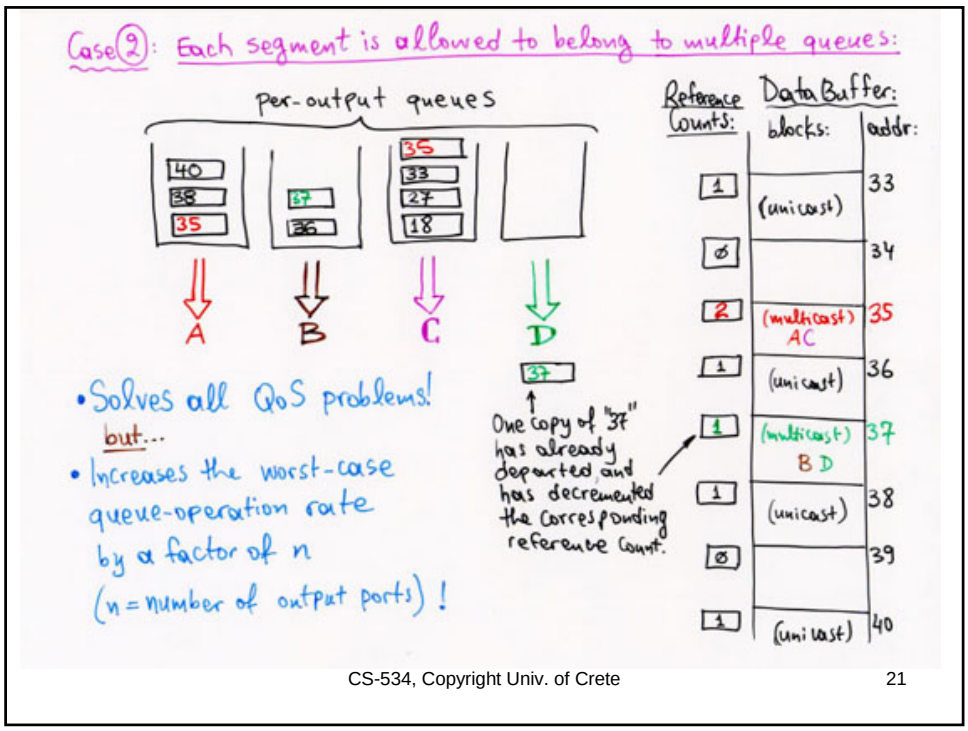
→ alternative: separate "packet descriptor" data structures, like for multicast segment descriptors — see below).

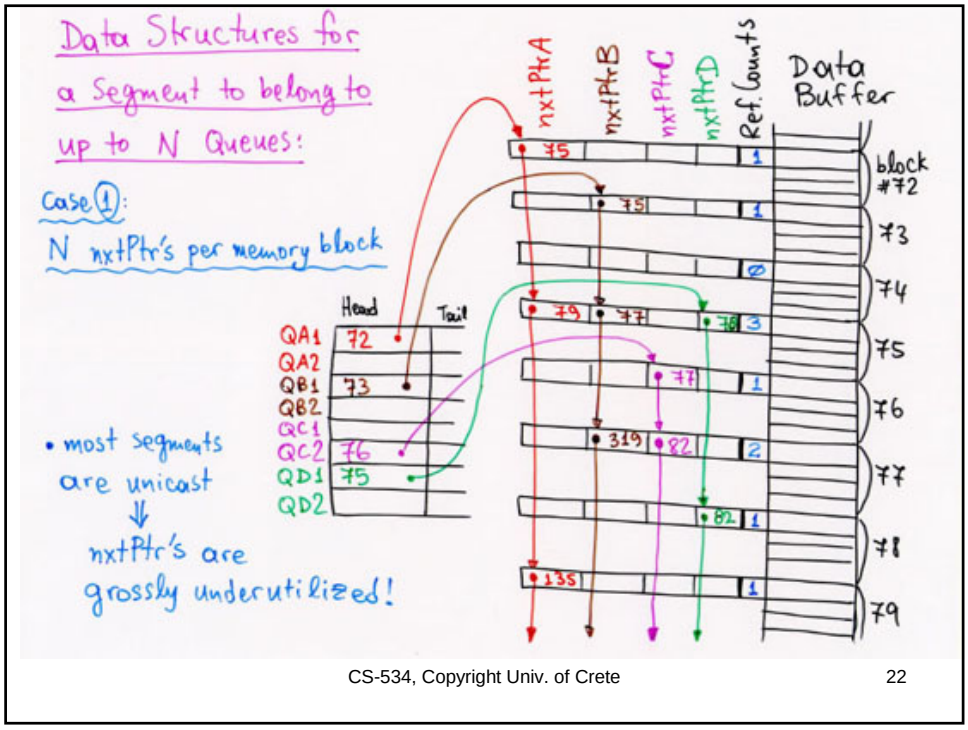
CS-534, Copyright Univ. of Crete 18

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 ⇒ HOL blocking
  - Each segment allowed in multiple queues ⇒ 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.




Case 2: Each segment is allowed to belong to multiple queues:
Per-output queues
40
38
35
37
36
35
33
27
18
A
B
C
D
37
One copy of "37" has already departed and has decremented the corresponding reference count.
Reference Counts:
Data Buffer:
blocks:
addr:
1
(unicast)
33
0
34
2
(multicast) AC
35
1
(unicast)
36
1
(multicast) B D
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) !


Data Structures for a Segment to belong to up to N Queues:
Case 1:
N nxtPtr's per memory block
nxtPtrA
nxtPtrB
nxtPtrC
nxtPtrD
Ref. Counts
Data Buffer
Head
Tail
QA1 72
QA2
QB1 73
QB2
QC1
QC2 76
QD1 75
QD2
75
1
75
1
0
79
77
76
3
77
1
319
82
2
82
1
135
1
block #72
73
74
75
76
77
78
79
• most segments are unicast
⇓
nxtPtr's are grossly underutilized!

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

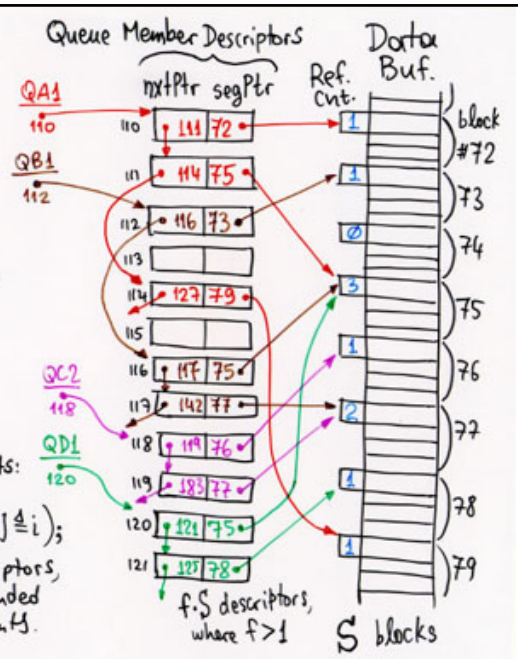
- twice the cost per nxtPtr (need a segPtr as well now)

but...

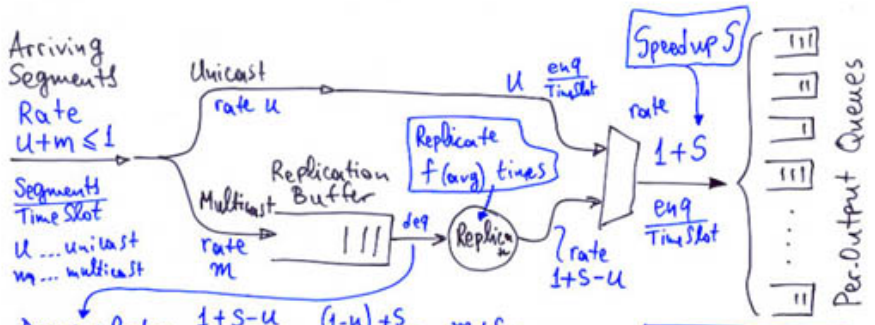
- much fewer than N·S descriptors (based on average ratio of unicast-to-multicast segments, and average fan-out of multicast segments – e.g. f=2)

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