3. Time Switching, Multi-Queue Memories, Shared Buffers, Output Queueing Family

Manolis Katevenis

CS-534 – Univ. of Crete and FORTH, Greece

`http://archvlsi.ics.forth.gr/~kateveni/534`

3.3 Multiple Queues 3.4 Multicast Queues

Table of Contents:

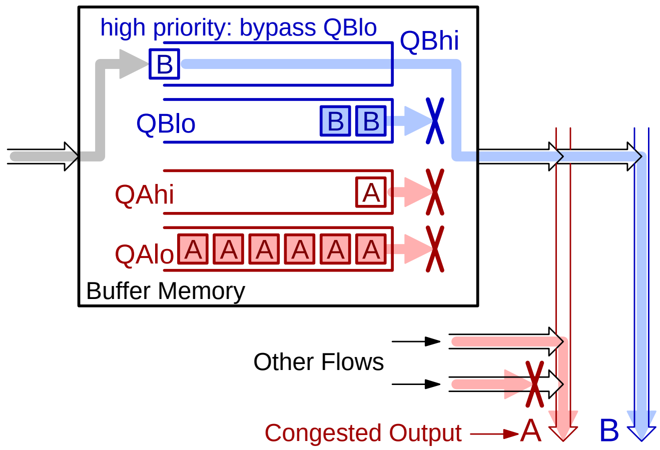
- **3.3 Multiple Queues within a Buffer Memory**
  - partitioned queue space: circular-buffer queue
  - shared queue space: linked-list queues
  - DRAM optimizations, free-list bypass / free-block cache
- **3.4 Queueing for Multicast Traffic**
  - each segment allowed in single queue
  - each segment allowed in multiple queues
  - decoupled linked-list node from data-block addresses

3.3 Multiple Queues within a Buffer Memory

Separate Destinations & Priorities ⇒ Multiple Queues

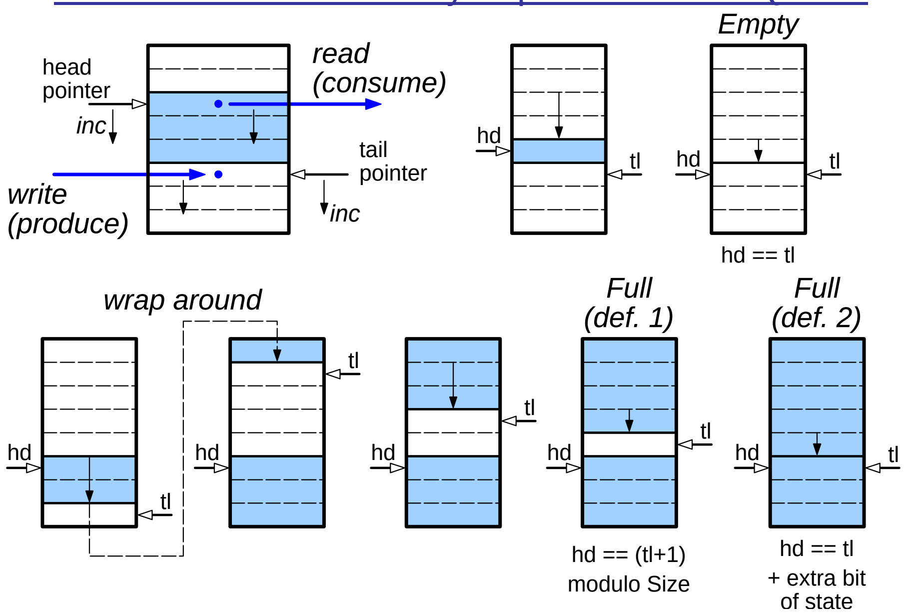
- Switch controller must have access to any packet that is candidate to depart next

⇒ Packets that are allowed to bypass others cannot reside in the same FIFO structure



- Controller needs separate *per-destination* and *per-priority* queue (FIFO) data structures to keep track of packets

Reminder: Circular Array Implem. of FIFO Queue



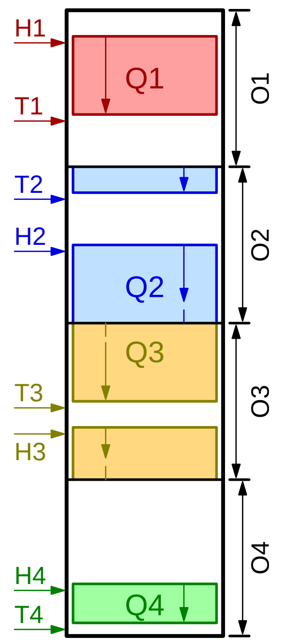
3.3 Multiple Queues – 1 of 2

Statically Partitioned Space

- Multiple queues within a same SRAM block
- Each queue: circular array implementation
- Control overhead: two pointer words per queue (head, tail), incrementor, comparator
- Queue space bounds (partitions) can be hardwired, or off-line configurable (when queues are empty); in the latter case, also need bounds pointers.

\+ Advantage: ***simplicity***.

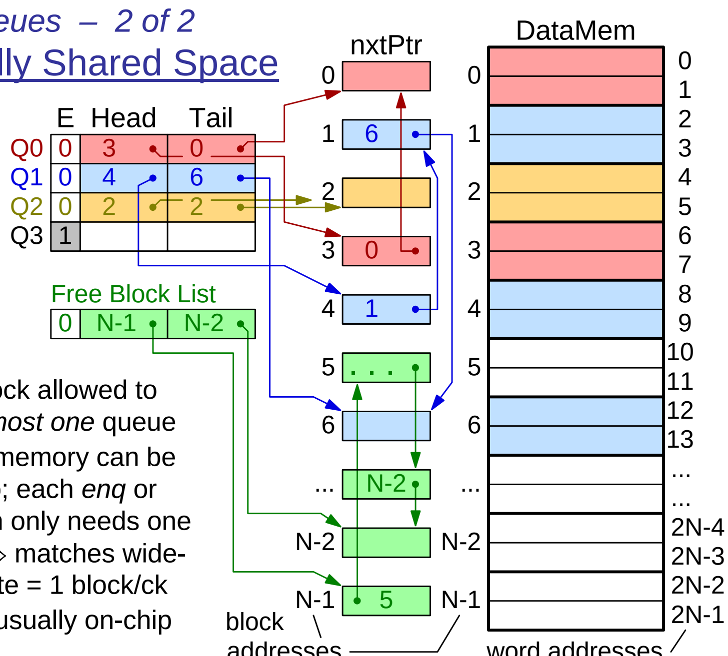
\- Disadvantage: ***partitioned*** memory space leads to ***underutilization*** – one queue may overflow while lots of empty space exists in other memory space partitions.



Multiple Queues – 2 of 2

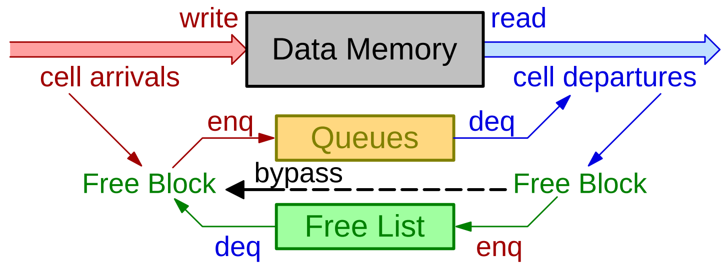
Dynamically Shared Space

- Linked List implem. of queues
- Pointers in separate memories: accessed in parallel



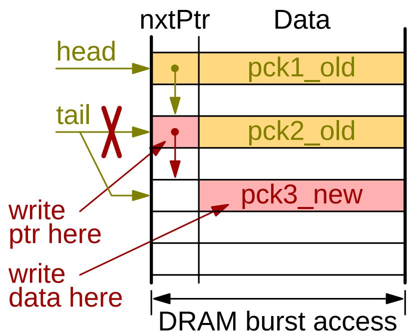
- Each data block allowed to belong in *at most one* queue
- Next-pointer memory can be large, off-chip; each *enq* or *deq* operation only needs one access to it ⇒ matches wide-mem. data rate = 1 block/ck
- *Empty/Hd/Tl* usually on-chip

Data vs. Pointer Access Rate – Free List Bypass

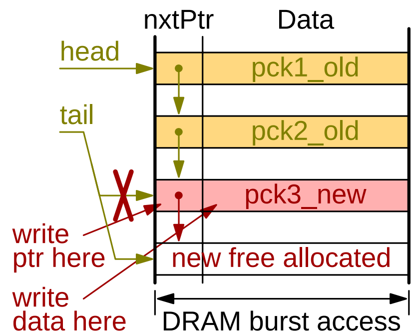


- Data memory throughput = 2 cells/cell-time (1 write + 1 read)
  ⇒ data memory access rate = 2 addresses/cell-time
- Both Queue & Free-List operations touch the Next-Pointers, once per op
  ⇒ naïve implementation would require 4 addresses/cell-time to *nxtPtr*
- *Free List Bypass:* put incoming cell into just freed block of departing cell
  ⇒ next -pointer memory access rate = 2 addresses/cell-time
- When no arrival or no departure, other side can use full 2 acc/cl-time rate
- Multicast: departure not always frees the block ⇒ use *Free Block Cache*

nxtPtr in DRAM – Free Block Preallocation



Conventional Enqueue



Enq. w. Free-Block Preallocation

- To economize on *nxtPtr* memory, place these pointers inside data DRAM
  ⇒ conventional *enq* costs twice the number of DRAM row *activate*'s
- Preallocate one free block per queue, at tail, to remedy this
- Reference: Nikologiannis, Katevenis: "Efficient per-flow queueing in DRAM at OC-192 line rate using out-of-order execution…", IEEE Int. Conf. Commun. (ICC) 2001.

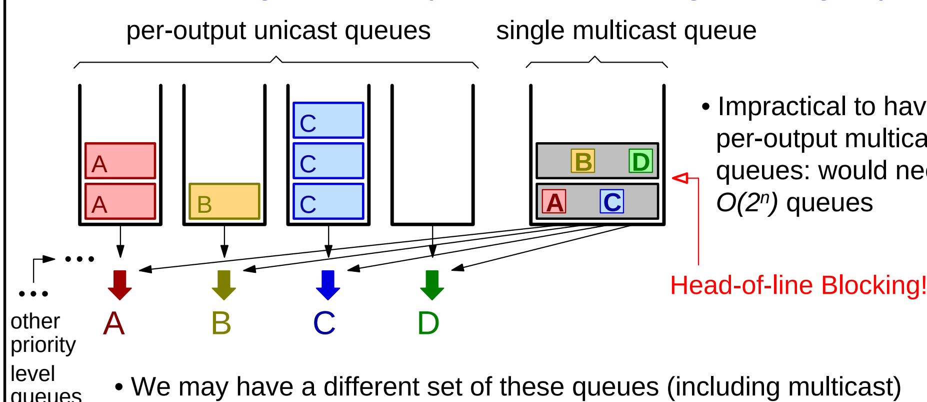
3.4 Queueing for Multicast Traffic

- 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.

3.4 - U.Crete - M. Katevenis - CS-534 9

Same or Different Queues with Unicast Traffic?

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

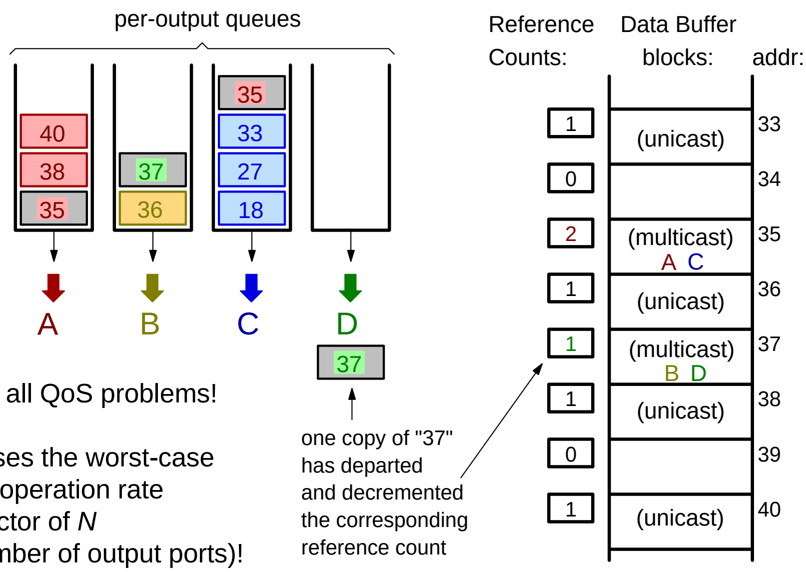


- Impractical to have per-output multicast queues: would need $O(2^n)$ queues

- 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.

3.4 - U.Crete - M. Katevenis - CS-534 10

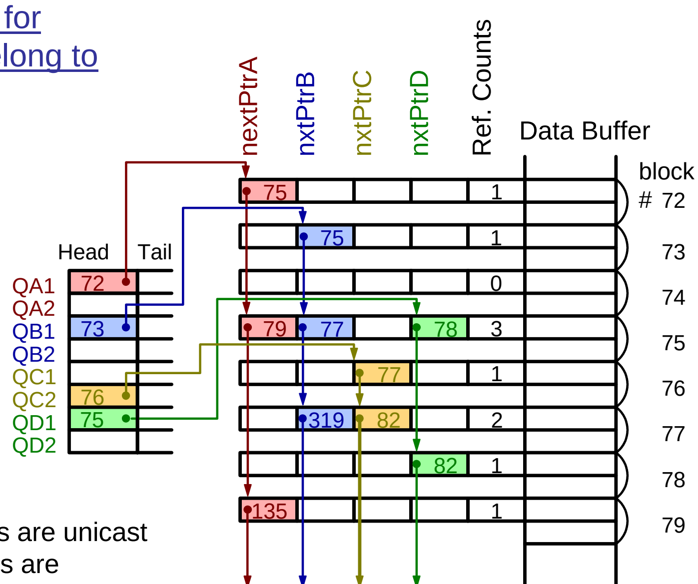
Case 2: Each segment is allowed to belong to multiple queues



- 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 2A: N nxtPtr's per memory block

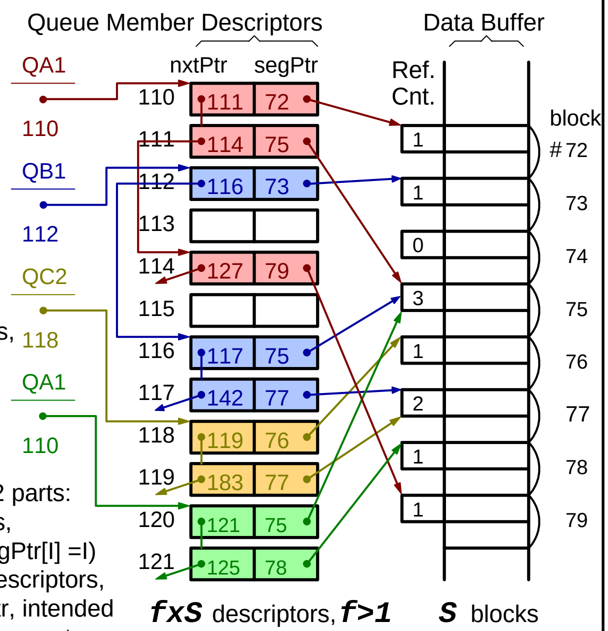


- Most segments are unicast → next pointers are grossly underutilized!

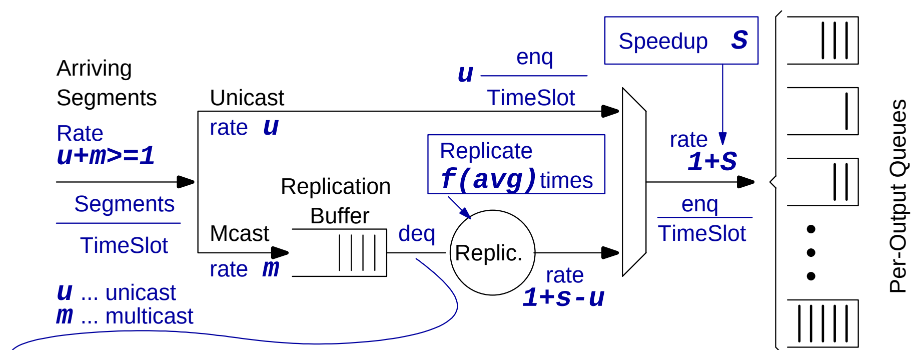
Case 2B: Decouple Linked List Nodes From Data Buffer Addresses

- twice the cost per nxtPtr (need a segPtr as well now) *but* …
- Much fewer than NxS descriptors (based on avg ratio of unicast-to-multicast segments, and avg fan-out of multicast segments, e.g. f =2)

Optimization:
Partition the address space of queue member descriptors into 2 parts:
- 0 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 to use by multicast segments



Enqueue operation rate for multicast segments into multiple per output queues



Dequeue Rate: $\frac{1+s-u}{f} = \frac{(1-u)+s}{f} >= \frac{m+s}{f} >= m \Rightarrow S >= m(f-1)$

•References:

–F. Chiussi, Y. Xia, V. Kumar: IEEE JSAC, April 1997, pp. 473-487.

–D. Stiliadis: IEEE HPSR 2003, June 2003, pp. 117-122.