CS-534: Packet Switch Architecture
Spring 2007

Department of Computer Science

Exercise Set 7:
Multi-Packet Queue Blocks, Multicast Queues

Assigned: Wed. 9 May 2007 (week 7) - Due: Wed. 16 May 2007 (week 8)

7.1 Multi-Packet Block Segmentation for Linked-List Queues

In the usual systems where variable-size packets are kept in multiple queues in a buffer memory, using linked lists of memory blocks (segments) as in section 3.1, each memory block (segment) contains a *single* packet, or fragment thereof. In the first block of each packet, the beginning of the packet is aligned with the beginning of the block; the last block of each packet may contain empty space, from some point to its end, even if it is not the last block on its queue; data from two different packets can never appear inside a same block. For such systems, we studied the segmentation of packets into blocks in exercise set 4, and concluded that the memory block (segment) size is chosen, usually, on the same order or slightly larger than the minimum packet size.

In this exercise, you will study an alternative scheme, which we can name *Multi-Packet Block* segmentation: data from multiple packets can co-exist inside a same memory block (segment). The packets that belong to a same queue are stored *contiguously* after one another, starting in the next byte of the same memory block after the last byte of their previous packet, whenever that previous packet finishes before the end of a block. In such systems, we can afford relatively large memory block sizes, because the only unused space is at the beginning of the first block of each queue and at the end of the last block of each queue --not at the end of the last block of each packet as in the usual, single-packet per block scheme. (If the number of queues is very large, on the same order as the number of packets in the buffer memory, then this scheme makes little sense). Multi-packet blocks (called *Envelopes*) were used in the paper:

- K. Kar, T.V. Lakshman, D. Stiliadis, L.Tassiulas: "Reduced Complexity Input Buffered Switches", *Proc. Hot Interconnects VIII,* 2000; http://www.bell-labs.com/user/stiliadi/

Assume that no special marker or pointer is needed to indicate the end of each packet (and hence the beginning of the next packet), because each packet includes enough information to indicate its precise length (e.g. a length field, or a special end-of-packet pattern inside its data). This is important, given that the number of packet boundaries inside a single memory block may be potentially large, hence keeping end-of-packet markers or pointers in the queue data structures would be akward. Thus, the only management data needed per memory block is a pointer to the next block on its queue. The head and tail pointers of each queue, on the other hand, need to be full pointers into the buffer memory, pointing to a specific byte position inside a specific block --not merely pointers to (the beginning of) a block.

Analyze the time cost of this scheme: is it better, worse, or similar to the cost of the single-packet per block scheme? By "time cost" we mean the number of management operations and memory accesses needed when enqueueing an incoming packet into a queue or when dequeueing a departing packet from a queue. Remember that consecutively arriving or departing packets will, in general, be enqueued to or dequeued from different queues each; the arrival or departure time of a variable-size packet will, in general, be proportional to its size --see exercise set 4 on this. Include the free list operations in your analysis. Analyze especially the worst case, and try to analyze the average case too.

7.2 Descriptor Memory Space needed for Multicast QoS Support

In section 3.1, we saw that the provision of quality of service for multicast traffic requires that it be possible for a segment to reside simultaneously in multiple (per-output) queues. Two schemes were considered for implementing this: *(i)* dedicated per-output and per-segment next-pointers (nxtPtr), or *(ii)* queue-member descriptors that are decoupled from the segment addresses.

Consider a shared-buffer switch with 16 output ports. The buffer memory is made of 16 synchronous DRAM chips, of size 2 M x 32 bits each, like the MT46V2M32 DDR SDRAM chip example in section 2.1. The segment size is 128 bytes. What is the size of the buffer memory, in MBytes, and what is the number of segments in it? How wide (how many bits) is a segment pointer? How many reference counts and how wide each do we need?

(a) According to the first scheme --dedicated per-output and per-segment next-pointers-- how many nxtPtr's will the system need to store? What is the size, in Mbits, of the memory required to store these pointers plus the required reference counts? Assume that this memory is built out of DDR static RAM chips that support burst-of-4 accesses only, of size 9 Mbits (256 K x 36 bits) per chip, like the MT57V256H36 DDR SRAM chip example in section 2.1. Further assume that the width of this memory should be such that each access touches all nxtPtr's and the reference count associated with one segment (remember that each access yields a burst of 4 words from each chip). How wide (how many chips) does this memory need to be? Note that the memory width must be an integer number of chips, but this number does not need to be a power of two. How deep does the memory need to be? How many SRAM chips do we need in total?

(b) Now consider the second scheme --queue-member descriptors that are decoupled from segment addresses. We choose to have a number of descriptors that is twice the number of buffer memory segments. How many descriptors will we have? What is the width (number of bits) of a descriptor pointer? How large (number of bits) is a descriptor? What is the size, in Mbits, of the memory required to store these descriptors plus the required reference counts? Assuming that this memory is also built out of 9 Mbit SRAM chips, and that a "nice" packing of descriptors and reference counts into memory words can be found, what is the (minimum) required number of chips for this memory?

(c) Assuming that the average fan-out of multicast segments is 4, what is the maximum percentage of multicast segments in the buffer memory, so that we do not run out of descriptors before we run out of buffer space?

(Hint for *optional* thought on question (b), for those interested: a "nice" packing of descriptors and reference counts into memory words could be a placement such that the number of descriptors per burst is not a power of 2. In that case, to avoid the need for a divider in the addressing circuit, we could make descriptor pointers to be true burst-number and burst-offset pointers, and agree that not all integer numbers are valid descriptor pointers; as long as the free list of descriptors is correctly initialized to only contain the valid descriptors, no one else should ever care...)

Up to the Home Page of CS-534

Last updated: 14 May 2007, by M. Katevenis.