CS-534: Packet Switch Architecture
Spring 2004

Exercise Set 3: Variable-Size-Packet Segmentation Overhead

Assigned: Fri. 12 March 2004 (week 3) - Due: Fri. 19 March 2004 (week 4)

3.1 Variable-Size-Packet Bit Rate for given Segment Access Rate

Consider that the 64-Byte-wide buffer memory of exercise 2.3 is used to store incoming (fixed-size) ATM cells, or (variable-size) IP packets that are being segmented into 64-Byte segments, as well as to later read such cells or segments on their way out. Memory utilization is precisely 50% writes and 50% reads; for the SRAM technologies that have a DQ-bus turn-around penalty, we perform the optimization of arranging read/write accesses in the following fashion: precisely four (4) segments are written consecutively (at 4 arbitrary addresses), then precisely four (4) segments are read consecutively (from 4 arbitrary addresses), then 4 other segments are written, etc.

(a) For each SRAM technology in exercise 2.3(a), what is the peak **incoming segment rate** that can be supported, in Msegments/s? *Hint:* Each incoming segment is written into a "random" memory location (address). Thus, for each incoming segment we need to perform an (independent) write memory access. Hence, the peak incoming segment rate that can be supported is one half (50% writes - 50% reads) of the peak (independent) access rate calculated in exercise 2.3(a), except for technologies that have a DQ-bus turn-around penalty where you need to derate their peak Maccesses/s by the turn-around overhead for our specific 4-write-4-read access pattern.

(b) Assume that the incoming traffic is ATM over SONET. For reasons of simplicity of memory management, each ATM cell is written into a different memory segment --hence, approximately 64-53 = 11 bytes in each segment remain unused (the exact number depends on details such as whether the header CRC is stored or just recomputed on the way out, whether any flow ID is stored together with the cell to assist in VP/VC translation in the outgoing path, etc). Thus, the peak incoming cell rate that can be supported is equal to the peak incoming segment rate that you calculated in question (a).

Translate this cell rate into an equivalent "SONET bit rate", for each SRAM technology considered in (a). Of course, SONET bit rates are strictly quantized, as listed in exercise 1.1, but, for the purposes of this exercise, assume that you can linearly scale the SONET bit rate to any number that is needed to provide the desired ATM cell rate; Assume that the percentage of SONET bit rate that is dedicated to SONET overhead (clock recovery, framing, etc) is as in exercise 1.2, i.e. 3.33 percent (3 bytes of overhead in every 90 SONET bytes). Compare the "SONET bit rate" that you find here to the buffer memory aggregate peak throughput in Gbits/s that you found in exercise 2.3(b), for each same technology. How and why do they differ?

(c) Assume, now, that the incoming traffic consists of 40-Byte (minimum sized) IP packets, which are carried in an "IP-over-SONET" technology (**not** IP-over-ATM-over-SONET). These minimum sized IP packets fit within one buffer memory segment (64 bytes), each. For reasons of simplicity of memory management, again, each such IP packet is written into a different memory segment --hence, approximately 64-40 = 24 bytes in each segment remain unused. Thus, the peak incoming packet rate that can be supported is equal to the peak incoming segment rate of question (a), or to the peak incoming cell rate of question (b).

Translate this packet rate into an equivalent "SONET bit rate", for each SRAM technology considered in (a). Unfortunately, I do not know the exact format of IP-over-SONET, so let us assume, for the purposes of this exercise, that the only SONET overhead, above and beyond the 40 bytes times 8 bits/byte = 320 bits of IP packet payload, is the same as for ATM over SONET, i.e. 3 bytes of overhead for every 87 payload bytes in every 90 SONET bytes (BEWARE: do **not** use this number in any real design of yours, because it is most probably **not** the real number!). Also, assume again, contrary to reality, that SONET bit rates are not quantized, and can scale linearly to provide the desired packet rate. Compare the bit rates that you find here to those of question (b) and to those of exercise 2.3(b), and explain the difference.

(d) Next, assume that the incoming traffic consists of 68-Byte IP packets. This is a "bad" size for our buffer memory, because it is just above our segment size (we assume that IP packet sizes are multiples of 4 bytes, otherwise, 65 bytes would be the worst size in this case). In this case, each IP packet needs two (2) memory segments to be written in. For reasons of simplicity of memory management, again, each such IP packet is written into **two** different memory segments --hence, approximately 128-68 = 60 bytes remain unused in every other segment (30 bytes per segment average fragmentation overhead). In this case, the peak incoming packet rate that can be supported is **half** of what it was in question (c).

Translate this packet rate into an equivalent "SONET bit rate", for each SRAM technology considered in (a), using the same IP-over-SONET assumptions used in question (c). Compare the bit rates that you find to those found earlier, and explain the difference.

(e) *--Optional Question--*
Assume again, as in question (c), that the incoming traffic consists of 40-Byte (minimum sized) IP packets. This time, however, the traffic arrives over a number of **Gigabit Ethernet** links (see also exercise 1.3). To calculate the peak packet rate of a Gigabit Ethernet link when carrying minimum sized IP packets, consider that:

- Peak packet rate is achieved over point-to-point links, where no collisions ever occur, and packets can be sent "back-to-back".
- Back-to-back packets over point-to-point Gigabit Ethernet links must be separated from each other by a 12-byte (minimum) "interframe gap".
- Each packet is preceeded by an 8-byte "preamble" (for receiver clock synchronization --no other useful information is carried in that).
- The ethernet header is 14 bytes; in our case, no IP packet information is contained in this header, so it does **not** need to be stored in our buffer memory.
- The ethernet packet body contains the (one, single) IP packet. The ethernet packet body size must be at least 46 bytes (so that the total ethernet packet be at least 64 bytes, for collision detection purposes) and at most 1500 bytes. In our case, the 40-byte IP packet is padded to 46 bytes to satisfy the minimum ethernet packet body requirement.
- After its body, the ethernet packet finishes with a 4-byte CRC; this CRC contains no IP packe information, so it does **not** need to be stored in our buffer memory.

Find the peak packet rate of a Gigabit Ethernet link when carrying minimum sized IP packets. Based on this, calculate how many incoming Gigabit Ethernet links can be supported by the buffer memory of this exercise, for each SRAM technology. The incoming traffic from all links is multiplexed and written into our (single) buffer memory. Essentially, you are asked to divide the peak incoming packet rate of question (c) by the peak packet rate of one Gigabit Ethernet link; give the resulting number, ever if it is not an integer number. Is the aggregate nominal "throughput" of these links (number of links, times "1 Gbps" nominal each) higher or lower than the equivalent "SONET bit rate" in (c) (for each same technology)? Is this good or bad for the Gigabit Ethernet technology?

(f) *--Optional Question--*
Answer question (e) in the case of 68-Byte IP packets, as in question (d). As in (d), two segments per packet are needed, hence two (independent) buffer memory accesses per packet. As in question (e), assume Gigabit Ethernet links; one difference, here, is that no padding is needed in the ethernet packet body, since the 68-Byte IP packet size satisfies the 46 to 1500 byte ethernet packet body requirement.

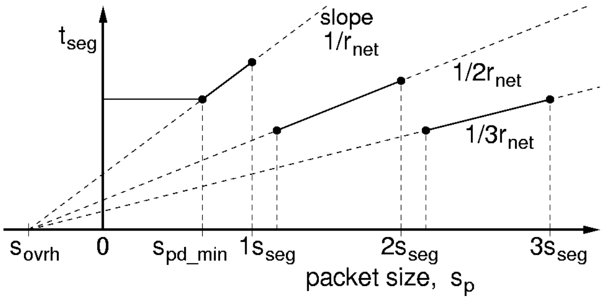
3.2 Segment Size Selection

--- Optional-Answer Exercise: ---
You have to read carefully this exercise, understand it, and think about it for at least 40 minutes. However, you are allowed not to answer it, especially if it looks like answering it will take you much more than the above time.

This exercise is a continuation and generalization of exercise exercise 3.1 above. In this exercise you have to deduce a mathematical formulation for the smallest possible segment size that will not increase the segment access rate beyond the value imposed by minimum packet size. Let us first define the necessary terminology and link technology parameters:

- $\mathbf{r_l}$: gross line rate. This is the link bit rate including overhead. For example, for gigabit ethernet, this is 1.25 Gbits/s, given the 8B/10B encoding on the line (see exercise 1.3).
- $\mathbf{r_{net}}$: net line rate. This is the net (useful) bit rate, excluding the (proportional) overhead of line encoding (the "proportional" overhead is the overhead whose size increases in proportion to the packet size). For example, for gigabit ethernet, r_{net} is 1.0 Gbits/s. For ATM over SONET, $r_{net} = (87/90) \times r_l$ (see exercise 1.2) (this is assuming that all 53 bytes of each ATM cell are stored in memory).
- $\mathbf{s_{min}}$: minimum packet size. This is the size of the smallest packet as stored in memory; For example, for IP, this is taken to be 40 bytes.
- $\mathbf{s_{pd\_min}}$: minimum padded packet size. For ethernet, packets that are smaller than s_{pd_min} are first padded to s_{pd_min} and then transmitted on the line; packets equal to or larger than s_{pd_min} are not paddded. The padding is not written to memory. As we discussed in exercise 3.1, s_{pd_min} is 46 bytes in gigabit ethernet.

- $\mathbf{s_{ovrh}}$: size of the fixed per-packet overhead. This is the fixed-size overhead that is added to each packet when transmitted on the line; its size does not grow in proportion to the packet size, as is the case for the r_l-r_{net} overhead. This overhead is not written to memory. In the case of gigabit ethernet, this is 38 bytes (12-byte interframe gap, plus 8-byte preamble, plus 14-byte ethernet header, plus 4-byte ethernet CRC).
- $\mathbf{s_p}$: packet size, for an arbitrary packet. On the line, this is always s_{pd_min} or greater; in memory, this is always s_{min} or greater.
- $\mathbf{t_l(s_p)}$: time on the line of a packet of size s_p (the time it takes to transmit or receive this packet on the line, measured from a "starting point" of this packet to the same starting point of the next packet, when packets appear back-to-back on the line). For packets of size up to s_{pd_min}, this is: $t_l(s_p) = (s_{pd\_min} + s_{ovrh}) / r_{net}$; for larger packets, the packet line time is: $t_l(s_p) = (s_p + s_{ovrh}) / r_{net}$.
- $\mathbf{s_{seg}}$: the segment size of the buffer memory; each packet is written to memory after being segmented into an integer number of segments of this size. This segment size is the design parameter that we want to determine.
- $\mathbf{n_{seg}(s_p)}$: the number of segments into which a packet of size s_p is segmented. This is the ceiling of s_p/s_{seg} (the smallest integer not below this ratio) (s_p here is the size of the packet as written in memory, hence it can be as small as s_{min}).
- $\mathbf{t_{seg}(s_p)}$: time available for each segment access in the buffer memory, when the packet size is s_p. This is: $t_{seg}(s_p) = t_l(s_p) / n_{seg}(s_p)$.



We want to find the smallest segment size s_{seg} that will maximize the worst-case available segment time $t_{seg}(s_p)$ for all packet sizes s_p. Start by plotting the available segment time, t_{seg}, as a function of packet size, s_p, for a given, fixed segment size, s_{seg}, as in the figure on the right. Observe the properties of the plot: which (discrete) values of packet size are the "most critical" ones? Clearly, this smallest segment size must be at least as large as s_{pd_min}, because all packets of size s_{pd_min} or less take $(s_{pd\_min} + s_{ovrh}) / r_{net}$ line time; if the segment size were less than s_{pd_min}, some packets (e.g. packets of size s_{pd_min}) would require a segment time less than this line time (i.e. a sub-multiple of this minimum packet line time). Hence, the worst-case segment time cannot be higher than: $(s_{pd\_min} + s_{ovrh}) / r_{net}$. Your task is to find the smallest segment size s_{seg} for which the available segment time never falls below the above value, for all packet sizes $s_p > s_{pd\_min}$. Given the above "most critical" discrete values of packet size, conversely, which (discrete) values of segment size s_{seg} are the "most critical" ones? Give some kind of a mathematical formulation or an algorithm for determining the segment size sought. Finally, apply your solution to the cases of (a) IP-over-SONET, as it was (ill- ?) defined in exercise 3.1(c-d); (b) Gigabit Ethernet, as in exercise 3.1(e-f).

Up to the Home Page of CS-534