CSE 121 Winter 2006
Home Work 3

Due: Thursday, Mar 9th, 2006

1. This is an exercise to go through the global replacement algorithm described in the Global Memory System paper (section 3.1). There are three hosts in the system and each has *four* pages in their memory. The following table shows the current local and global pages of these hosts.

Node	Local Pages	Global Pages
A	1, 2, 3, 4	-
B	10, 11	15, 17
C	-	20, 22, 23, 24

   Assume that the age of page is proportional to its number (i.e., age(1) < age(20)) at this point of time. Now, adjust the local and global pages of hosts for the request stream B_{23}, A_{24}, A_8, B_{16}, B_2, A_{11} where A_x means host A is requesting for page x.

2. Suppose a host H is using G global pages; is that equivalent in performance to H having G extra local pages? Why or why not? (Think about the number of pages read/written to the disk.)

3. A pager demon continuously monitors the number of free physical page frames available in main memory. Whenever the number of free frames falls below certain threshold (5 - 10% of the total pages in the system), it goes through the address space and writes the oldest pages to disk (if dirty) and adds the cleaned page frames to the free list. Write pseudocode for a pager function that tries to free X pages from address spaces in L4 μ-kernel using the *grant, map* and *flush* primitives. Please state any assumptions you make about memory subsystem and also make clear distinctions between simple function calls and IPC.

4. Why is careful TLB management on machines with untagged TLBs more critical for μ-kernels than traditional monolithic kernels?s

5. List three differences between μ-kernels (in particular Mach and L4) and virtual machine monitors?

6. Explain whether the following functions are implemented in the L4 μ-kernel or the Exokernel. Why or why not?

- Assigning a physical page
- Reading from I/O devices
- Trap/Interrupt handler
- Handling TLB misses
- Page fault handling

7. Which of the following types of bugs can be detected by Nooks? Why or why not?

   - Trying to access a NULL pointer in the kernel that is passed from a driver
   - Driver trying to write a read-only kernel object
   - Driver trying to use an object (not a kernel object) that is already freed.
   - Infinite loop in the driver
   - Buffer overflow in the driver code