CS414 Fall 2004 Homework 7. Due in class on November 4, 2004

1. Suppose that you have written a program that needs to transfer small files from a nearby server. You do a test and discover that for a 1GB transfer, the network connection runs at 100MB per second. But when you transfer your files (using a separate TCP connection for each file), the file transfer speed is only 4MB per second. Explain where the factor of 25 was lost.

2. In class, we learned that the TCP protocol uses a sliding window algorithm for two reasons. First, this allows it to overcome packet loss. But the algorithm also permits the sender to match its rate to what the network can accommodate, and also to the rate of the receiver. Specifically, TCP operates by increasing its sending rate until packet loss is detected, then reducing the rate. It does this by making the sliding window size larger, or smaller, respectively.

    a) Suppose that you were to use TCP over a wireless network, in which packet loss occurs frequently. How will TCP behave?

    b) Is the behavior you described in (a) the right reaction to packet loss in a wireless network?

    c) Now suppose that you are designing a large corporate network with mostly wired links, but some wireless links to mobile machines "at the edge." What might you consider doing to avoid the problem you described in part (a)?

    d) [Extra credit*] Read about the "End to End Argument". (J. Saltzer, D. Reed and D. Clark: "End-to-end arguments in system design", ACM Trans. Comp. Sys., 2(4):277-88, Nov. 1984). Does your answer to part (c) violate the principle advocated in this paper? Explain.

    * Bumps your cumulative homework score up by 3 points, but not beyond the "maximum" possible.

3. A computer system consists of five machines, named M0 … M4. Each machine has an attached input device, and each reads a single bit from its device (hence, 0 or 1).

    a) Assuming that no failures occur and that messages are delivered promptly and reliably, design a protocol whereby the machines can "vote" for its value (0 or 1). Your protocol should pick a value that at least one machine actually voted for (you can't just say "always pick 0"), Each machine must know the outcome of the election at the end (you can't just say "process 0 now picks the first value it received" – all need to know, and all need the same value). Design your protocol to run in "rounds", such that in each round, each machine sends four messages (one to each of its counterparts), and receives four (one from each of its counterparts). Assume that no failures occur.

b) Modify your protocol from (a) to work correctly even if a single machine fails while the protocol is running (you may assume that the faulty machine crashes, halting at some step in the protocol and that this type of failure is "reported" to each operational machine, in the form of a special message from the network that says "process 2 has failed")

c) Modify your protocol from (a) to work correctly even if some machine might behave maliciously, for example by ignoring some messages, telling one machine that its input was a 0 and another that its input was a 1, etc. It cannot, however, impersonate some other machine – and more generally, you may assume that the faulty machine cannot prevent the correct machines from running rounds, or tamper with messages sent from one correct machine to another correct machine.

d) Can the problem from part (c) be solved with just three machines instead of five? Either explain how to do it, or explain briefly why this isn't possible.

4. Suppose that a program has a virtual address space of size 20MB and is running on a computer with 2MB of memory (so: if you run *k* instances of the program, the VM load on the computer would be 20*k* MB, but the computer would still have 2MB of physical memory; for this problem *k* is equal to 1). You would like to speed up the program, so you begin to add memory. At first, each time you add 1MB of memory you find that the program indeed speeds up. But after the physical memory size reaches 10MB, adding memory stops having any effect.

   a) Explain why extra memory stops helping, even though the virtual memory of the program is twice as large as the physical memory of the computer.

   b) Your close friend Doug "The Bug" Crump drops by and shows you that inside your computer, the "motherboard" has a speed setting. By changing it, he is able to double the speed of the CPU, without affecting the speed of anything else. To your surprise, you now find that adding physical memory now "helps" again, and you can obtain a further speedup. Explain why this happened. (Hint: This is actually not at all easy. If nobody gets it, Ken won't be surprised. Second hint: if you try this, don't be surprised if your computer overheats and melts down!).

5. When modifying the Linux kernel late one night, Doug accidentally modifies the context switching software so that sometimes, when context switching from process P0 to process P1, the system forgets to flush the TLB and main-memory (L2) cache.

   a) What sorts of problems might result from this mistake?

   b) Doug realizes what he has done, but in fixing it, modifies Linux to flush the TLB and cache not just on every context switch operation, but also on every interrupt (including page faults, system calls, device interrupts, etc). What impact would you expect this to have on the system?