CS 677 – Homework 1

There are 10 questions. Each question is worth 10 points.

1. Use one example to show how to organize a web site into a three-tiered architecture. What is each tier expected to do?

2. List a few advantages and disadvantages of user-level threads.

3. A per-CPU runqueue is preferred over a centralized run-queue in multiprocessor CPU-schedulers. Why?

4. Not every node in a peer-to-peer network should become superpeer. What are reasonable requirements that a superpeer should meet?

5. List the contents maintained by the Process Control Block. What are the additional fields that would be required to implement user-level threads? Kernel-level threads? Light-weight processes?

6. Your Friend, Lee T. Hacker, claims that his Shortest Job First CPU scheduler implementation for Linux improves performance drastically. Should you believe him? Why/Why not? What's the issue with SJF?

7. Your friend, Sting Y. User, is renting computer-time on a big, powerful supercomputer running MLFQ scheduler (Multi Level Feedback Queue). Users need to pay for the number of cpu-cycles they use. Your friend's idea is this : "Just before my task's time quantum expires, I issue a frivolous I/O request, which makes my process voluntarily relinquish CPU. This ensures my process is always classified as "interactive" by the scheduler, and because I don't use up my complete time-quantum, I never get charged". Will your friend's idea work? If yes, how will you fix the "problem" with MLFQ?

8. (a) You develop a multithreaded application and it runs just fine on a single-CPU system. What changes, if any, would you need to make to run it on a multi-CPU system? (The OS remains the same in both cases).

   (b) Converting a uniprocessor OS to be multiprocessor-capable is quite hard. Why? What are the changes that might be required for a multiprocessor OS?

9. What are some of the challenges that make Process-migration difficult? What state associated with a process needs to be trasferred?

10. Lee.T.Hacker has another idea: "Instead of emulating a program by translating every instruction every time the program is run, I'm going to emulate it *once* and save the translated program. Next time I run the program, I don't need to emulate the instructions since I can just run the translated program". What can (possibly) go wrong?