

CSCI 5105: Foundations of Modern Operating Systems
Spring 2008
Final Exam
Due: 4pm, 5/13/08

General Instructions:

- The exam is take-home. On-campus students must submit their solution in hard copy to the front desk in the Computer Science Main Office (CS/EE Bldg 4-192) by 4pm on Tue 5/13/08. UNITE students can either submit their solution through the UNITE office (in which case it should be timestamped by the due date/time), or email their solution to the instructor by the due date/time.
- You must write or type your answers clearly (Typing your answers is highly recommended). Attach any loose sheets securely and **put your name and student ID clearly on your answer paper.**
- There are a total of 3 questions, all with subparts. The questions in the exam combine for a maximum of 100 points.
- All your answers must be original and must be done on your own. You are allowed to consult your textbooks, lecture notes and online course material including the additional reading material. It's on your honor not to discuss or copy your solutions from other students or external sources.
- Read each question carefully before answering.
- Good luck with the exam!

1. We discussed a superpeer election algorithm for a DHT network, that elects nodes based on the top k bits of their node IDs.
 - (a) **(8 pts)** Outline at least two problems with such an election algorithm.
 - (b) **(12 pts)** If our goal is to elect the highest capacity nodes (e.g., those meeting certain storage or bandwidth requirements) as superpeers, is there another election algorithm discussed in class that may be more suitable? Justify your choice of the algorithm, and clearly describe what modifications you'll have to make to have it work in a DHT network.

2. Suppose we are using lottery scheduling to achieve proportional-share CPU allocation on a symmetric multiprocessor (SMP) system with 2 CPUs. Assume each process i is assigned t_i tickets in proportion to its weight w_i . Since lottery scheduling is designed for a uniprocessor environment, we make the following modification to the algorithm to make it suitable for a multiprocessor environment: At each scheduling instant¹, we conduct 2 lotteries and pick the winning ticket from each. If these tickets belong to different processes, we assign a CPU to each of the winning processes, otherwise, we run the winning process (of both tickets) on 1 CPU and leave the other idle. For your answers below, you can ignore any scheduling, caching, and load imbalance overheads.
 - (a) **(4 pts)** What is the probability that process i would be scheduled at a scheduling instant?
 - (b) **(4 pts)** What is the probability of leaving 1 CPU idle?
 - (c) **(4 pts)** Suppose you have two processes with weights in the ratio of 1:2, what would be the actual ratio of their expected CPU allocation? Assume that both processes are CPU-intensive and always runnable.
 - (d) **(8 pts)** Based on your answers above, would the suggested extension of lottery scheduling be useful for an SMP system? Why/why not?
 - (e) **(12 pts)** Lottery scheduling was originally developed for CPU scheduling. In general, which other resources besides the CPU—network, memory, and disk—would lottery scheduling be useful for? Explain your answer for each of these resources.

¹Assume that both CPUs are scheduled synchronously.

3. Most laptops have a standby or hibernation feature, which saves a user's current state and data (e.g., open windows, open files, runtime environment, etc.) when the laptop becomes inactive, and resumes from the saved state when the user comes back. Suppose you work for a software company *HiberMobile* that wants you to build a *mobile hibernation* utility, which would provide users with a similar ability to save and restore their state (including the OS, runtime environment and user preferences) and data across machines located in different parts of the network. With this utility, for instance, a user might trigger the hibernation on their home PC but be able to resume from the same state on their office machine, or maybe in a Cyber-cafe in a different country. Having taken CSCI 5105, you are made in-charge of the design and implementation of this utility. You can assume that the home and remote machines have the same hardware architecture, but may have only intermittent connectivity between them (so that they may be disconnected for long time periods), and the identity of the remote machine may not be known at the time of hibernation on the home machine. Further, you can assume that these machines will have your mobile hibernation utility installed along with any supporting software needed.
- (a) **(16 pts)** There are two important problems you need to solve: (i) save and restore of system/user state and data, and (ii) transfer of this state and data to the remote machine. Describe how you will solve each problem, given the assumptions above.
 - (b) **(12 pts)** Briefly describe the sequence of steps (from suspension on home machine to resumption on the remote machine) that would take place during mobile hibernation using your proposed solution.
 - (c) **(8 pts)** What are the limitations of your approach? In particular, are there certain state and data that you cannot restore easily given the assumptions above?
 - (d) **(12 pts)** What optimizations could you do to for a faster resume on the remote machine if:
 - i. The machines were connected through a reliable high-speed connection?
 - ii. The identity of the remote machine was known at the time of suspension?
 - iii. The same user habitually uses both machines, so that the two machines have similar states most of the time (e.g., a user using her home machine at night and office machine in the day)?