CMPSCI 677: Distributed Operating Systems

Final Exam

May 22, 2002

General instructions:

- Please typeset your solutions if possible. If you hand-write your solutions, please be legible.
- This exam is due in 29 hours (at 5pm tomorrow). You can email me a solution in pdf or postscript format (no Word documents please) or drop off a hard-copy in my office. Off-campus students can use email or FAX.
- Do not forget to put down your name and student number on the exam books.
- If your answer depends on any specific assumptions, please state them clearly.
- This is an open book, open notes exam. *You shall not seek help from or discuss the exam with ANY human. If you do, you will be awarded a zero for the exam and will be given a grade of "F" for the course.* Send me email if you need a clarification on any question.
- Explain your answers clearly and be concise. *Do not write long essays.*
- Good luck.

---

1. **Short answer questions** (30 points)
   Answer the following questions in brief.
   (a) (6 pts) Assume that Alice wants to send a message m to Bob. Instead of encrypting m with Bob's public key, she generates a session key K and then sends $\{K(m), PublicKey_{bob}(K)\}$. Explain why this scheme is generally better than sending $PublicKey_{bob}(m)$.
   (b) (6 pts) Give an example where CORBA's dynamic invocation mechanism is useful.
   (c) (6 pts) Explain briefly why message logging allows a distributed application to take infrequent checkpoints and yet recover quickly from a crash.
   (d) (6 pts) True or False: If a process obtains all its locks before it begins execution, it can never deadlock. Explain your answer.
   (e) (6 pts) Is the scenario where your browser downloads a Java applet to execute an example of strong or weak code mobility?
2. **Distributed File Systems** (20 points)
   (a) (4 pts) The textbook states that NFS implements the remote access model to file handling. It can also be argued that NFS supports the upload/download model. Explain why.
   (b) (4 pts) NFS allows the same file to have different names on different clients that mount the directory containing this file. Give one advantage and one disadvantage of this feature.

(c) (6 pts) The Coda file system supports disconnected operations by migrating updated files to the server upon reconnection. A conflict arises if both the server version and the client cached version were updated concurrently. Coda can detect such conflicting updates to the same file but does not resolve such conflicts—it is up to the user to resolve a conflict. What is the rationale behind asking users to resolve conflicts rather than having the file system resolve conflicts automatically?

(d) (6 pts) A peer-to-peer data sharing system such as Gnutella is an example of distributed file system. Would you agree or disagree with this statement? To justify your stance, list some specific features of a (distributed) file system that Gnutella supports and some features it doesn't.

3. **Caching and Replication** (30 points)

(a) (8 pts) The quorum-based replicated writes protocol studied in class replicates a file on N servers. An update to this file requires at least $(N/2+1)$ servers that agree to do this write. A read to the file requires an application to first contact a majority of the servers and have all of them agree upon the version number of the file (after which that version of the file is then read). This two-step process causes the performance of read requests to suffer. Is it possible to improve the performance of read requests by devising a quorum-based protocol that allows an application to contact *only one* server and read the file from that server without contacting any of the other servers? If so, provide such a protocol. If not, provide a counter-example to show that this is impossible.

(b) (8 pts) Rumor spreading or gossiping is a well-known technique to implement epidemic protocols. Explain why rumor spreading can never guarantee eventual consistency (i.e., never guarantee that all servers will eventually receive an update).

(c) (6 pts) Consider a web proxy that employs *leases* to maintain consistency of cached web pages. Is it necessary to tightly synchronize the clocks of the proxy and the web server to implement leases? Why or why not?

(d) (8pts) Consider a web proxy that employs the *pull approach* to maintain consistency of cached web pages. Is it possible to use the pull approach to provide strong consistency guarantees to client requests? If so, explain how this can be done. If not, explain why this is not possible. (Note: strong consistency guarantees that a proxy will never use stale data to answer client requests, and hence, clients will always receive the up-to-date versions of web pages from the proxy).

4. **Multimedia Operating Systems** (20 points)

Consider a simplified form of the Start Time Fair Queuing (SFQ) algorithm studied in class. Assume that each multimedia application (task) is assigned a weight w_i. The scheduler maintains a counter S_i for each task to measure the weighted CPU service received thus far. If a task runs on the CPU for a quantum q, its counter is incremented as $S_i = S_i + \frac{q}{w_i}$. At the beginning of each quantum, SFQ schedules the task with the minimum S_i for execution; ties are broken arbitrarily. The counter of a newly arriving task is initialized to minimum counter of any thread in the system. For simplicity, assume that task are always compute-bound and never do any I/O and that $q = 1$.
Note: The above description of SFQ should provide you all the information you need to answer this question and you shouldn't have to refer to additional SFQ-related material.

(a) (10 pts) At $t = 0$, the system starts with two task with weights $w_1 = 1$ and $w_2 = 10$. Show how SFQ schedules these task (by showing the order in which these task run and the corresponding counter values). At $t = 1000$, a third task with $w_3 = 1$ arrives. Show how SFQ schedules these three tasks.

(b) (10 pts) Consider the same example with a two processor system. Show how SFQ schedules the first two threads and indicate their counter values. Next show how SFQ schedules threads after the third thread arrives. Based on this example, do you think it is possible to use SFQ to schedule threads in a multi-processor system? Why or why not?