OGI SCHOOL OF SCIENCE & ENGINEERING
OREGON HEALTH & SCIENCE UNIVERSITY

CSE515 — Distributed Computing Systems

Final Exam: Monday, 15th March 2004

This is a take-home final. Please type or word-process your answers if possible. This exam paper is available on the Web in PDF and plain text; you are invited to copy appropriate parts of the questions into your answers. Please attempt all questions. Exam answers are due at or before 5pm on Wednesday, 17th March.

Please submit electronic copies of your answers by e-mail. ***Attach*** them in Postscript, PDF or plain text formats (*not* MS Word and *not* RTF), and make sure that the Subject line contains "*CSE515 Final Exam*". Send them to *cse515-hw@cse.ogi.edu*.

You may use the textbook, your notes, and any materials that we have distributed in class to help you write your answers to the exam. You may ***not*** consult with other people! You may use other published source material, but if you do so, *you must specifically reference your source* in your answer.

Each question should take about a half hour to answer — plus the time that it takes to remind yourself of the underlying material. Please write answers that are direct and to the point. If we ask how A and B interact, there is no need to define A and B! Confine yourself to answering the question that is asked.You may ask for clarification of any question by email (send the email to *black* and *walpole*).

I know that all of you are capable of copying pages of true facts out of the textbook. This is a waste of everyone's time. Such behavior will not gain you any marks. In fact, it will *loose* you marks, because we will penalize students who waffle.

Your goal in writing your exam answers should be to convince us that you have understood and assimilated what we have discussed during the quarter, and can apply that knowledge to engineering distributed computer systems. You don't have to write fine literature, but what you write does have to be intelligible.

Question 1. Objects and Distributed Systems

Question 1.1. List ***four*** ways in which an object-based model of computation seems to be a good match for distributed systems

Question 1.2. List ***four*** ways in which an object-based model of computation is probably a bad match for distributed systems.

Question 1.3. Explain how an RPC system or a remote message send system such as Java RMI might deal with

(a) the loss of a *reply* to a remote call or method invocation,

(b) the crash of the server,.

Question 2. Vector Clocks

The symbol $\rightarrow$ denotes Lamport's happened-before relation. a and b are separate events, and V_a is a vector of length n representing the time when event a occurred, as measured by a Vector clock.

Question 2.1. What characteristics of the system determine value of n? Why might this be a problem in a practical system?

Question 2.2. Define $V_a = V_b$ and $V_a > V_b$.

Question 2.3. If $a \rightarrow b$, what do we know about V_a and V_b?

Question 2.4. If neither $a \rightarrow b$ nor $b \rightarrow a$, what do we know about V_a and V_b?

Question 2.5. Explain, in your own words, how Vector clocks can be implemented.

Question 2.6. What are the advantages and disadvantages of Vector clocks compared to Lamport clocks?

Warning: the discussion of Vector clocks in the textbook is at best confusing, at worse wrong! If you are unclear on the concepts, we recommend that you use another reference.

Question 3. Transactions

The two-phase commit protocol is used to implement atomic commitment of transactions in a distributed system. The players in the protocol are a worker process on host *W* and two data repositories on hosts *R* and *S*.

Question 3.1. Describe the messages that are sent between *W, R* and *S* when a transaction *T* executing at *W* needs to read a data item *r* from *R* and to update a data item *s* at *S*.

Question 3.2. Describe the messages that are sent between *W, R* and *S* when *W* attempts to *commit* a transaction.

Question 3.3. What is the latest time (in terms of the messages that you defined in response to part 3.2) at which *R* might decide to *abort* a transaction that *W* wishes to *commit*?

Question 3.4. Why might *R* decide to abort a transaction?

Question 4. Distributed File Systems

Unix Semantics and *Session Semantics* are two different consistency semantics commonly used in distributed file systems:

Question 4.1. Define Unix Semantics by explaining how the data returned by a read relates to the data last written to the same file.

Question 4.2. Define Session Semantics by explaining how the data returned by a read relates to the data last written to the same file.

Question 4.3. Explain the implications of Unix Semantics on cache management and discuss the impact of these semantics on performance.

Question 4.4. Outline a cache management approach for a system with Session Semantics.

Question 4.5. Why does Session Semantics improve performance?

Question 5. Naming

Question 5.1. Outline three key differences between the "super-root" and "mounting" approaches to building a name space.

Question 5.2. Describe the relative advantages and disadvantages of iterative and recursive name resolution in a distributed naming service.

Question 6. Distributed Shared Memory

Question 6.1. What is false sharing?

Question 6.2. Why does false sharing degrade performance?

Question 6.3. Discuss the advantages and disadvantages of the following three approaches for reducing false sharing.

(a) Reducing the page size.
(b) Temporarily pinning pages.
(c) Carefully placing each data structure on a separate page.