Distributed Systems 2010: Exercises

Here are some exercises to aid your revision for Distributed Systems. They are not comprehensive—there are parts of the course that are not covered. However, they provide good practice for the material and tone that is likely to be in the exam. These have all been used as supervision questions for this or the new Concurrent and Distributed Systems courses, so they aren't completely from space.

One

A software module controls a car park of known capacity. Calls to the module's procedures `enter()` and `exit()` are triggered when cars enter and leave via the barriers.

Suppose that the software module is distributed between the embedded systems that run the entry and exit barriers. These communicate via a wireless network that is not always good at forwarding packets; some data may be lost, delayed, or garbled. Give pseudocode for the enter and exit procedures. Be sure to describe how your software modules communicate, what your requirements are (such as in terms of clock synchronisation), and how these requirements can be achieved. What kind of middleware would you use if you had a choice?

Two

Suppose that you are designing an online game. In this game, player-controlled characters move through a world consisting of rooms. A room contains zero or more player characters, zero or more non-player characters (that is, controlled by computer), and zero or more objects. Movement from one room to another is allowed through doors that are defined as part of the game world. Each door specifies the room that it leads to, so physically impossible situations are easy (for example, going from room A to room C may involve going through room B while one can walk from room C to room A directly). Characters can see and hear happenings within the room they're in and hear happenings in adjacent rooms (where "adjacent rooms" means rooms that one can walk to through a door).

The current state of the world can be broken up in various ways. Some applies to a specific character (is that character hungry?) while some, such as a room's contents, are room-specific. Still other state, like the time of day, is global.

1. You are not confident that one machine will be able to handle the load of your fantastically successful new game so you make it distributed. How would you allocate threads of control? Explain your approach, paying particular attention to how data must flow between your threads of control.
2. You decide to use middleware, to make your life easier. What services would be desirable?
3. Would you rather use OOM or event-driven middleware? Explain your answer.
4. Suppose that characters may possess magical abilities that allow them to travel directly from one room to another without the use of doors. How does this affect your design?
5. Sketch the design of an RBAC system for the game. What roles would be required? How should privileges be assigned to things (objects, rooms, characters, whatever) in the game world?

Three

Optimistic concurrency control (OCC) requires that each object have a version number; the object is also the unit of shadow copying.

Care has to be taken in deciding what the "objects" correspond to. In what ways are the tradeoffs involved in this the same as those in settling on locking granularity when using, for example, semaphores? In what ways are the tradeoffs different? (You might want to think about the role of the OCC validator.)

Four

Behind the scenes, Facebook is a large distributed system. In order to handle enormous request volumes while maintaining acceptable response times, data about people, events, "pages", and so on are stored on many machines. These data are assembled to hide this, giving Facebook the appearance of a centralised system.

1. This hiding doesn't always work; sometimes you can see behind the curtain. How do the fundamental characteristics of distributed systems lead to this? As part of your answer, give three examples of how things could "not work." (If you haven't used Facebook, you should find out what it is and make up some reasoned examples of possible misbehaviour.)

2. For each of the three examples that you provide above, which distributed algorithm covered so far in this course could help, if it was implemented completely and correctly?

3. There are lots of reasons why a complete and correct implementation might be undesirable. For one of your three examples, explain why this might be the case. (Hint: think about the resources required by a complete and correct implementation and how much application functionality would really suffer were the algorithm to be substandard.)

Five

1. Complete the sentence, "A vector clock allows each process to know about what _____ have been _____ by every other process in the group."

2. Why does this make it critical that all processes in the group receive all messages?

3. Give an example of an application where causal order is sufficient while total order is not. Explain why this is the case.

Six

Distributed strong consistency requires an algorithm like two-phase commit (2PC). As a part of this, a Commit Manager must be selected—an important decision as the reliability and performance of the CM has a direct impact on the number of times the commit must be retried. Sketch one application where the choice of CM doesn't matter much (it could be selected randomly or by a simple election algorithm) and one where it is critical. Be sure to address the cost, in terms of application performance or correctness, of commit retries.

Seven

Your notes talk about the "bully" and "ring" election algorithms. Suppose that you want to select a validator for optimistic concurrency control. What are the implications of using each of these two election algorithms?

Eight

Recall content delivery networks (CDNs) (maybe from DC1, maybe from your mates). In brief, their job is to serve copies of objects to clients from servers that are close by in the network topology, where close may be in terms of latency, cost, hops, or whatever. CDN nodes tend to use HTTP to deliver the content to clients but some mechanism must load each node with content and ensure that this content is up to date. Suppose that middleware is to used for this. What are the pros and cons of each of remote procedure call, object-oriented middleware, message-oriented middleware, and event-based middleware for implementing a CDN? Don't just list the general advantages and disadvantages of each from your notes; make specific reference to the CDN application.