COMP 4000: Distributed Operating Systems

Winter 2008 Final Exam

April 16, 2008

Instructor: Anil Somayaji

2.5 hours, Open Book

Answer 3 out of the following 5 questions (you may omit 2). If you answer more than 3 questions, clearly indicate which ones should be graded. Please write your answers in a separate exam booklet, or, alternately type them on a computer and email them to `soma@ccsl.carleton.ca` or submit them on a provided USB key.

The exam is open book, open note, open Internet. The only thing you may not do is discuss questions with other individuals. In other words, no emailing/IM/texting/whatever with other folks in the class during the exam!

Note: Please do not simly answer the sub-questions individually and sequentially! Instead, please write a small essay that addresses all of these questions. Be specific where possible but make appropriate generalizations. Be concise—the essays will be long enough if you really answer the questions and time is short.

While you may structure your essays around the given questions, you are not obligated to do so; instead, what is important is for you to construct a coherent argument on the topic that roughly addresses the specific sub-questions. I will be grading each question holistically. A coherent essay that leaves out a few minor points will get a better grade than one that completely answers each sub-question but doesn't connect those answers together to form a larger argument. Above all, show me what you understand, not what you remember.

Good luck!

1. Discuss how well have distributed operating systems been able to replicate the semantics and behavior of traditional, single host operating systems such as UNIX. Address the following questions:

   (a) What are the key abstractions provided by a traditional OS?

   (b) How well do these abstractions translate into a distributed context? What semantics remain the same, and what are changed? Give specific examples.

   (c) What implications do changes in semantics have on legacy code? New development?

   (d) How do these changes affect users?

2. Remote procedure calls are a common abstration of network communication. To what extent are RPC mechanisms a "good" distributed operating system abstraction? Specifically:

   (a) At a high level, how are remote procedure calls built on top of standard network protocols such as TCP/IP?

(b) Why do RPC mechanisms keep being reinvented (reimplemented)? What has remained consistent and what has changed in RPC mechanisms?

(c) How do RPC mechanisms address connectivity and end host failures? Is error recovery a strength or a weakness of RPC mechanisms?

(d) In what ways are good RPC implementations secure? Insecure? Cite specific examples.

3. Discuss your criteria for evaluating distributed operating systems.

   (a) Consider the following criteria for evaluating a distributed operating system or distributed OS mechanism: performance (latency, bandwidth, storage and RAM overhead, etc.), fault tolerance, security, compatibility, usability (for programmers and users), scalability, "elegance". How would you prioritize these when judging a system? Any other criteria you'd add?

   (b) What was one of you favorate papers this term, in terms of technical content (not writing)? What was one of your least favorite ones? Evaluate each according to your criteria.

   (c) What qualities would your ideal, **technically feasible**, distributed system have? Specifically, what trade-offs would you make to create your ideal system? Please specify your target usage scenario.

4. A recent article in the Communications of the ACM argues that "disk is the new RAM". Their basic argument is that by accessing lots of disks in parallel you can simulate having a very, very large amount of RAM. Evaluate this claim.

   (a) In what ways could a large number of disks (1000+) running in parallel act like a very large bank of RAM? How would it be different?

   (b) Describe an existing or hypothetical system for making parallel disks simulate RAM. How would it work?

   (c) What workloads would be best suited for such a system? For these workloads, how useful is the "simulated RAM" abstraction? Or, would it be just as easy to use a traditional filesystem or database abstraction of the disks?

5. How do past distributed operating systems and distributed OS mechanisms managed trust? Specifically, what access to they give to administrators? Regular users? Outside access (anonymous access)? Discuss at least three systems and classify how they approach trust. (You may use your own classification system.)