Take-Home Final Examination

EECS 443 Advanced Operating Systems

Winter 2009

Due: Friday, March 13 2009 at 11:59PM. Late submissions will not be graded.

Instructions: The exam has 4 parts testing different skills/knowledge you have (hopefully) acquired during the quarter. There are two homework-style questions, one paper summary and one research/design question. Since this is an exam, *you should not discuss these problems with anyone else.* You can refer to your textbook, class notes, handouts, and papers reviewed during the quarter. You should not consult any other material while working on the exam.

Declaration: I hereby declare that I have not provided and/or received any help during this exam.

Name:

Signature:

Please drop the signed page under my door by the exam's due date and time.

Problems

1. *30'/15%* Processes in traditional operating systems are consider a unit of independent activity. As such, processes provide a dual function, acting as protection domains and resource principals. What is wrong with this picture, what paper you have read identify the problem, and what solution do the authors propose?

2. *30'/15%* Briefly explain the difference between between synchronous and external synchronous I/O (as proposed in [3]); use an example to illustrate it.

3. *2hr/30%* Prepare a one-page summary of the article by Haeberlen et al. [1] appearing in the 21st ACM SOSP. The summary should include: (i) a brief summary, (ii) a paragraph of the most important ideas, (iii) a paragraph describing their biggest flaws, and (iv) a last paragraph where you state potential future research, relevance, etc. Please use short, to the point sentences when writing your summary.

4. *?hr/40%* There is a growing interest in the networked systems community on increasing application availability. Software failures severely reduce system availability – a recent study showed that software defects account for up to 40% of systems failures [2]. Qin et al. [4] argue that many bugs are correlated with an application's execution environment and propose looking at bugs as "allergens". The basic idea is to, after detecting a bug, roll back an application to a recent checkpoint, dynamically change its execution environment (hopefully removing the offending allergen) based on the failure symptoms, and re-execute the buggy code region in the new environment. If the re-execution successfully pass through the problematic section, the environmental changes are disable to avoid potential side-effects.

   However, as you know, most bugs can be traced to programming errors. Unlike the environment, changing a third party application is a "bit" more difficult. How far can we push the high-level analogy? Could you vaccinate an application – adding some basic code that allows it to keep on going? Is there something like a placebo effect we could explore here (where a placebo could be a `no-op`?

   You are to propose a potential research project that explore related ideas, including a short-term exploration phase that could allow you to decide if there is something to it. When writing your one-page proposal you may want to follow the guidelines in J. Wilkes' "CSP Project Startup Documents" (available in the course schedule).

References

[1] A. Haeberlen, P. Kouznetsov, and P. Druschel. PeerReview: practical accountability for distributed systems. In *Proc. of the ACM SOSP*, Stevenson, WA, October 2007. http://www.sosp2007.org/program.html.

[2] E. Marcus and H. Stern. *Blueprint for high availability*. John Willey & Sons, 2000.

[3] E. Nightingale, K. Veeraghavan, P. Chen, and J. Flinn. Rethink the Sync. In *Proc. of the USENIX OSDI*, Seattle, WA, November 2006.

[4] F. Qin, J. Tucek, J. Sundaresan, and Y. Zhou. Rx: treating bugs as allergies – a safe method to survive software failures. In *Proc. of the ACM SOSP*, Brighton, UK, October.