UNIVERSITY OF WISCONSIN-MADISON
Computer Sciences Department

Operating Systems Depth Exam **Spring 2005**

Instructions: There are **ONLY SIX** questions on this exam; answer **ALL SIX** of them. This exam has three (3) pages.

Question 1. Failure and Recovery Semantics:

Consider two server designs: (1) the NFS file service and (2) the Birrell and Nelson remote procedure call service. An NFS server needs to retain no state about it clients, while a Birell and Nelson RPC server keep a unique ID to track each instance of a client connection.

A. For each of these two designs, describe how the client would detect a server failure (assume that the server host crashed and will shortly reboot). Describe any differences in the way that the client would respond to such failures. Do these failures need to be made visible to the user code in the client or can they be contained and handled within the client run-time code?
B. NFS retains no state about clients while the Birrell and Nelson RPC does. Is this choice merely a design preference or is it dictated by the semantics of the services being provided? Explain?

Question 2. Byzantine Generals Meet Reality:

One solution to the Byzantine Generals problem makes the following assumptions about the underlying system:

A. Every message that is sent is delivered correctly.
B. The receiver of a message can reliably determine the identity of the process from which it was (most recently) sent.
C. The receiver of a signed message can reliably determine whether the message has been modified since it was signed.
D. Anyone can verify the authenticity of a signature.

Discuss whether each of these assumptions is reasonable in light of the way distributed systems really work. If an assumption is reasonable, describe how a system would support such a feature. Where applicable, provide examples from systems that you know. If an assumption is unreasonable, explain why the feature is difficult to provide.

Question 3. File Consistency:

AFS provides a well-defined cache consistency model. In this question, we explore AFS and discuss ramifications of its design and implementation.

A. Central to AFS consistency is the notion of a *callback*. Describe what a callback is and why it is important to AFS, how and when one is established, and at least one circumstances when a callback might be "broken".
B. The AFS consistency model is often described as "last writer wins." Describe what this term means and why it is appropriate to use in describing AFS. In particular, give an example scenario where this issue arises.
C. The interface between AFS clients and servers in original AFS is a "whole file" interface (in contrast to block-based NFS). Discuss why the AFS designers chose this whole-file approach, and then describe its strengths and weaknesses (examples are fine).

Question 4. Scheduling with Working Sets:

Your task is to develop a scheduler that is optimized to handle workloads containing many memory-intensive, fairly long-running non-interactive jobs.

A. What information does the scheduler need from the virtual memory system to make the best decisions? Describe how the virtual memory can obtain, track, and record the needed information.
B. What scheduling policy would you develop to optimize both throughput and fairness for this workload? Describe your algorithm in either sentences or pseudo-code.

Question 5. Web Server Structure:

You are to develop a high-performance web server that is able to handle multiple HTTP requests concurrently. You can assume you are executing on a uniprocessor.

A. How would you structure your web server to obtain high throughput? You can use user-level threads, kernel-level threads, processes and/or events and you should consider both static and dynamic content. Describe the role of each thread, process, or event handler.
B. Imagine that you have an additional goal of providing the shortest average response time for requests. Describe how you would modify your web server to schedule the shortest requests first. What factors throughout the system determine the service time of a given request?

Question 6: Hardware versus Software RAID

RAIDs are often used in data storage. We now explore some aspects of RAID design.

A. Many RAID systems are implemented in "hardware", i.e., as a box that connects to your system via SCSI bus. Within the box, there are many specialized hardware components that the RAID architect can take advantage of to improve performance. One common component is NVRAM - non-volatile memory that is battery-backed and hence can be used as a reliable repository for data. How can NVRAM be used to improve the performance and reliability of a RAID system?
B. A "software" RAID system is an in-kernel pseudo-device driver that emulates RAID functionality on top of disks that are attached to the machine. To the file system above, it appears as a single, large disk, but internally, the software RAID driver spreads disk requests across the devices. What are some of the major differences between software and hardware RAID? (what advantages and disadvantages does each type of RAID have?)
C. Finally, consider a **distributed** RAID system, consisting of numerous server machines with disks, with a client striping data in RAID-5 style across the machines (disks) of the system. When the client needs to update a single block in a stripe, 4 distinct I/Os occur. Assume a failure occurs in the middle of the RAID-5 update; more specifically, assume that a server crashes after one disk has been updated but before the other has, leading to an **on-disk inconsistency**. What can one do in order to detect and recover from such an inconsistency?