**15-440 Homework #2**
**Kesden/Spring 2012**

**Design: Communication, Migration, Checkpointing, and Directory Services**

1. In class, we discussed the design and implementation of Java's RMI. Unfortunately, this system is brittle in the sense that hardware failure on one host can make object unavailable.

   Please consider a system that substantially maintains Java's RMI interface, but allows for the check pointing of objects, and their migration in the event of failure.

   (a) Please describe the most important challenges in designing such a system

   (b) Please sketch out a solution and describe its overall design. You may assume that we understand how the existing RMI system is designed and implemented.

   (c) Please describe changes to the RMI API, or the behavior of the RMI system as visible to the application program. Focus on only those changes that are most critical to your design or most impactful for the application programmer.

**Distributed File Systems**

2. In class we observed that AFS and NFS manage consistency differently. AFS issues callbacks upon updates. NFS validates the client cache periodically.

   (a) [2 points] Do either of these mechanisms eliminate the window of vulnerability? If so, how? If not, is possible to eliminate the window of vulnerability? Why or why not?

   (b) Which mechanism will result in less network traffic in the event that many dozens of clients have the same file open for high-frequency random-access reads?

3. Consider Coda's whole-file semantics, including the behavior of open() and close as well as of caching.

   (a) How does it enable disconnect and weakly connected operation, e.g. use of the file system even when network connectivity is poor or non-existent?

   (b) [2 points] In what ways does it limit file system performance and capability?

**Design: Special Purpose Distributed File Systems**

4. Consider the design of a distributed file system for light-weight mobile devices, especially smart phones, such as common iPhone, BlackBerry, Android, and Windows Mobile devices.

    (a) The file system should be robust without involving any off-line backups, e.g. tape.
    (b) It should view the device's storage as a cache for the actual data, but not necessarily the primary copy.
    (c) It should assume a workload similar to what we see with these devices now, e.g. notes, calendars, photos
    (d) It should support user data, not necessarily user programs
    (e) It should facilitate the migration from one device to another and the use of the data on at least one host computer
    (f) User data should stay private to that user, but should be very quick to access, especially from the device, itself.

    Assume the following properties of the systems:

    (a) Files are generally "small", e.g. text messages, notes, short documents, and cellphone photos, but not long hi-res videos, databases, etc.
    (b) Latency is very high, e.g. 500mS
    (c) Bandwidth is modest, but not terrible for downloads, e.g. 1Mbps
    (d) Bandwidth for uploads can be outright bad, e.g. 0.20 Mbps
    (e) Although the data can be changed from multiple hosts, it will not be accessed concurrently, since there is only one user.
    (f) The storage on the each of the mobile device and host are large relative to the user's mobile needs
    (g) Files access generally requires the whole file, rather than random access to only some part of it.
    (h) Off-device storage is "Free"
    (i) The wired Internet is "Fast and wide"

(a) What are the most important challenges presented by these requirements?

(b) Please describe the architecture of your solution, include especially descriptions of caching, replication, checkpointing, and the protection of privacy.

**Data-Intensive Scalable Computing (DISC)**

5. How would it impact Hadoop's performance if it were to be implemented over AFS instead of HDFS? Why?

6. The output of a Mapper is written into the local filesystem instead of the global filesystem. Why?

   Your answer should explain both why writing into the global file system would be undesirable as well as why it would be of minimal benefit.

7. Why does Hadoop sort records en route to a Reducer? How would it affect things if these records were processed by the Reducer in the order in which they were received from the various Mappers?

8. What happens if a Mapper or Reducer fails?