Name: AndrewID:

15-440 Homework #1
Fall 2014 (Kesden)
Due: Tuesday, 10/7/2014

1. Networks

(a) Given a 100Mbps network with an optimum sliding-window size of 1024 bytes, what would be the optimal window size if the bitrate was increased to 1Gbps and the maximum allowable run of the cable was cut in half? Why?

(b) When transmitting at the link layer, we normally "frame" data. Why? For example, why do we not just stream bits?

(c) Consider global scale communication over the next 30 years. In which dimension, latency or bandwidth, will we mostly likely see the greatest improvement? Why?

2. Middleware/RPC/RMI

(a) Consider the implementation of an RPC system in a homogenous environment (same hardware, same OS, same language, same, same, same). Is it possible to implement a pass-by-reference (not necessarily pass-by-address) mechanism? If not, why not? If so, in what ways might it be best to relax the semantics of a typical local pass-by-reference situation? Why?

(b) Consider the implementation of an RPC system in a heterogeneous environment (different processor architecture, different OS, different programming language, different, different, different). How might the heterogeneity complicate the model? Please consider each of the following:

i) Simple primitives (think back to 213 for how they can differ from system to system)
ii) Complex data types, including structs and strings,
iii) Higher-order language and library data structures, such as linked lists, maps, etc.
iv) Programming paradigms (function pointers, jump table, functors, etc)

(c) Consider Java's RMI facility, which generates stubs at compile time. Could it, instead, generate the stubs at runtime? For example, could it disassemble a class file, or inspect an object's properties at runtime, rather than at compile time? If not, why not. If so, what would be the advantages and disadvantages of this model?

(d) Consider Java's RMI facility, which only plays nicely with classes that implement the *Serializable* or *Remote* interfaces. Would it be possible to implement an RMI facility in Java that worked for all classes? For example, by using a combination of the class file, as well as reflection and other Java mechanisms to decompose, serialize, and reconstitute instances by brute force? If so, please explain any necessary limitations. If not, please example why not.

3. Distributed Concurrency Control

In class we discussed enforcing mutual exclusion, among other ways, via a central server, majority voting, and token ring.

(a) Which of these systems requires the fewest messages under heavy contention? How many messages are required per request?

(b) Which of these systems requires the most messages under heavy contention? Why?

(c) Which of these systems is most robust to failure? Why?

(d) The *voting protocol*, and the *voting district* protocol, are based upon participants reaching an agreement as to who can enter the critical section. What happens in the event of a tie that could otherwise risk deadlock?

(e) Many coordinator election protocols have analogous mutual exclusion protocols and vice-versa. What is the most important similarity of the two problems? What is the most important difference? Focus your answer on the nature of the problem, itself, not the solution.

(f) In the event of a partitioning, techniques such as token ring can result in two or more distinct groups, each with its own coordinator. How can this be prevented, while enabling progress?

4. Logical time

(a) One form of logical time is per-host sequence numbers, e.g. “5.1” is time 5 on host 1. Another form is Lamport’s logical time. What advantage does Lamport time offer? At what cost?

(b) In class, we discussed a simple form of vector time. What relationship(s) among timestamps can be discerned from this form of vector logical time, but not Lamport logical time?

(c) Consider the amazing progress we’ve made in data communication networks over the years. Will it –ever-- be possible to sync physical clocks rapidly enough to use as the basis for correct synchronization of a global distributed system? Why or why not?

5. Databases and Friends

(a) What is the *CAP conjecture [Theorum]?* Please expand the acronym and explain it, and also explain the intuition behind it.

(b) What does *BASE* stand for*? Please also explain each property*.

(c) What does *ACID* stand for? Please also explain each property*?*

(d) Please consider *BASE* and *ACID*. Please provide general guidance of the circumstances under which each is a uniquely appropriate model. Please also provide and explain one concrete example for each case.

(e) Assume a participant in a 2PC aborts after acknowledging a pre-commit, how does the system ensure correctness?

(f) Consider a distributed transaction implemented via *2PC*, at what point does a resource used by a transaction become unavailable? What about become available again? Why is this locking necessary?

(g) How does *2PL* ensure that deadlock is not possible?

6. Replication and Quorums

(a) Consider replication to multiple servers based upon a write-one, read-all static quorum with version numbering. What steps must be taken upon a write to ensure the correctness of the version number?

(b) Consider Coda Version Vectors (CVVs). Give an example of two concurrent CVVs. What is indicated by concurrent CVVs? Please describe one situation in which this might occur.

(c) Consider a situation with 5 replica servers employing a write-2/read-4 policy and version numbering. Please design and describe a locking scheme that ensures that version numbers remain correct, even in light of concurrent writes. You do not need to consider failure. But, you do need to describe all necessary communication and state, such as communication between or among servers and/or participants.

(d) Please consider the special case of a mobile client using a read-one/write-all quorum. Please design an efficient locking protocol to ensure concurrency control. But, please ensure that your protocol permits the lock to be acquired at one replica and released by another.

(e) Coda Version Vectors (CVVs) are a form of logical time stamp, specifically a vector time stamp, which are used to aid in the management of replication. What could cause two CVVs be *concurrent* (and non-identical)? Why? Illustrate your answer with an example involving 2 servers and 2 clients.