

Name:

email:

ID:

Midterm Exam
CS555
12 Mar 2009

You have 1 hr. 20 min. for this exam. The exam has 6 pages. There are 100 possible points. Show all your work for partial credit.

Definitions

Each question is worth 1 point. Answer all questions in this section.

1. Define the following

a) Page rank

Answer: Google's metric for ranking web pages based on the interconnections between web pages.

b) Symmetric-key Cipher

Answer: Encryption algorithm where both communicators hold the same secret/key.

c) Replication

Answer: keeping multiple copies of a piece of data in the system to improve availability or performance (or both)

d) Nonce

Answer: A piece of random data included in a message to insure the message is unique.

e) Mutual Exclusion

Answer: A guarantee that only one process/task is executing a critical code section at once.

f) Stable Property

Answer: Predicate of a distributed system that remains true once it has become true. (Strictly it is one that is true in any state reachable from a state in which it is true.)

g) Copy Set

Answer: The set of nodes in IVY that are sharing a page.

h) Relation Graph (from Connections)

Answer: The representation of temporal context links between documents in connections.

i) Non-determinism

Answer: Feature of CSP that allows for the system to make a random choice between alternatives in a guarded command.

j) Intentional Anycast

Answer: Ins primitive that delivers a message to one of the destinations having the given attributes in its name.

Short Answer

Each question gives its value in the question. Answer all questions in this section. This section is worth 50 points.

2. After a change has been made to a DNS primary name server, how long until all nodes using the DNS have a consistent view of that change? (5 points) You may assume that all nodes can talk to each other throughout the update process, and only the content of the zone has changed (no servers have been added to or removed from the system). Also, none of the caching or refresh times are updated. Can you calculate a similar value for Grapevine? (1 point) Explain why not or explain how to calculate it. (4 points).

Answer:

There are two values to consider, the caching time for the name and the update time for the secondary server. In the worst case, the change is made immediately after a secondary server refreshes itself, it continues handing out old names for one refresh interval. If an end node gets the old name immediately before the refresh interval expires, it will get the old name and can cache it for the caching time. The old name can thus last for one refresh interval + one caching time. Both of these values are set at the primary server.

One cannot calculate such a value for Grapevine because update messages may be arbitrarily delayed by the Grapevine message system. The paper mentions a forced reconciliation at the end of the day, and mentioning that is worth some points.

3. Consider a system implementing the signed messages solution to the Byzantine Generals problem. Currently General 0 is acting as the commander, and a message signed by general 1 and then general 2 has the format *message:1:2*. A general receives the following two messages:

attack:0:1:4:3
retreat:0:2:6:4

What does the general that received these messages general know? (1 point) Why? (3 points) Does anyone else know the same fact? (1 point) Why? (3 points) How many traitors does the signed message variant protect against? (2 points)

Answer:

General 0 is a traitor. Because message signatures cannot be forged, these messages indicate that general 0 has given contradictory commands to generals 1 and 2. They are both directly signed by general 0 and have different information.

If general 4 is loyal, it knows that general 0 is a traitor as well. That general's signature appears in both signature chains, so it has seen the contradictory output as well. (Of course if general 4 is a traitor, this may not matter).

Signed messages protect against an arbitrary number of traitors. If there are not at least 2 loyal generals, the solution isn't meaningful, so $n + 2$ is an acceptable answer as well.

Name:

email:

ID:

4. We have discussed several systems that use optimistic consistency to reconcile uncoordinated changes to replicated files. Name one such system (2 points). Explain what an optimistic system does when it discovers that uncoordinated changes to different replicas conflict. (2 points) Uncoordinated changes to some kinds of files are easier to deal with than others. Give an example of a kind of file that optimistic consistency works well on and one that it works poorly on (2 points). Explain what properties of a file or its usage determines how well optimistic systems can successfully deal with uncoordinated changes to it. (4 points)

Answer:

Coda, LOCUS, and FICUS all use optimistic consistency. When an optimistic system discovers uncoordinated changes, it attempts to create a single unified copy of the file automatically, and reports the conflict if the system cannot resolve it. Files that have well defined update semantics that are known to the consistency system are most likely to be successfully reconciled. For example mail files, directories, databases all have well defined operations that LOCUS and FICUS can use to resolve conflicts automatically. Source code does not.

5. Recall that when a client (A) is authenticating to a server (B) using Kerberos, the first message from A to B includes an authenticator that includes the current time and a ticket that contains an expiration time. Consider a server that receives a message in which the ticket's expiration time is valid, but the time at the server disagrees with the current time in the authenticator by a large amount. What should the server do in response to the message? (2 points) Why? (3 points)

Answer: The server should discard the message, as it is likely a replay. It is also possible that A's clock has gotten reset, but even in that case the server must discard the packet as it cannot tell the difference.

Name:

email:

ID:

6. On most systems for transmitting voice (telephones, voice over IP) between humans, error detection and correction is minimal. Explain why. (2 points) Imagine that you are designing a system to transmit voice commands over the network to a computer controlling a power plant. Does placing a computer on one end of the voice system require error correction in the network (explain your answer)? (3 points)

Answer: In the first case, humans are the endpoints of the communication and can (and do) initiate their own error resolution protocol ("say that again"). The second case is more interesting, in that a computer has more limited ability to recognize voices. Changing one endpoint to a computer doesn't really change the problem. That endpoint still needs to be able to deal with garbled commands - for example the human says something inconsistent - so the end-to-end argument says that adding the error correction is not necessary, though it may be helpful for performance.

7. Briefly explain what soft state is. (3 points) Name 2 systems that use it. (2 points)

Answer:

Soft state is data with a lifetime that must be refreshed regularly by its owner, or it times out and disappears. SNS, JINI (leases), and the CAN all use soft state.

8. A hint is approximate data that a system must validate before using, or that can act as a starting point for a computation. Characterize each of these concepts as a hint or not.
- Notify in Mesa
 - A DNS name
 - The ProbOwner field in IVY
 - A read quorum
 - A FICUS graft point

Answer: The notify and ProbOwner field are hints.

Long answer

This section is worth 40 points. Each question gives its value. Do all questions in this section.

8. Several users have downloaded the source to a software package and each attempts to compile it on a traditional UNIX system. Each discovers that the software expects a set of header files to be installed in `/usr/include` that are installed in `/usr/local/include` on the users' machines. This compiler uses a search path to resolve relative names in the source, and the developer of the software has coded the header files as relative names.

- a) The compiler supports additions to the search path, so one user adds `/usr/local/include` to the compiler's search path. Explain how the search path is used to search the name space (this is analogous to the DNS search path). (2 points) Explain some problems this user might encounter if there are files with the same names in different directories in the search path (especially if some of those files are in the new component of the search path). (3 points)

Answer:

Each component of the search path is prepended, in turn, to the relative path names to make absolute paths. The first absolute path that resolves to a file is used.

The user may not have complete control over the compiler's search path through the name space. It may require work to find a search path that resolves all the relative names the way the software developer intended.

- b) Another user resolves all the relative header file names to absolute path names, by editing the source files. Where ambiguity existed, the user chose the correct file by trial and error. Explain that action in terms of name space operations. (2 points) Explain the potential drawbacks of this solution. (3 points)

Answer: The user has applied a complete closure of the names relative to this system. This code is even less portable than the original relative names, because only the layout on this machine is guaranteed to compile correctly.

- c) A third user discovers that `/usr/include` is always searched first by the compiler, and resolves ambiguities by linking files not in `/usr/include` into that directory. (This user has sufficient rights on their machine to make such links). Explain this action in terms of name space operations. (2 points) Explain the potential drawbacks of this solution. (3 points)

Answer:

This user has added aliases to the global name space to resolve ambiguities for this program. Altering the name space in this way alters it for everyone, so others compiling on this machine (or this user compiling other programs) may find files in unexpected places.

- d) Explain how would a Plan 9 user would resolve this issue. (5 points)

Answer: A Plan 9 user would create a name space around the compiling process that placed the header files in a single directory using Plan 9's user-centric naming system. This would essentially be a combination of the b & c solutions, but one that applied only to the compiling process.

Name:

email:

ID:

9. The ISIS system and Time Warp both use underlying ideas from Lamport's discussion of causality to create functional systems. This question explores how those systems implement those ideas.

a) There are three rules that define Lamport's "happened before" relation. Give them. (6 points)

Answer:

A "happened before" B if they are events in the same process and A did happen before B (processes imply an internal order). A "happened before" B if A is a message send event and B is a message receive event. If A "happened before" B and B "happened before" C, A "happened before" C.

b) Time Warp messages are processed in causal order, where the virtual time of the messages is taken as a logical clock. Explain how this order is guaranteed in Time Warp. (5 points)

Answer:

Messages are processed as they are received in real time. If a message is found to be misordered, the processes that are causally affected by this are rolled back to the virtual time at which the message arrived and restarted.

c) ISIS's CBCAST messages are also processed in causal order, but a different mechanism is used to enforce that order. Explain how messages are ordered in ISIS. (5 points)

Answer: Messages are delayed by ISIS until all processes in a group agree on the next message to deliver. These messages are ordered by logical timestamps. That is explicit message blocking is used.

d) For one of those systems (Time Warp or ISIS) explain a shortcoming of their causality guaranteeing system. (4 points)

Answer:

Using rollback in Time Warp makes it difficult to incorporate user input, because rolling back events outside the virtual time is difficult. In ISIS, the overhead of ensuring pure causality is wasted on applications that can operate with looser guarantees.