

CSE 598Z
Midterm Exam
7 October 2004
Prof. Douglas Thain

Short answer:

1 - Sun RPC requires all operations to be idempotent and persistent while Birrell RPC does not. This means that Sun RPC can hide network failures while Birrell RPC exposes them to the user. However, Sun RPCs are slower than Birrell RPCs because they must write synchronously to disk.

2 - The AFS production system uses callbacks for consistency management, whereas the prototype checked a file's metadata on every operation. This change was made because the prototype was very metadata intensive.

3 - The higher levels are more highly replicated because they are traversed by all, but change infrequently. The opposite holds for lower levels of the filesystem tree.

4 - Direct flocking: An agent talks to multiple pools to run jobs.
Gateway flocking: Two trusting pools exchange jobs between themselves.
Gateway flocking was too complicated to implement and operate.

5 - Only Sprite provides process migration. Sprite assumes each workstation has a primary owner, while Amoeba assumes all CPUs are common property.

6 - Finger successors point to the first few logical nodes past each of the logical fingers used for searching. The finger successors are an optimization that allow a node to find nodes that are both logically closer to the target node and physically close to the current node.

7 - An adversary can damage a loyal peer only when the loyal peer calls a vote. So, the rate is limited by the holder of the data, not the attacker.

8 - If a process rolls back from time t to a checkpoint at time c , then all messages sent between c and t are candidates for antimessages. As the process re-runs the interval from c to t , TWOS checks to make sure any messages sent match those previously sent. If they do not, an antimessage is sent.

1 – Linda

S/Net protocol:	Leichter Protocol:
<p>A waits B waits C broadcasts (25) A matches (25) and broadcasts delete (25) B also matches and broadcasts delete (25) C sends B:proceed and A:wait C broadcasts (25) A matches (25) and broadcasts delete (25) C sends A:proceed B broadcasts ("F",f) C broadcasts delete ("F",f) B sends C:proceed A broadcasts ("G",g) C broadcasts delete ("G",g) A sends C:proceed</p>	<p>A broadcasts template (retries) B broadcasts template (retries) C emits (25) to local storage C matches (25) to A's template C sends (25) to A A consumes (25) C emits (25) to local storage C matches (25) to B's template C sends (25) to B B consumes (25) C broadcasts template ("F",int f) B emits ("F",f) B matches template B sends ("F",f) to C C consumes ("F",f) C broadcasts template ("G",int g) A emits ("G",g) A matches template A sends ("G",g) to C C consumes ("G",g)</p>

2 – File System Semantics

Imagine two processes that are both reading and writing a file:

Time	Process A:	Process B:
0	open	
1	read	
2		open
3	write	
4		read
5	close	
6		write
7		close

Unix:

The changes made by process A will be immediately visible by process B at time 4, because data in the Unix buffer cache is equally visible to all processes. At the end, the changes made by both processes will take effect.

NFS:

The changes made by process A will not necessarily be visible to process B at time 4: it depends on when the periodic flush of A's buffer cache occurs. However, if A and B write different parts of the file, both of their changes will take effect.

AFS:

When process B opens the file at time 2, it will receive a complete copy of the file at that time. So, B will definitely not see the changes made by A at all. In fact, because B closes the file last, B's changes will completely overwrite the file, and A's changes will be lost.

3 - Amoeba separates data access into two types of servers: the bullet server and the directory server. The bullet server provides access to inode-like data structures, while the directory server simply maps names to bullets. Because an application need not have ANY name for a bullet, it is impossible to tell which are in use. So, Amoeba automatically deletes bullets that have not been touched for some time. It is the client's responsibility to periodically touch the bullets that it uses.

xFS separates data access into two types of servers: the storage server and the manager server. The storage server provides access to block-level data, while the manager server is responsible for loading and interpreting inodes. Because only xFS clients can access storage servers, clients are responsible for recording what blocks they use in s-files. A cleaner is responsible for reading the s-files and deleting what is not in use.

These systems have different approaches because one is an open system and the other is a closed system. Amoeba is an open system. Because arbitrary components work together, no single component knows what data is in use. In contrast, xFS is closed. Although it is inconvenient to access, all of the information for garbage collection is in a known set of clients.

4 - Each node continuously tracks a local minimum virtual time (LMVT) which is the minimum of the local VT and the VT of any messages in the input queue. A central node periodically queries all nodes in the system to collect their LMVTs. The minimum of all LMVTs so collected is a lower bound on the global virtual time, so we call this the estimated global virtual time (EGVT.) The central node then transmits the EGVT to all nodes in the system, which may then free any messages in the output queue with $VT < EGVT$. The more frequently that the central node runs its algorithm, the less storage is used, but the more messages that are generated. If the only worry is running out of storage completely, you could minimize messages by only garbage collecting at the suggestion of a full node.

(The answer above is fairly complete, and would be worth 3/4. However, it won't quite work. The central node does not capture the system in a consistent state, so it is possible that the EGVT would be overestimated. A working solution must find a way to capture the system in a consistent state. A sufficient way to do this is to have the garbage collector pause every process before measuring the EGVT. A more clever way to do this is to run the Chandy-Lamport algorithm, but that will come later in the semester.)