Outline for February 15, 2001

1. Greetings and felicitations!
   a. Tuesday, Feb 20 3-4:30: Friday Feb 23 1:10-2:30; go to 1101 Hart Hall to view
2. Application: clock synchronization in the face of faults
   a. interactive convergence algorithm
   b. interactive consistency algorithm
3. Example of time protocol: NTP
4. Distributed File Systems
   a. Goals: network transparency, high availability
   b. Architecture: flat (all servers); have specific file servers and clients
   c. Critical services: name server (resolves file names), cache manager (speeds accessses up)
5. Building Blocks
   a. Mounting: binds differing name spaces to a single global view of the name space
6. Mounting
   a. Put the name space at a point in the file tree, as a directory (mount point)
   b. Mount table maps mount points to remote file servers
   c. Maintain mount info at client: each client mounts file systems individually; updates in client (NFS)
   d. Maintain mount info at server: every client sees identical name space; updates at server only (Sprite)
7. Caching
   a. Reduces delays in file accesses
   b. Copy data to local client ; exploits temporal locality
8. Hints
   a. Like caching, but regard data as not reliable
   b. Example: store name resolution data, and work from closest back; invalidate cache entries on failure
9. Encryption
   a. Use Needham-Schroeder (describe generic protocol)
10. Naming
    a. Name resolution: map name to address or object (or multiple oblects)
    b. Name space: collection of names; need not share name resolution
    c. Approach 1: names identify host: /net/olympus/home/bishop; unique, location dependent (NFS)
    d. Approach 2: mount remote directories locally: need to know host system for mount, but then it's location transparent (NFS)
    e. Approach 3: single global directory resolving all name references (Sprite, Apollo); unique, location transparent, but typically limited to single organizational units
    f. Contexts
       i. Partition of name space; context identifies the name resoultion mechanism to use
       ii. Example: Tilde partitions name space into directory trees based upon project; context is the set of tilde trees that a process has in its environment
    g. Name servers
11. Writing Policy
    a. Write-through: client writes to file, server immediately updates file written
    b. Delayed write at server: client writes to file, server may hold before updating file; idea is that data may not need to be written at all because client may delete it; problem is crashes loose that data
    c. Delayed write at client: writes sit at client until file is closed, then are flushed to server. Idea is that files are open for a very short time, so this cuts burden on servers
12. Cache consistency
    a. server-initiated: servers inform cache managers when data no longer valid
    b. client-initiated: client cache managers check validity of data before returning it to callers
    c. disallow caching when concurrent-write sharing: file open at multiple clients, and at least one for writing (either server tracks who has file open and how, or lock it)
    d. problem: sequential-write sharing: recently updated file (by one client) is opened for writing by a second client. Second may have outdated blocks in cache (cache timestamps, and compare with real timestamps); first

client may not have flushed cached changes yet (server requires clients to flush cache when another client opens file)

Fault-Tolerant Clock Synchronization

Introduction

The goal is to synchronize the time of clocks on different systems. The protocol includes both faulty and non-faulty clocks. The assumptions are that initially all clocks are synchronized to within some small value δ, that non-faulty clocks run at the correct rate (that is, one tick per second), and a nonfaulty process can read a non-faulty clock with an error of at most ε. In what follows, we shall assume $\varepsilon = 0$.

Notation

- n processes
- p_i process

Interactive Convergence Protocol

This assumes that no two non-faulty clocks differ by more than δ. All processes execute this protocol simultaneously.

1. p_i obtains the value of the other processes' clocks (for example, by using the OM(m) protocol). Call these values $v_1, \ldots, v_n$.
2. For all $j < n$, if $|v_j - v_i| > \mathrm{d}$, set $v_j' = v_i$. Otherwise, $v_j' = v_j$.
3. Set p_i's clock to $(\sum_j v_j')/n$.

Example

Suppose p_0, p_1, p_2, and p_3 wish to synchronize their clocks. Take $\delta = 10$, $C_0 = 2$, $C_1 = 5$,$C_2 = 8$, and $C_3 = 10$. Then: after this protocol is used, all the clocks are set to $(2 + 5 + 8 + 10)/4 = 25/4 = 6$.
Now suppose p_3's clock is faulty and drifts to $C_3 = 25$. Then:

- $C_0 = (2 + 5 + 8 + 2)/4 = 17/4 = 4$
- $C_1 = (2 + 5 + 8 + 5)/4 = 20/4 = 5$
- $C_2 = (2 + 5 + 8 + 8)/4 = 23/4 = 6$

After the next round, assuming p_3 reports any value δ away from C_0, C_1, and C_2:

- $C_0 = (4 + 5 + 6 + 4)/4 = 19/4 = 5$
- $C_1 = (4 + 5 + 6 + 5)/4 = 20/4 = 5$
- $C_2 = (4 + 5 + 6 + 6)/4 = 21/4 = 5$

Now assume C_3 is a two-faced clock. The danger is that p_3 will report a value within δ of C_1 to p_1, and not within δ of C_0 and C_2. So, begin with the same values as above, except that p_3 reports $C_3 = 1$ to p_1 and $C_3 = 25$ to p_0 and p_2:

- $C_0 = (2 + 5 + 8 + 2)/4 = 17/4 = 4$
- $C_1 = (2 + 5 + 8 + 1)/4 = 16/4 = 4$
- $C_2 = (2 + 5 + 8 + 8)/4 = 23/4 = 6$

At the next round, p_3 reports $C_3 = 15$ to p_2 and $C_3 = 0$ to p_0 and p_1.

- $C_0 = (4 + 4 + 6 + 0)/4 = 14/4 = 4$
- $C_1 = (4 + 4 + 6 + 0)/4 = 14/4 = 4$
- $C_2 = (4 + 4 + 6 + 15)/4 = 29/4 = 7$

By continuing in this fashion, p_3 can prevent the value of the clocks of the non-faulty processors from converging.

Interactive Consistency Protocol

This assumes that no two non-faulty clocks differ by more than δ. All processes execute this protocol simultaneously.

1. p_i obtains the value of the other processes' clocks (for example, by using the OM(m) protocol). Call these values $v_1, \ldots, v_n$.
2. Set p_i's clock to the median of $v_1, \ldots, v_n$.

Example

Suppose p_0, p_1, p_2, and p_3 wish to synchronize their clocks. Take $\delta = 10$, $C_0 = 2$, $C_1 = 5$,$C_2 = 8$, and $C_3 = 10$. Then: after this protocol is used, all the clocks are set to *median*(2,5, 8, 10) = (5 + 8)/2 = 6.
Now suppose p_3's clock is faulty and drifts to $C_3 = 25$. Then:

- C_0 = *median*(2, 5, 8, 25) = (5 + 8)/2 = 6
- C_1 = *median*(2, 5, 8, 25) = (5 + 8)/2 = 6
- C_2 = *median*(2, 5, 8, 25) = (5 + 8)/2 = 6

Now assume C_3 is a two-faced clock. Begin with the same values as above, except that p_3 reports $C_3 = 1$ to p_1 and $C_3 = 25$ to p_0 and p_2. All apply an agreement protocol:
p_3 invokes OM(1)

p_3 sends 1 to p1 and 25 to p_0 and p_2.
p_0 receives 25 from p_3 and invokes OM(0).
p_0 sends 25 to p_1 and p_2.
p_1 receives value 25.
p_2 receives value 25.
p_1 receives 1 from p_3 and invokes OM(0).
p_1 sends 1 to p_0 and p_2.
p_0 receives value 1.
p_2 receives value 1.
p2 receives 25 from p_3 and invokes OM(0).
p_2 sends 25 to p_0 and p_1.
p_0 receives value 25.
p_1 receives value 25.
p_0 computes majority (25, 1, 25) and takes the value at the source to be 25.
p_1 computes majority (25, 1, 25) and takes the value at the source to be 25.
p_2 computes majority (25, 1, 25) and takes the value at the source to be 25.

- C_0 = *median*(2, 5, 8, 25) = (5 + 8)/2 = 6
- C_1 = *median*(2, 5, 8, 25) = (5 + 8)/2 = 6
- C_2 = *median*(2, 5, 8, 25) = (5 + 8)/2 = 6

Notice thatall arrive at the same value.