

10/27/10

Midterm #2

OHE 122

10:00 - 11:20

during class

Wednesday

11/3 - m/w

Topics

Deadlock

Memory Management

- uniprogrammed, Fixed Partitions, Base & Bounds, Segmentation,

Virtual
Memory

Paging, Multi-level
translation

- TLB, IPT, Swapping
- Page replacement policies

Remote Procedure Calls

Projects 2 & 3 - ^{system calls} RPCs & V.M.
Protection Systems

Scenario: Tax Computation Software

- No output can be produced, when computing someone's taxes, except ~~\$~~ as a return data to the requesting process
- no disk output at all by tax software

Approach: Someone, involved in writing tax software, has come up

with two ways to "communicate" with a collaborator process

- page fault rate
- CPU percentage

If a process can manipulate the state/operation of the OS & another process is able to detect the manipulation, they can share data - 1 bit @ a time

Distributed Mutual Exclusion

2 Basic Ways

- Centralized
- Fully distributed
- no centralization

Centralized

- Have 1 mutual exclusion server
- It handles all requests to enter CRs
- A Request message is sent from the Client to the server
- If no Client is in that particular C.R., the Server replies with an OK msg
- If a Client is in that C.R., the Server queues the request

Upon ~~a~~ exiting a C.R., a Client sends a Release msg to the Server

- If a Client is waiting for that C.R., the server sends 1 OK msg to a waiting client

For this to work, C.R.s need unique identifiers

- + Correct
- + Fair
- Doesn't scale well
- Single point of failure - Server

Fully Distributed Approach

No central server

- all clients work together

No centralized decision making

- clients make group decisions

Requirements

1. Reliable communication

• No lost messages

2. Globally unique identifier for each group member

* 3. Total ordering of events

• An event, in a distributed system, is a Send/Receive

• All group members agree on the order of Sends

How to ensure "total ordering"?

Key: A group member does not process a received Request msg until it "knows" it cannot receive a Request msg from ANY group member with an earlier time stamp.

1 new requirement: Messages from a single group member are received in timestamp order & no 2 msgs have the timestamp.

Solution: Each group member maintains a "last timestamp received" table

Example: 3 Members A, B, & C

Member A

Table

A	B	C
100	99 99	101

B(99) msg arrives

Msg. Queue

~~B(99)~~, A(100), C(101)

Member B

A	B	C
100	99	101

B(99), A(100), C(101)

Member C

A	B	C
99	99	102

B(99), C(101),
C(102)

each group member

Process for Total Event Ordering

0. Receive a msg

1. Extract timestamp & member's ID

2. Update last timestamp, in my table, for that member.

3. Insert the msg into my Msg Queue in timestamp order

99 4. Extract the earliest timestamp value from my table

5. Process any msg, in timestamp order, with a timestamp \leq value from step 4.