10/27/10

Miltern #2
OHE 122
10:00 - 11:20
We does day
We does day
11/3 - mlw

Topics
Deadlock
Memory Management
· uniprogrammed, Fixed Partitions,
· uniprogrammed, Fixed Partitions, Base & Bounds, Segmentation,
Partin M. History
virtual translation
Virtual Stranslation Memory TLB, IPT, Swapping
l. Page replacement policies
Remote Procedure Calls
to 00/15
345 (CMC COL.)
Projects 283-RPCs& V.M.
Projects 283-RPCs& V.M. Protection Systems
Projects 283-RPCs& V.M. Protection Systems
Projects 283-RPCs& V.M. Protection Systems
Projects 28 3-RPCs& V.M. Protection Systems
Projects 283-RPCS& V.M. Protection Systems
Projects 283-RPCs& V.M. Protection Systems
Projects 283-RPCs& V.M. Protection Systems
Projects 283-RPCS V.M. Protection Systems
Projects 283-RPCs& V.M. Protection Systems

Scenario: Tax Computation Software

· No output can be produced,

when computing someone's

taxes, except & as & return

dutu 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 collaborater process

page fault rate

cru 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 · Centrolized · Fully distributed · no centralizeation

· Have I 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 Elient 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 rall 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

4 3. Total ordering of events

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

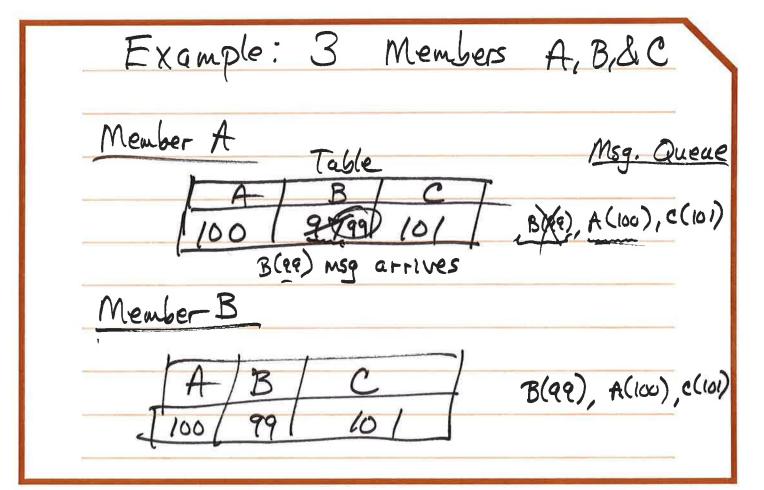
All group members agree on the order of Sends

How to ensure "total ordering"?

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

I 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



Meml	A 99	B 99	102/	13(48),	c(101),

each group member Process for Total Event Ordering Receive a msq 1. Extract timestamp & member's 2. Update last timestamp, in my table, for that member. 3. Insert the msg into my Msg Quele in timestamp order
4. Extract the earliest timestamp value from my table

<u> </u>	Process	any	msq, in	Umestang
	order,	with a	msg, in timestamp	L =
	value	from	step 4.	
-				