Research Project Presentation (April 30 in class)

- Each student must attend the presentation class
- Each group has 15 minutes to present its research project
- Each group must email me its presentation PPT file BEFORE 4 pm on April 30
- Each presentation should have no more than 20 slides
- Your presentation takes 10% towards your final grade and your peers will grade it when you present your slides
- The grader will collect a grading sheet from each group and the average score will be used as your grade in presentation
- The score from each group should be in a scale of 0 ~ 100
- The grading sheet has been emailed to you by the grader today

Incoming Due Dates

- Please submit your Assignment#3 to the grader in class on April 23. He will grade them and then return them to you in class on April 30.

- Please email your electronic version (Microsoft DOC format) of research project report to me (txie@sdsu.edu) before 4 pm on April 30.

- Also, you need to submit a printed version of your research project report to the grader on April 30 in class.

About the submission of your project report

- Submit your printed IEEE double-column format final research report in the class of April 30.

Note: Please strictly follow the “sample_doc.doc” file posted on our class web page.

Preview of Midterm Exam

- There will be five questions.
- The first one will be “True or False?”, which tests your understanding of basic concepts.
- The second one will be “Multiple Choices” and it also measures your understanding of principles of distributed systems.
- The rest 3 questions come from 3 different chapters.

Samples of Question One

- ____F____ The faults classified as omission failures refer to cases when an component not only behaves erroneously, but also fails to behave consistently when interacting with multiple other components.
- ____T____ IP packets are addressed to computers, ports belong to the TCP and UDP level.
- ____F____ If a correct process issues *multicast(g,m)* and then *multicast(g,m')*, then every correct process that delivers m' will deliver m before m'. This is called total ordering.

Sample of Question Two

- Lock timeouts can be used to resolve deadlocks. However, it has the following problems: a, b, c,

(a) Locks may be broken when there is no deadlock.

(b) If the system is overloaded, lock timeouts will happen more often and long transactions will be penalized.

(c) It is hard to select a suitable length for a timeout.

(d)Locks must be broken when there is a deadlock.

Sample of Question Two

- In Chandy and Lamport ‘snapshot’ algorithm for determining global states of distributed systems, which of the following items were assumed?

  (a) Neither channels nor process fail

  a, b, c, d

  (b) Any process may initiate a global snapshot at any time

  (c) The graph of processes and channels is strongly connected

  (d) The processes may continue their execution and send and receive normal messages while the snapshot takes place

Sample of Question Two

- The definition of uniform agreement is c

  (a) If a correct process delivers message m, then all correct processes in group(m) will eventually deliver m.

  (b) All correct processes in a group agree on a value.

  (c) If a process, whether it is correct or fails, delivers message m, then all correct processes in group(m) will eventually deliver m.

  (d) All processes in a group, whether they are correct or fail, agree on a value.

Other Sample Questions

- Understand the byzantine generals problem.
- Understand the three choices of RMI invocation semantics.
- Understand the logical time vector.

Sample Question

A client attempts to synchronize with a time server. It records the round-trip times and timestamps returned by the server in the table below.

Which of these times should it use to set its clock? To what time should it set it? Estimate the accuracy of the setting with respect to the server's clock. If it is known that the time between sending and receiving a message in the system concerned is at least 8 ms, do your answers change?

Round-trip (ms)	*Time (hr:min:sec)*
22	*10:54:23.674*
25	*10:54:25.450*
20	*10:54:28.342*

Solution

The client should choose the minimum round-trip time of 20 ms = 0.02 s. It then estimates the current time to be 10:54:28.342 + 0.02/2 = 10:54:28.352. The accuracy is ± 10 ms.

If the minimum message transfer time is known to be 8 ms, then the setting remains the same but the accuracy improves to ± 2 ms.

Sample Question

- In the Bully algorithm, a recovering process starts an election and will become the new coordinator if it has a higher identifier than the current incumbent. Is this a necessary feature of the algorithm?

Solution

- First note that this is an undesirable feature if there is no advantage to using a higher-numbered process: the re-election is wasteful. However, the numbering of processes may reflect their relative advantage (for example, with higher-numbered processes executing at faster machines). In this case, the advantage may be worth the re-election costs. Re-election costs include the message rounds needed to implement the election; they also may include application-specific state transfer from the old coordinator to the new coordinator.

Solution (continued)

- To avoid a re-election, a recovering process could merely send a *requestStatus* message to successive lower-numbered processes to discover whether another process is already elected, and elect itself only if it receives a negative response. Thereafter, the algorithm can operate as before: if the newly-recovered process discovers the coordinator to have failed, or if it receives an election message, it sends a coordinator message to the remaining processes.

Sample Question

A server manages the objects a_1, a_2,... a_n. The server provides two operations for its clients:

read (i) returns the value of a_i;

write(i, Value) assigns *Value* to a_i.

The transactions *T* and *U* are defined as follows:

T: x= read (j); y = read (i); write(j, 44); write(i, 33);
U: x= read(k); write(i, 55); y = read (j); write(k, 66).

Give three serially equivalent interleavings of the transactions *T* and *U*.

Solution

The interleavings of T and U are serially equivalent if they produce the same outputs (in x and y) and have the same effect on the objects as some serial execution of the two transactions. The two possible serial executions and their effects are:

If T runs before U: $x_T = a_j^0$; $y_T = a_i^0$; $x_U = a_k^0$; $y_U = 44$; $a_i = 55$; $a_j = 44$; $a_k = 66$.

If U runs before T: $x_T = a_j^0$; $y_T = 55$; $x_U = a_k^0$; $y_U = a_j^0$; $a_i = 33$; $a_j = 44$; $a_k = 66$,

where x_T and y_T are the values of x and y in transaction T; x_U and y_U are the values of x and y in transaction U and a_i^0, a_j^0 and a_k^0, are the initial values of a_i, a_j and a_k

Solution (continued)

We show two examples of serially equivalent interleavings:

A

T	*U*
x:=read(j)	
	x:= read(k)
	write(i, 55)
y:=read (i)	
	y:=read (j)
	write(k, 66)
write(j, 44)	
write(i, 33)	

B

T	*U*
x:=read(j)	
y:=read (i)	
	x:= read(k)
write(j, 44)	
write(i, 33)	
	write(i, 55)
	y:=read (j)
	write(k, 66)

A is equivalent to *U* before *T*. y_T gets the value of 55 written by *U* and at the end a_i =33; a_j =44; a_k = 66.

B is equivalent to *T* before *U*. y_U gets the value of 44 written by *T* and at the end a_i =55; a_j =44; a_k = 66.