Name _____ _____ Student ID _____

Cpt. S. 464/564 Distributed Systems Concepts and Programming

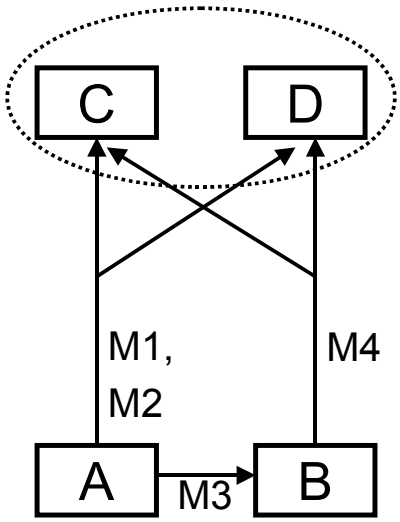
Midterm Examination
October 17, 2001

183 possible points (464) ____ check here if 464 student
237 possible points (564) ____ check here if 564 student

22.5% of final grade
8 Pages

Write the answers to the following questions. If you need more space, use the back of a sheet and clearly mention this on the front. (You generally should not; answers are not expected to take longer than the space provided, but extra length is not penalized so long as it is coherent and not unfocused rambling.) If you are not sure about some part of the question, state any assumptions you are making about the question, then answer it based on those assumptions. If they are reasonable, there will be no points off.

1) [27 points] Ordering (c.f. text 2.7)



In the example to the left, the group that the ordered multicast is sent to is designated by the oval, and represents participants C and D.
In our scenario, (in real time order) A sends M1 then M2 to the group, sends M3 to B, then B sends M4 to the group.

a) [6 points] Give all examples of legal orderings of {M1,M2,M4} that could be delivered to the group, assuming FIFO ordering.

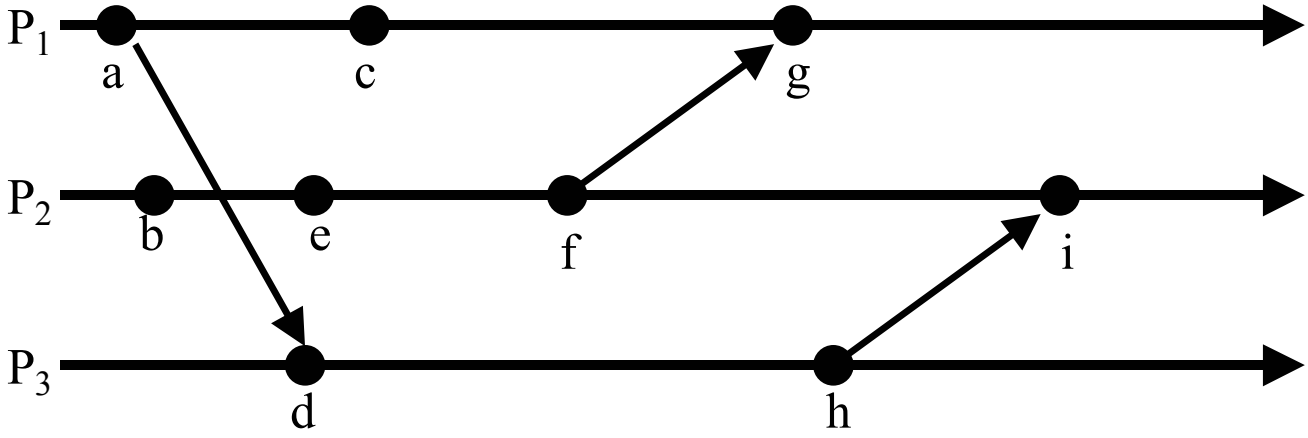
b) [6 points] Give all examples of legal orderings of {M1,M2,M4} that could be delivered to the group, assuming causal ordering.

c) [6 points] Give all examples of legal orderings of {M1,M2,M4} that could be delivered to the group, assuming causal ordering and that M3 was never sent.

d) [9 points] In general (not necessarily the scenario above), which of the following orderings can group members get delivered a different ordering of messages: FIFO, causal, total? For example, C might get delivered {M5,M6,M7} while D gets {M6,M5,M7}. Put a check by those orderings which this can happen for:

__ _FIFO __ _causal ____total

2) **[20 points] Cuts** (c.f. text 2.9.1)



a) [4 points] Draw an inconsistent cut on the figure above, and mark it "IC"

b) [4 points] Draw a strongly consistent cut on the figure above, and mark it "SCC"

c) [4 points] Draw a consistent cut on the figure above, and mark it "CC"

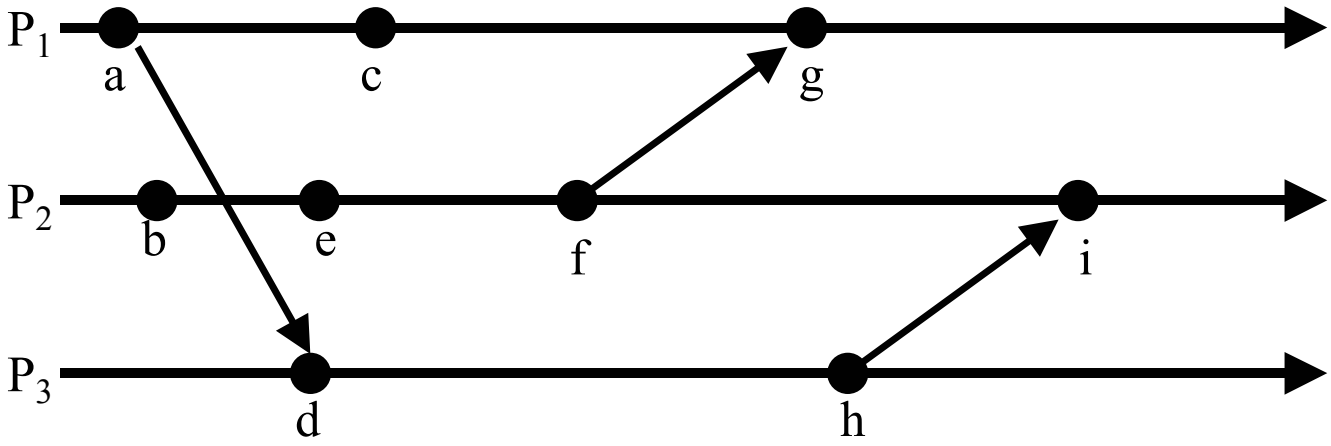
d) [8 points] The frontier of a cut is defined as the latest events from each process represented in the cut's states. For example, the frontier {c,e,d} represents a strongly consistent cut. Give two examples of a frontier of an inconsistent cut that can be made by changing only one of the events in the frontier {c,e,d}.

3) [16 points] Local vs. Remote Procedure Calls (c.f. Intro)

List four potential assumptions that are normally true for a local procedure call yet are not usually true for a remote procedure call (RPC). For each assumption, explain in a sentence or two how or why it is not true for an RPC.

4) [25 points] Logical Time (c.f. text 2.7.1;)

In the following diagram, P_i denotes process *i*, a dot on or just on top of a process's timeline denotes an event at the process, a thick arrow between two dots denotes a message sent between two processes. Physical time increases from left to right, in the direction of the thin horizontal arrows. (Note: an arrow denotes a message between two processes, and not any other process's time line the arrow crosses; i.e., the message from **a** to **d** is from P_1 to P_3, and does not involve



P_2.)

a) [8 points] On the diagram above, write the Logical Clock (LC) time at its processor for each event to the left of the dot for that event.

b) [17 points] Fill out the empty cells in the table to give the relations between each event: "→" denotes "happened before", "←" denotes "happened after", and "||" denotes "concurrent". (For examples of this notation, because there is a message from 'a' to 'd', it is filled in "→" in the [a,d] cell and "←" in the [d,a] cell. Also, 'b' and 'a' are concurrent, and are so marked.)

	a	b	c	d	e	f	g	h	i
a	—	||	→	→	||	||	→	→	→
b	||	—							
c	←		—						
d	←			—					
e	||				—				
f	||					—			
g	←						—		
h	←							—	
i	←								—

5) [564 only; 14 points] CORBA smart proxies/stubs vs. interceptors (c.f. "CORBA Hooks")

a) [3 points] What is a CORBA smart proxy (also called "smart stub")?

A CORBA smart proxy is an object inserted between the client and the ORB's proxy. It has the exact same methods as the proxy (and the remote servant), and can be coded by the application programmer.

b) [3 points] What is a CORBA interceptor?

A CORBA interceptor is an object loaded in the ORB that can "intercept" requests and replies --- get a callback when they occur, and take action based on this.

c) [4 points] Name one use that smart proxies are a better choice for than interceptors, and why.

A smart proxy is a better choice for caching, or any time you have applications-specific behavior or adaptation, because the parameters are unpacked nicely for you, so it is just like programming a servant implementation.

d) [4 points] Name one use that interceptors are a better choice for than smart proxies, and why.

An interceptor is a better choice any time you need to write some system modifications that will occur for every interface type being used, because to do this with smart proxies would require writing it for every interface in the system. Examples: logging, debugging, authentication/access control.

6) [17 points] Synchronous vs. Asynchronous Invocations

a) [3 points] What is a synchronous invocation interface?

b) [3 points] What is an asynchronous invocation interface?

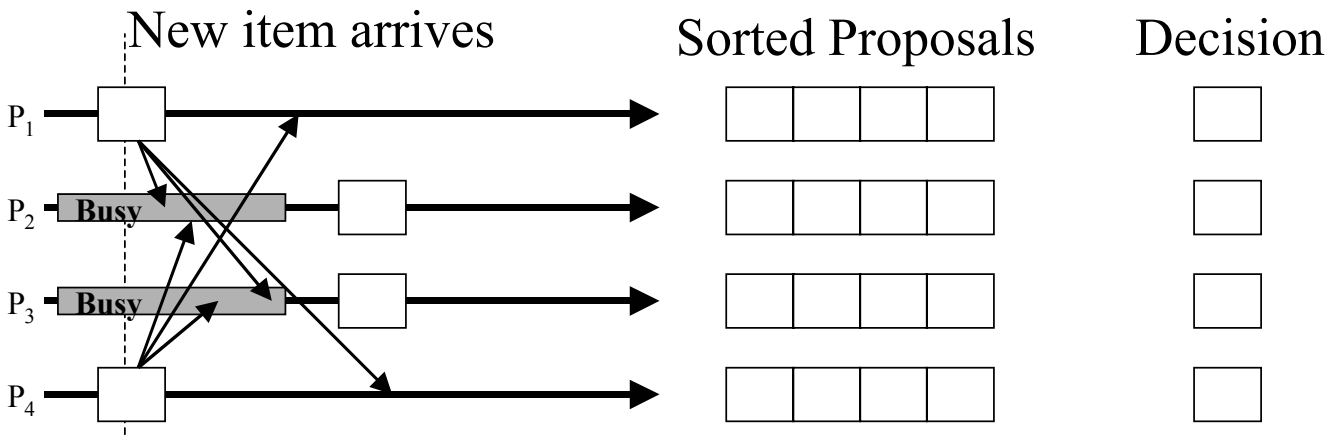
c) [3 points] Describe one advantage of synchronous invocation interfaces.

d) [3 points] Describe one advantage of asynchronous invocation interfaces.

e) [5 points] Name a technique or mechanism that allows synchronous invocation interfaces to be used with most or all of the advantages of asynchronous invocations.

7) [28 or 49 points] Distributed Consensus (c.f. text 2.9.2)

Consider the consensus example below, using an algorithm identical to that described in the textbook and discussed in class, with a set of assembly line machines reaching consensus on which one will handle the next item. In the example below, when the new item arrives (denoted by the dashed line), P_1 and P_4 are not busy and send in an immediate proposal, but P_2 and P_3 are busy then and send in the proposal at the time indicated by the blank box on their timelines.



a) [8 points] Write in the blank boxes on the timeline diagrams the values that each of the processes will propose.

b) [8 points] Draw in arrows depicting messages that P_2 and P_3 would send in this scenario

c) [8 points] Write in the proposals' values after they have been sorted, indicating the proposals that each process would receive, under the heading "Sorted Proposals".

d) [4 points] Write in the consensus decision that each process will make, assuming it uses the "maximum" function to decide, under the heading "Decision".

e) [564 only; 5 points] What kind or category of function must each process use to decide?

f) [564 only; 4 points] Give two examples of this kind of a function.

g) [564 only; 4 points] Give two examples of functions that are not of this kind and thus would not be appropriate for use.

h) [564 only; 8 points] Name the strength ordering and reliability that this protocol must have to work correctly

8) [9 or 21 points] Partitions (c.f. text 2.9.7)

a) [4 points] Name the bad thing typically happens in general to an application's state if its replicas are separated into two or more partitions yet all its replicas continue to service requests.

b) [5 points] Describe how the primary partition technique prevents this bad thing from happening.

c) [564 only; 7 points] Briefly describe the technique discussed in class that allows more of the replicas to continue to service requests, compared to the baseline primary partition technique. Under what constraints or assumptions or caveats can these extra replicas make progress?

d) [564 only; 5 points] Can this constraint or assumption or caveat be safely hidden from the clients of the extra replicas, or should they be informed? Justify your answer.

9) [14 or 21 points] Fundamental Tradeoffs (c.f. text 3.2.1)

a) [6 points] List and define the two design choice possibilities with respect to distribution transparency.

b) [4 points] Define data shipping.

c) [4 points] Define function shipping.

d) [564 only; 7 points] Give a concrete scenario where code shipping is beneficial but data shipping is not. Justify your choice in terms of runtime parameters of the application or system.

10) [12 points] Reasons to Distribute Applications

One main reason to distribute an application program is when the problem being solved has an inherently distributed nature. Describe two real or made-up distributed applications that fall into this category, in a brief paragraph each. Be sure to explain why the problem being solved is inherently distributed

11) [15 points] Concurrency Control (c.f. text 2.10.3 slides Chap2c#23-24)

Add the use of lock variables into the code below to provide the most possible concurrent access yet providing consistency and preventing intermediate results from being accessible. Your solution should not allow deadlock. Note: you do not need to declare your lock variables, just use them, even just writing "**lock**(X)" or "**unlock**(Y)".

Database: **int** A, B, C, D, E, F

transaction AB is
lock(A); lock(B)
A = A*2; B=A*3;
unlock(A); unlock(B)
end;

transaction CD is
lock(C); lock(D)
C = C*4; C=D*5;
unlock(C); unlock(D)
end;

transaction EF is
lock(E); lock(F)
E = E*6; E=F*7;
unlock(E); unlock(F)
end;

transaction ABC is
lock(A); lock(B); lock(C);
A = A*6; B=A*4; C=A+B;
unlock(A); unlock(B); unlock(C)
end;

transaction DEF is
lock(D); lock(E); lock(F);
D = D*2; E=D+3; F=D-E;
unlock(D); unlock(E); unlock(F);
end