CptS 464/564 Homework #2

Given: Thursday, February 13, 2003
Due: Tuesday, February 25 at the beginning of class
Weight: 5% of final grade

This homework must be done with a word processor, not handwritten.

.

1) [20 points – All students] Understanding IDL-generated Code

The IDL interface bar:

```
interface bar {
  long method2(in long param3, out float param4);
};
```

when run through the Orbacus jidl IDL compiler creates files `_barStub.java` and `barPOA.java` for the client-side proxy (stub) and the server-side skeleton, respectively, as well as two files containing Java interfaces that are implemented by the proxy and skeleton. These files can be found in the lab in /net/orbacus/cs564/HW2/. Print out bar.idl, barPOA.java, and _barStub.java. Mark up the printout in the following ways:

- interface name (bar): draw an oval around
- method name (method2): draw a rectangle around
- parameter name (param3, param4) & return value: draw a line under.

Make sure to underline ***all*** places where the (unnamed) return value, which will eventually be returned to the client, appears. Your markings do not need to be computer-drawn: pencil or pen is fine, though if you want to pull the text into a word processor that is fine too. (FYI, the IDL to Java mapping uses "Holders" because Java is pass-by-value.)

The point of this exercise is to help you become familiar with what goes on in the stubs and skeletons. Your should strive to understand what is happening in each of the files as well as performing the rather mechanical task of finding all the identifiers. Ask yourself "why is this identifier used here" for each occurrence you find.

2) [20 points – All students] Asynchronous Method Invocations

Explain briefly the advantages and disadvantages of asynchronous method invocations compared to synchronous ones (assuming that threads are not used. Describe a concrete situation where asynchronous invocations are advantageous and one where synchronous invocations are advantageous.

3) [40 points – All students] Ordered delivery

For each of the following, explain briefly in general terms the kinds of situations where a delivery system providing the weaker ordering is insufficient but one providing the stronger ordering is sufficient.

a) Unordered insufficient, but FIFO sufficient
b) FIFO insufficient, but causal sufficient
c) Causal insufficient, but total sufficient
d) Total (based on logical time) insufficient, but temporal-based total sufficient

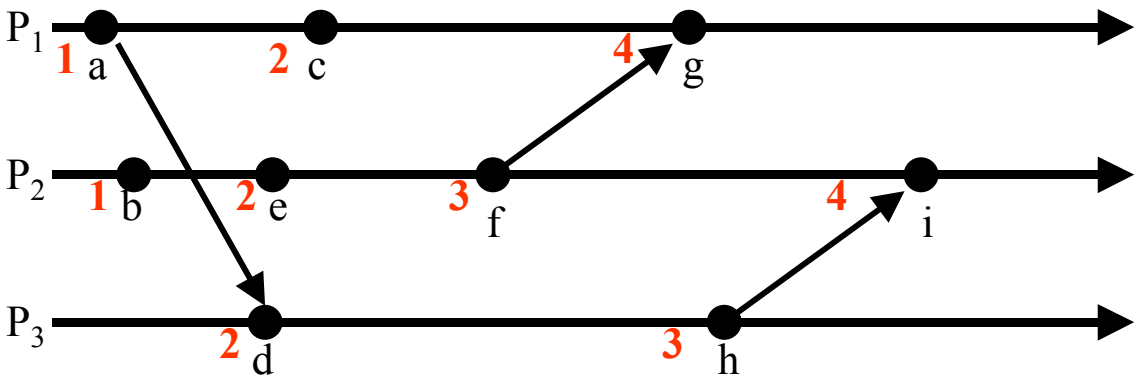
4) [30 points; 564 students only] Logical Time paper by Lamport

Graduate students: read Leslie Lamport's seminal paper from July 1978 *Communications of the ACM*, entitled "Time, clocks, and the ordering of events in a distributed system". It can be found at http://doi.acm.org/10.1145/359545.359563 .

Explain what can go wrong in the resource scheduling algorithm on p. 561 in the presence of crash failures, where a host goes down cleanly (not making any errors, sending bogus messages, etc) but undetectably. In your explanation, identify 3 situations (role and step in protocol) where a crash failure causes problems, and briefly describe what would go wrong (this happens, then that happens, then something else happens, and then –oops—the system is no longer working the way we expected). For each of the failures, propose a solution outline. Your solution outline may assume the presence of any service discussed in Chapter 2 in the textbook (specify which one, of course, and how you use it).

5) [20 points – All students] Event ordering

Refer to the following diagram. Fill out the empty cells in the table to give the relations between each event: "→" denotes "the row event happened before the column event", "←" denotes "the



row event happened after the column event", and "||" denotes "the two events are concurrent". (For example, because 'a' precedes 'c' in P_1, box (a,c) contains "→", and box (c,a) contains "←". Also, 'b' and 'a' are concurrent, and are so marked. (You may print this page and fill in the boxes by hand if you wish.)

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