University of Rochester **Mobile Computing (CSC290e)**

Final Exam

Amy Murphy

12 December 2001

Read before beginning: Please write clearly. Illegible answers cannot be graded. Be sure to identify all of your answers in your blue book. All questions are worth 10 points. Answer all questions.

While I have tried to make the questions as unambiguous as possible, if you need to make any assumptions in order to continue, state these as clearly as possible as part of your answer. In the name of fairness, I would like to avoid answering questions during the exam.

Please turn in both your bluebook and this exam sheet. Exams can be picked up after Tuesday of next week.

1. **Bluetooth and 802.11.**: Briefly explain why Bluetooth and 802.11 can conflict with one another when they are running in the same physical space. Describe one option which would solve this problem (your solution need not be complicated).

   Both technologies operate in the 2.4 GHz range. Even though they use different means to divide the available bandwidth, the radio waves can still cause interference on one another. One way to fix this is to assign each technology to its own frequency range, avoiding possible interference entirely. Another alternative is to redesign the protocols to work together, effectively making one technology out of two.

2. **Mobile IP.**: One of the advantages of IP tunneling is to allow the incremental deployment of of new protocols in the existing Internet.

   (a) What is IP tunneling?

   Effectively using regular IP to create a path between two non-IP subnetworks. The IP backbone hides the protocol differences between the subnets.

   (b) How is IP tunneling used in Mobile IP?

   In Mobile IP, an IP tunnel is set up between the home agent and the mobile host (MH), hiding the mobility from the correspondent node (CN). Basically the original packet from the CN is encapsulated in an IP packet and sent to the MN.

   (c) If a mobile host uses a foreign agent to receive and forward its messages, in most cases the mobile host need not even care about the tunneling and IP-in-IP encapsulation. Why is this *not* true in the case of multicast messages being forwarded from the home to the foreign agent and then to the mobile host?

   If the foreign agent simply decapsulates the multicast packet, it will have the destination of the group address, and will not be properly received by the mobile host which is to receive the packet.

3. **Mobile Agents.**: Briefly define strong and weak mobility. It might be helpful in your definitions to give an example of the behavior of a strongly mobile agent vs. a weakly mobile agent. Which is possible in Java?

   Strong mobility moves code, data, and execution state, allowing an agent to migrate in the middle of a method execution and resume exactly where it left off when it arrives at the destination. Weak mobility just moves code and data, but leaves the execution state. A weakly mobile agent typically restarts at the beginning of the main (or run) method. Only weak mobility is enabled by Java.

4. **Mobile Code Analysis.**: The figure on the last page shows the analysis of the traffic overhead for a single invocation of a network management application over one to twenty nodes using mobile

agents, client server, and remote evaluation. At each host, 20kb of data are collected. Briefly explain why the mobile agent curve is shaped the way it is, and name/describe the technique which helps to flatten out this curve.

Each time the mobile agent moves, it gets larger because it is carrying all the data from all of its previous hops. In the other scenarios, each piece of data is only sent once on the network, while with mobile agents, the data traverses the network multiple times (once for each host in the path visited by the mobile agent). Semantic compression flattens out the curve, and eventually makes mobile agents a viable implementation option. With semantic compression, the data is processed before the agent migrates, allowing it to carry less data than would need to be transmitted in the other options.

5. **Disconnected Transitive Communication.**: The core DTC model shows how a single message propagates through the hosts of the mobile ad hoc network. One alternative is to replicate the message when it goes to the next hop, effectively producing several copies of the same message which traverse independently through the network. Discuss one advantage and one disadvantage of this alternative.

   The primary advantage is an increase in the likelihood that the message will reach the destination, or that it will reach it more quickly. Disadvantages include the overhead in the storage required at all the nodes to support the replication, and the overhead in the messages required to propagate all the replicas.

6. **Lime.**: When two processes execute on two different host computers, you cannot make any assumptions about the order in which the operations are executed. The same is also true for operations which execute in different processes on the same machine. In LIME, both issues arise.

   Consider the scenario with two hosts, `host1` and `host2`, which are within communication range of one another. Assume each host is running LIME, the LIME servers are engaged, Agent 0 is on `host1`, Agent 1 is on `host2`, and each agent has a shared, default tuple space.

   The pseudocode below shows a few operations for each agent (line numbers are given in parenthesis on the left of each statement). Because of the unpredictable order of execution of operations, you cannot predict with complete accuracy the final values of `t1` and `t2`.

   To answer this problem, list all the possible values for t1 and t2, and for each set of values, give one interleaving (e.g., a sequence of line numbers) that produces those values. Hint, there are more than 2 and less than 5 possible output value combinations. Assume that initially the tuple spaces are empty.

   Agent 0
   ```
   (1) p=<string?>
   (2) t1 = rdp(host1, UNSPECIFIED, p)
   (3) out(<hello>)
   (4) t2 = rd(p)
   ```

   Agent 1
   ```
   (11) t=<hi>
   (12) out(t)
   (13) migrate(host1)
   ```

   (a) t1=null t2=hi: 11 12 13 1 2 3 4

   (b) t1=null t2=hello: 11 12 13 1 2 3 4

   (c) t1=hi, t2=hi: 11 12 1 2 3 13 4

   (d) t1=hi, t2=hello: 11 12 1 2 3 13 4

7. **Distributed and Mobile Computing.**: (Note: this question is long to read, but NOT difficult to answer. Read carefully.) In the first week of class class, we talked about the fact that distributed consensus is impossible. Basically, this means that it is impossible to know (in a faulty

environment) that both participants in a communication agree on a particular outcome. In terms of the "coordinated attack" problem, this means that the two generals cannot agree on whether or not to *attack at dawn*.

In a mobile setting, the link between any pair of hosts is unpredictable (i.e., faulty). Think about LIME, a middleware which works in the mobile environment, and in particular, think about the `out(otherAgent, t)` operation, where the `otherAgent` is on a connected host when the first agent issues the `out` operation.

When this tuple, `t`, is being transmitted to the other host, if the link between the two hosts disappears, distributed consensus tells us that we cannot know the actual status of the tuple (i.e., if it actually arrived at the other host or not). Another way to think of this is that the sender and receiver can each perceive that the transmission was successful or was a failure. If transmission is using a standard TCP the following three cases are possible:

Sender Perceives	Receiver Perceives
success	success
failure	success
failure	failure

As you know, the current version of LIME does not handle this kind of sudden disconnection, but suppose we were to extend it to do so. Think about an extension to your Maze application where the players were finding gold as they visit the rooms of the maze, and put that gold into their tuple space. At some point, one of the players decides to give 10 pieces of its gold to another player (on a different host, in the same room) using the `out(otherAgent,t)` operation. Suppose a disconnection occurred while the tuple containing the gold, `t`, was being transferred, and the sender perceived failure.

(a) Let our modified version of LIME choose to insert the tuple as a misplaced tuple into the sender's tuple space. What is the result to the total amount of gold in the game in the case where the receiver perceived success?

The total amount of gold will increase because the piece of gold both reaches the destination (the receiver perceives success) and the sender reinserts it as misplaced into the tuple space.

(b) Let our modified version of LIME choose to ignore the failure and do nothing. What is the result to the total amount of gold in the game in the case where the receiver perceived failure?

The total amount of gold will decrease because the piece of gold did not reach the destination (the receiver perceives failure) and the sender does nothing to compensate for this loss.

(c) Suppose that we were not willing to accept either of these consequences. Briefly describe one option. Remember, in the maze we are guaranteed that at the end of the game, the players will meet back in the first room of the maze.

We could have the sender remember each piece of gold which it transferred, and for which transmission was perceived as failure. Then, when the sender and receiver meet at the end of the game, the sender can ask the receiver whether or not the transmission was successful or not. This also requires the receiver to remember the last piece of gold (tuple) successfully received from each sender.