Homework 2 (Due: 5pm Fri 4/29)

1. Develop your own RFID protocol.

The goal of this part of the homework is for you to design an RFID inventory protocol. You will code up your protocol as a program and run it in an emulated environment to see how well it performs. *We recommend that you do this part with a partner (only two people per team).*

Scenario

The RFID systems we explore contain one RFID reader and multiple tags. The goal of the RFID reader is to perform an "inventory", in which it gathers the identifiers of the set of RFID tags that are in range. The rules of communication in our simplified world are as follows. The reader broadcasts messages that may be heard by all tags, and provides the tags with power so that they may send an immediate reply back to the reader. Tags can only hear the reader; they cannot hear if there are replies by other tags. If multiple tags transmit then there is a collision at the reader and none of the messages can be successfully received, but the reader knows there was an error. Since this is wireless, any message from the reader or tags may be received in error independently at each receiver due to noise. These errors are indistinguishable from a collision.

Our Strawman Protocol

We have provided a strawman protocol to perform the inventory. It runs at both reader and tags. You can run it as:

```
java RFIDSim <#trials> <#tags per trial> <% channel error rate>
```

The number of tags is the key parameter you will be varying as you experiment. You can set the other parameters to reasonable numbers and explore. 1% is a small error rate to remind you that messages do get lost sometimes; 10% is a better value to stress test that your protocol works when there are errors; if you find all tags at 0% error but not 10% error then your protocol may be flawed. The number of trials should be enough that the variation in the average is small. 100 is likely ample, and a smaller number of trials is OK if the running time gets large.

The strawman protocol works as follows. Tags start in an uninventoried state. The reader repeatedly sends out query messages. If an uninventoried tag gets a query, it picks a random number in the range 0 to 31, and only if the number is 0 does it reply with its EPC. If the reader gets an EPC, it adds it to its list and sends an ack. If a tag that just sent an EPC gets an ack as its next message then it sets its inventoried flag and remains quiet. If the reader hears no replies or errors in 32 consecutive queries then it decides it has heard from all the tags and returns.

The simulator then prints out the key evaluation parameters which are the number of bytes sent/received over the channel and average tags missed. The number of bytes is a measure of simulated time. This is your performance metric -- lower is better. The average tags missed is a check that the protocol is finding all of the tags; more than an occasional missed tag likely means your protocol is broken. Other statistics printed are simply for your information.

Your RFID Protocol

Your job is to design an RFID inventory protocol that finds all of the tags in the shortest amount of time. It should give consistently good performance over a range of numbers of tags. As you can imagine, our strawman is not very good. For very few tags, say 2, there will often be no reply to a query, wasting time. For many tags, say 100, there will often be collisions in reply to query, again wasting time. It is also the case that our strawman can easily miss tags by returning too early. Your protocol should be (much) better.

Your protocol comprises the messages sent between the reader and tags and vice versa, as well as what happens at the reader and at the tags in response to these messages and how to generate the messages in the first place. You should code up your protocol in both RFIDReader and RFIDTag, replacing the strawman protocol. *We strongly advise you to develop your protocol in stages, testing that each stage you add works (and presumably improves performance) rather than coding and testing performance at the end.*

You are free to design your protocol as you like but should act plausibly by observing at least these rules, which are intended as a reasonable model of the scenario:

1. The reader has no a priori knowledge of the number of tags or their identifiers. You cannot assume the identifiers are sequential.
2. The reader and tags can communicate with each other only via the channel. This means that your tag cannot access any reader state directly or vice versa. You must not change RFIDChannel.
3. Tags cannot communicate directly with each other via the channel (or by accessing state). They can only exchange messages with the reader.
4. Tags can perform computation and keep a modest amount of state.
5. Tags can only reply with messages in response to a reader command; unlike the reader they cannot set timers to wake up later and take action.

As you design your protocol, draw on the ideas we have discussed in class. Multi-access protocols like Ethernet are clearly relevant. You will have to think about reliability and errors too. You are welcome to draw on the text (section 4.7) and read up on RFID tags, including the EPC Class 1 Gen 2 RFID specification (see link on the Homework page). However, realize that 1) you do not have to implement the real RFID standard; and 2) there is much detail you can read about RFID that is superfluous for this exercise.

Questions and Turn-in

1. Turn in you the code for your protocol so that we can run it. Clearly separate your protocol implementation from support code such, e.g., GUIs.
2. Turn in a brief description of how your protocol works at a high-level. Point out the important features and *why* you designed it the way you did. Contrast it with the strawman to tell us why it is better. [Target half a page.]
3. Give us a graph to show how well your design works versus the strawman. For each design, plot the average number of bytes sent/received and average missed tags versus the number of tags, where the number of tags ideally ranges from 1 to 256. Stop with fewer tags if the

computational time becomes too large. (You will want to generate these graphs as you experiment with your protocol, not simply at the end, to see how well it is doing.)

4. Give us a graph to show how well your design works as a MAC. Plot the fraction of EPC identifier bytes out of the total bytes sent/received versus the number of tags. The non-EPC identifier bytes represent the overhead of coordinating tags.
5. Suppose that when two or more tags reply the reader does not always get a collision. If one tag has a stronger signal than the other, that tag may be received, with no indication that other tags sent a message. This is called capture, and it is a more realistic physical model in some cases. Give a scenario to describe how capture can cause an inventory protocol to fail when the protocol worked correctly without capture.

2. Selfish MAC Protocols.

The CSMA/CD and CSMA/CA multi-access protocols that we studied assume that each node implements the protocol correctly. Suppose you want to build a selfish node that gets more bandwidth when it competes with a regular node that correctly runs the protocol.

a) For the CSMA/CD and CSMA/CA cases in turn, describe the how your selfish node would change the protocol. Roughly how much of the channel bandwidth would the selfish and the regular node get when they compete to send packets?
b) Can you build these selfish nodes? Say whether this is easy or hard and why if you are (i) a consumer who buys a computer; or (ii) a manufacturer of NICs.
c) You have successfully built your selfish node, and are so happy with it that you make another one for your classmate. For the CSMA/CD and CSMA/CA cases in turn, describe what happens when the two selfish nodes compete with each other. Roughly how much of the channel bandwidth would each selfish node get?

3. Confused Bridges

Suppose you have a switched LAN and two computers, A and B, with the same MAC address are attached to different parts of the LAN. What happens?

a) Specifically, say what happens at the switches for the cases of C sending a packet to A, C sending to B, A sending to C, and B sending to C assuming these computers are the only hosts on the switched LAN.
b) Now let there be more hosts on the LAN that may send traffic. How does this change your answer to part a)?

4. Textbook

Questions 4.16, 4.40, 5.6, 5.12

□□□