CSE 466 Exam II — 15 Dec 2004

Name: ________________________________ ID#: ______________________________

Please read through the *entire* examination first! This exam was intended to be completed in 50 minutes. There are 4 problems for a total of 100 points. The point value of each problem is indicated in the table below. There will be generous partial credit so make sure to get to every problem.

Each problem is on a separate sheet of paper. Write you answer *neatly* in the space provided. Do not use any other paper to hand in your answers.

Good luck.

Problem	Max Score	Score
1	20	
2	30	
3	30	
4	20	
TOTAL	100	

1. TinyOS Terms (20 points)

Define the following terms related to features of the TinyOS operating system and provide a basic definition as well as an example of when each feature would be used. Relate your examples to the Flock project, if possible.

a) “Provide” an interface

b) “Use” an interface

c) Hardware interrupt

d) Task

2. TinyOS Abstractions (30 points)

GenericComm uses a TinyOS parameterized interfaces for messages that are received.

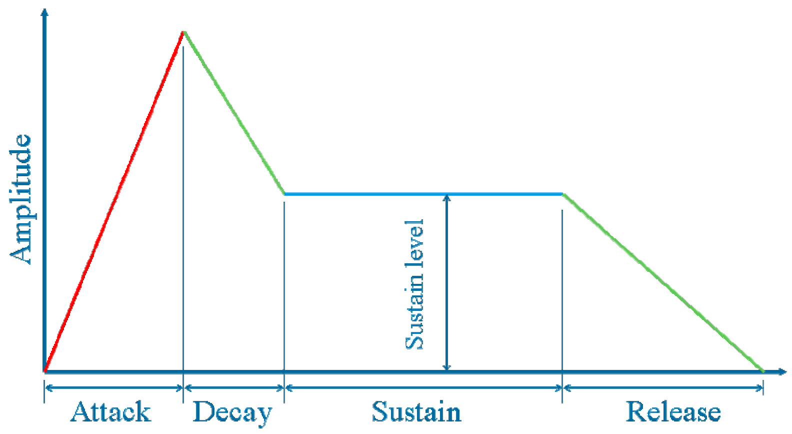
a) Describe how this feature relates to the Active Message model used for TinyOS radio packets.

b) What if we didn’t have parameterized interfaces? Every module wanting to use GenericComm would have to connect to the same interface. What would have to change to make this work? Describe any changes that would need to be made at the interface, in GenericComm, and in modules using the interface.

c) Are there any performance implications to not having parameterized interfaces?

3. Sound Generation (30 points)

Describe a strategy for generating a sound waveform with an ADSR (attack-decay-sustain-release) envelope as shown below.



a) Start by describing how an audio frequency is generated. How is the frequency varied?

b) In the flock project, we did not vary the amplitude of our sound. How would you go about varying the amplitude using the same scheme for sound generation?

c) Describe the timers you’d need, what they would be used for, and how their events would be handled.

4. Radio Protocols (20 points)

One of the principal issues in saving radio power is deciding when to power-up the radio so that it can hear other packets and when to put it to sleep without missing communications. This is a synchronization problem. One way to deal with it is the method you used in constructing your flock node. Basically, you made sure to keep the radio operating for a long enough period of time that you would hear most packets from neighbors (that had just finished their bird call).

We've devised a new method. It uses a microphone to detect a special note at the end of every bird call. We'll have each song end with an inaudible note in the ultrasound range (25KHz). We'll add a microphone to our birds so that they can listen to nearby songs. If they hear sound at that high frequency, they'll then know to listen for a radio packet as soon as it ends.

Discuss the following aspects of this approach:

a) At what frequency must the ultrasound be sampled?

b) Assuming that our microcontroller runs at 10MHz and that it takes X instructions per *millisecond* to handle the radio stack and Y instructions for each sample of ultrasound, can we handle simultaneous radio transmission and ultrasound detection on this platform? Derive an inequality that relates X and Y assuming the sample rate for the ultrasound is as your answer for part (a).

c) Typical multiple access networks use exponential backoff when a collision is detected after two or more devices try to transmit at the same time. What could we do to exploit the ultrasound output at the end of every bird call to minimize the probability of this initial collision?