Fishnet Assignment 4: An Audio Application

Due: Monday Mar 10, 2003 at the beginning of class. Out: Friday Feb 21, 2002.
CSE/EE461 Winter 2003; Wetherall.

The purpose of this assignment is for you to understand the structure of network applications by developing an audio application that runs in a separate process and communicates over the network by using the fishnet node you have already developed.

1 Amphibian

For the first half of this assignment, your goal is to modify your transport protocol so that it can be used by applications that run on your IPAQ but outside of your Fishnet node. For reasons of modularity, applications that send data are not usually implemented in the same program as the transport protocol itself, in the way that you have with the transfer command. Rather, Web servers and so forth run in separate processes than the kernel, where TCP/IP is implemented. Specifically, your job is to get the separate fishperf application we provide (think of it as a replacement for the transfer command) to run using your transport protocol. To do this, you will need to learn about Amphibian, a new part of Fishnet that provides a "faux socket" interface between applications and Fishnet nodes. Sockets are the API use by real applications such as Web servers.

Amphibian works differently when used in applications and inside a Fishnet node. Applications include the header file amphibian_app.h and link against libamphibian_app. They use a "faux socket" API to access the Fishnet, e.g., fish_socket() instead of the real socket() call. We have provided you with fishperf in the distribution, an application that uses Amphibian to send data between Fishnet nodes and times the transfer to see how well the transport protocol is working. You can look at its source code to see how it uses these calls. You will use fishperf to test your node when you have made it Amphibian-ready, rather than the in-node transfer of a test pattern as before.

On the other side, to make fishperf work, your Fishnet node needs to do three things. First, include the header file amphibian.h to access Amphibian functionality (but you still link against libfish, as before). Second, register callbacks with Amphibian so that it knows which function to call when applications open connections, send data, close connections, and so forth. Third, call other Amphibian functions directly to initialize Amphibian, deliver data to a waiting application, complete a waiting connection, and so forth.

You should proceed using the following steps:

- Read amphibian.h to learn more about the API. As with fish.h, the file amphibian.h is the definitive version of the Amphibian in-node API. Further documentation on the real socket interface, on which Amphibian faux sockets are modeled, is linked off the course web pages.

- Look at your test pattern command. Wherever that command calls or is called by your transport protocol code there is an interaction between an application and your transport protocol that you will need to revise to use Amphibian.

- Revise your code until you can run the fishperf program and use it to drive your transport protocol. Note that this will probably cause your "transfer" command to cease working, which is fine. However, leave the single-letter printouts in the transport protocol for turn-in.

2 FishFone

In the second part of this assignment, your job is to design and implement a FishFone application that provides a walkie-talkie service between IPAQs with the following features:

- *Amphibian Faux Sockets.* Your application should use the amphibian "faux socket" interface you build in the first part of this assignment.

- *Transport Usage*. Your application should be able to both i) connect to a remote application to send audio samples over the network, and ii) accept connections from a remote application to receive audio samples over the network. To do this you should use the transport protocol you developed in the previous assignment. Your fishfone should accept connections on the FISH_SERVICE_FONE port as defined in amphibian_app.h. Each connection should only be used to send data in one direction, from the client initiating the call to the server accepting the call; you need to use multiple connections to have a two-way conversation.

- *Half-Duplex Audio.* Unfortunately, your IPAQ does not allow you to simultaneously play and record sound. To work around this, your application should support both operations, but only do one of them at a time. We suggest you detect whether a button is depressed on the IPAQ and use this as a user interface signal for switching between send and receive mode. That is, when a button is depressed, you should make establish a connection, take audio samples from the microphone on your IPAQ and continue sending them while the button remains depressed, and stop taking audio samples and close the connection when the button is released. Otherwise, when the button is not depressed, you should accept connections from other fishfones and receive samples from over the network, and send them to the speaker on your IPAQ while the connection lasts. We will supply you with code to help read and play audio samples and detect when buttons are pressed.

When you are finished, you should be able to call up your classmates on your fishfone and talk to them. To develop your application, you will probably want to tackle reading audio and playing audio separately. For example, try sending a known sounddfile and get playout working before you handle recording audio samples. Note that, unlike previous assignments, you cannot run and test your code in a private fishnet because the development environment does not have the same microphone and speaker devices as the IPAQs.

3 Discussion Questions

1. The fishfone as described above uses many transport connections. A simpler design would be to use only a single long-lived connection and send data across it in both directions. Describe the pros and cons of this design compared to the one above.

2. Describe which features of your transport protocol are a good fit to the fishfone application and which are not. Are the features that are not a good fit simply unnecessary, or are they problematic, and why? If problematic, how can we best deal with them?

3. Describe one way in which you would like to improve your design.

4 Turn In

1. Turn-in all of your source code for your entire node electronically.

2. Provide us with a live demonstration of your fishfone running on your IPAQ with your fishnet node and communicating with our fishfone.

—END—