Introduction to Fishnet

Tushar Jain

January 9, 2005

Parts of this document are taken from an earlier document describing Fishnet when it was implemented in Ruby.

Fishnet is a software project for teaching the core principles of network protocol design and implementation. This document describes our goals in designing Fishnet, its software architecture, and how it can be used to study network protocol design. Other documents describe the specific assignments in detail. Fishnet was originally developed by Neil Spring, David Wetherall, Sushant Jain, and Janet Davis. We have completely rewritten the system as of Winter 2005, converting the implementation from Ruby to Java. Java has the advantage of being strongly typed and very well documented. Also it is a language that most students in this department are familiar with since it is used in the intro courses.

1 Goals

The main goal in moving Fishnet from Ruby to Java was to allow students to work in a language they were already familiar with. Also development cycle is made easier in Java since a lot of errors and logic can be caught and enforced during compile time. It was noticed that in Ruby students would spend a lot of time debugging, just to catch a simple typo. Also there are some very good IDEs (like Eclipse) for Java that have good visual debugging features.

This transition was also made possible since we are not using the IPAQs this quarter, thus we do not have to worry about Fishnet being able to run on the IPAQs.

Most of the previous design has been maintained. Some changes were made in the implementation and class hierarchies.

Students should read the original Ruby document referred to earlier. It explains in detail what Fishnet was designed for and what its goals are.

2 Fishnet Programs

Fishnet This is the main class implementing the Fishnet programming model. Using command line arguments a simulation or emulation mode is started. This is the main code implementing the Fishnet programming model. In simulation mode (simulate), all nodes run in the same address space, and the simulator switches between them by scheduling and delivering packets between the various nodes. The command line arguments specify the number of nodes to simulate and the fishnet script file to use for interconnecting the nodes. In emulated mode (emulate), fishnet starts a single node that connects to the trawler (see below); the command line arguments specify where to find the trawler and what local port to use to communicate with peer nodes.

Trawler The Trawler manages a Fishnet emulation. A student can run their own Fishnet trawler; in this case, the student would run all of the nodes in the emulation. Note that the trawler and each of the emulated nodes should be started in their own window, to keep the input and output distinct. Once this is working, a student can connect to the shared, course-wide trawler; in this case, students create one or two emulated fishnet nodes (again, in their own window) and demonstrate that their solutions interoperate with the implementations provided by other students (and faculty). The trawler will run indefinitely waiting for nodes to connect to it; students must explicitly kill any trawlers and any emulated nodes that they start, or they too will run forever. A key aspect of the trawler is that you have to pick a port for the trawler to listen for incoming emulated node connections, and another port for every emulated node to listen for incoming packets from other emulated nodes. These ports can be arbitrary numbers between 1K and 64K, as long as they don't conflict with other ports that are in use on the same machine. In general, many of the ports less than 1024 are in active use by the host operating system, e.g., for receiving normal Internet packets for such services as email, so you want to pick port numbers higher than that. If you happen to pick a port that is already in use, you'll get back an error message and you can try again with a different number. Note that the trawler assigns each emulated node its own unique fishnet address. This is equivalent to the Internet' s DHCP protocol. In simulation mode, the simulator assigns each node a unique fishnet address.

topogen.rb This utility generates sample topology files under command line control. The generated topology files are in the format expected by the Fishnet Simulator and Trawler. Note that an emulated Fishnet node does not control its topology; the trawler specifies which nodes are neighbors.

3 Fishnet Files

The distribution has three groups of files. In the scripts/ directory you will find scripts and topology files for using with the Fishnet simulator and emulator. Files with .fish extensions are scripts for use with the simulator, and files with .topo extensions are topology files for use with the emulator. In the proj/ directory, you will find the files implementing the network protocol code that runs on Fishnet. These files are modifiable by students. In the lib/ directory, you will find the files implementing the Fishnet runtime environment supporting student network protocol code. *These files should NOT be modified by students.* In particular, we are likely to need to provide updated versions of these files to support later assignments, fix bugs, etc, and so any changes you make will need to be remade to other versions. Further, any changes to files in lib/ may prevent your code from interoperating with that of other student groups. However, students should read and understand the code in the lib/ directory, as it will be useful during debugging. It is ok to add temporary debug messages into these files, if that helps track down the source of a problem. In general, though, students should limit their changes to the files in proj/.

Classes in Proj

Node This is the code that students write to implement a protocol ona single node. This class uses methods defined in the Manager abstract class to talk to the network. The code in this class should not know whether it is running inside a simulation or as a seperate node in an emulation.

PingRequest A simple helper class to store information about pings needed by Node. You do not have to continue using this class. It is provided merely to implement the example ping functionality. You are free to reimplement that as you see fit.

Classes in Lib

Described here are the classes that the students need to be aware of in order to expand the functionality of the Node class.

Manager This is the abstract class that the Node uses to talk to the network and use to add timers.

Packet This class defines the headers for a packet that is sent across the network. It provides method to pack and unpack itself.

LinkState This class defines the headers for a Link State packet. It also knows how to pack and unpack itself. Students do not have to worry about this class till assignment 2.

Transport This class defines the headers for a Transport packet. It also knows how to pack and unpack itself. Students do not have to worry about this class till assignment 3.

Protocol This class defines the various protocols recognized by Fishnet.

Callback This class provides a way to create callbacks in Java. There is already an *addTimer* method in the Node class that shows how to use this class to create a callback.

Utility Provides some useful general purpose functions. Specifically there are two methods that convert Strings to byte arrays and vice-versa, using ASCI encoding. Students might find these helpful.

4 How To Compile And Run

There is a Makefile provided to simply the compilation step. This Makefile has only been tested with the version of make on attu. There is no guanrantee that it will work with make on different systems. Thus, the javac command to use is below:

```
javac lib/*.java proj/*.java
```

This assumes the command is run from the directory right above the lib/ and proj/ directories. This assumption is made for all commands described below.

There are two perl scripts provided as wrappers over the java calls to run Fishnet and Trawler.

fishnet.pl Usage: perl fishnet.pl [arguments to Fishnet]

You can run fishnet.pl without any arguments to get a description of the arguments that can be provided to Fishnet. Alternatively the java command to run Fishnet is:
java -cp lib/:proj/ Fishnet [args]
Note: In Windows replace the : with ;

trawler.pl Usage: perl trawler.pl [arguments to Trawler]

You can run trawler.pl without any arguments to get a description of the arguments that can be provided to Trawler. Alternatively the java command to run Trawler is:
java -cp lib/:proj/ Trawler [args]
Note: In Windows replace the : with ;

5 An Example: Ping

In order to test out the various parts of Fishnet, we implemented a simple "hello world" program on Fishnet. This code takes keyboard commands, and sends the text to the specified node (e.g., the command "5 Hi there" will send the text "Hi there" in a packet to node 5). The receiving node then copies the contents of the incoming packet into a response packet sent back to the source. This implements a version of the Internet's "ping" protocol used to check that a remote host is alive try man ping and see RFC 792 Internet Control Message Protocol The "ping " example illustrates the various entry points to send messages and establish wakeup calls, as well as the various upcall entries into student code to start a node, to receive a packet, to receive keyboard input, and to be woken up after a timer expires. Note that the node software is provided its fishnet address in Node::start; this allows your code to be generic whether the fishnet address is specified on the command line or provided dynamically.

6 Getting Started

Here is a quick guide to getting started:

1. Download the Fishnet source from the course website. The downloaded file should be named something like fishnet-1.1.tar.gz

2. Run tar -xzvf fishnet-1.1.tar.gz to unpack the Fishnet source

3. Change to the directory fishnet-1.1

4. Type make, to compile the code.

5. To run a simple example with ping in the simulator, type the following:

   ```
   perl fishnet.pl simulate 2 scripts/pingtest.fish
   ```

   This sets up a network with two nodes connected to each other. At time 10, the first node pings the second node.

6. To try out ping in the emulator, do the following (Linux and OS X only):

   - Open three new terminals.
   - In terminal 1 type,

     ```
     perl trawler.pl 8888 scripts/three.topo
     ```

   - In terminal 2, type

     ```
     perl fishnet.pl emulate localhost 8888 10000
     ```

     You should see a line like the following, which tells you that this node has the address 0:

     ```
     Node 0 started.
     ```

   - In terminal 3, type

     ```
     perl fishnet.pl emulate localhost 8888 10001
     0 Hello World!
     ```

   - You should see text in terminal 2 showing that a ping was received and a reply was sent. In terminal 3, you should see text showing that a ping was sent and a reply was received, ending with the following text:

     ```
     Ping reply from 0: "Hello world!"
     ```

   - Type exit in terminals 2 and 3 to quit the Fishnet nodes. In the first terminal, type ˆC to quit the trawler.

7. Read proj/Node.java to understand how the ping protocol is implemented. Then, get started on assignment 1. Have fun!