Program 2: L2 Protocol Implementation (revision 1)

CPE 464 Fall 2017

Due by 23:59:59, Friday October 20th.
Handin closes 23:59:59, Sunday October 29th.
This assignment is to be done individually.

Program Description

In this assignment you will be implementing a data link layer, including forwarding logic, encapsulation, and L3 to L2 address mapping. This assignment will use the `fishnet` overlay network to send and receive packets. Fishnet has already been introduced in lecture and is available for download from the course web site. The fishnet specification document describes all the protocols that form fishnet.

Base Functionality

Base functionality is implementing L2 encapsulation, forwarding logic, generating ARP requests, and responding to ARP requests. You do not need to implement any ARP caching logic or data structures. For base functionality provide your own implementations of the following functions:

- fish_l2.fishnode_l2_receive
- fish_l2.fish_l2_send
- fish_arp.arp_received
- fish_arp.send_arp_request

Full Functionality

Full functionality is implementing the ARP cache / data structure in addition to base functionality. This requires you to provide your own implementations for the following functions:

- fish_arp.add_arp_entry
- fish_arp.resolve_fnaddr

This is more complicated because it is possible for more than one packet to be sent while the node is waiting for resolution to complete. Each additional packet must **not** elicit another ARP. Instead, they should all be queued up and transmitted once the original ARP request has been satisfied (or failed).

Other Requirements

Your program must meet all requirements as specified in the course syllabus. In addition you are responsible for the following requirements:

1. Do not copy any code off the web or from other students. You may make use of the code I handed out for the first programming assignment and the code I handed out for this programming assignment.
2. Your fishnode must be able to send and deliver L2 broadcast packets to the built-in protocol implementations and the "applications". Certain applications, like `fishdump`, will not work correctly unless your L2 implementation correctly delivers the frames.
3. Your fishnode must use the same command line parameters as the provided example, goldentrout.
4. Your fishnode must work on unix4. Test your program on both of these architectures. Because a program compiled for Linux/Sparc will not run on Linux/Intel, your binaries must be given unique names on the different machines. Append "-<OS Name>-<Processor>" to the base name for each binary. The Makefile provided with programming assignment 1 demonstrates how to do this. Modify it to work correctly for this program. All your executables must be in the same directory.
5. Name your makefile: Makefile. Your makefile should provide a clean target that deletes all of the .o and executable files.
6. Your fishnode should never terminate, except as described below. It should loop waiting to process packets from the fishnet.
7. fishnode may exit in one of two ways:
   (a) Crash. If your program crashes it automatically fails the test case which caused it to crash.
   (b) Sent a `SIGINT` signal (ctrl-c). If fishnode is sent a SIGINT it must exit gracefully, and print the line:

   ```
   Fishnode exiting cleanly.
   ```

   before it exits. This will require you to catch and handle SIGINT.
8. You must set the function pointers to your own implementation(s) before joining a fishnet. Failing to do so will result in a zero on the submission.
9. You must name your source file fishnode.c or fishnode.cpp. You must name your header file fishnode.h. Create no other files.

What to turn in

Submit via `handin` to the `464_p2` directory of the `bellardo` account (syntax: `handin bellardo 464_p2 <file1> <file2> ...`):

- your source and header file(s).
- all necessary source and libraries to compile your code, including `fish.h` and `libfish.a`. **DO NOT** submit any of the fishnet executables (fishhead, fishdump, ping, etc).

- A makefile (called `Makefile`) that will build `fishnode-Linux-i686` on unix4 from your source when invoked with no target or with the appropriate "all" target . The makefile must also remove all binary and object files when invoked with the "`clean`" target. Refer to the example makefile provided with program 1 if you need more guidance on this. There is no constraint placed on the names of the object file(s), however I suggest giving them either a platform unique name or platform unique location, so you don't have problems when moving between different machines.
- A README file that contains:
  - Your name.
  - Any special instructions.
  - Any other thing you want me to know while I am grading it.

  The README file should be **plain text,** i.e, **not a Word document**, and should be named "README", all capitals with no extension.

Fishnet Documentation

Fishnet documentation is in doxygen comments (which are similar to javadoc) in the `fish.h` header file. The web page corresponding to the doxygen comments is available at `http://users.csc.calpoly.edu/~bellardo/fishnet/` . The web page is easier typically easier to follow than the actual comments.

Tips

- Your code will be unit tested. Your implementation of the required functions with be tested by repeatedly calling them with different parameters and verifying the output. Every one of your own functions that replaces a built-in function (via the function pointer mechanism) must behave *exactly* like the built-in function. It can't do any more or any less. Make sure you carefully read and understand what is expected of each function.
- Some timeouts can be rather finicky. For instance, if you register a timeout callback (assume 30s in the future) immediately *before* sending a packet and then retransmit during the timeout (presumably 30s later), it may actually be 29.999999s later due to the order in which you registered the callback. Pay careful attention to this. In some cases it may be necessary to add a small amount of padding (.1s, for example) to account for this. In other cases simply reordering your code suffices to fix the problem.

Important Fishnet Constants

The following important constants are defined in `fish.h`. Make sure you use them where appropriate in your fishnode.

- `ALL_L2_NEIGHBORS` – The fishnet broadcast address.
- `MTU` – The maximum possible size of a fishnet packet, including the fishnet header.

Fishnet Layer 2 Addresses

`fish.h` defines the type `fn_l2addr_t` for storing L2 addresses. In addition it defines two macros (`FNL2_EQ` and `FNL2_VALID`) for working with L2 addresses. The global constant `ALL_L2_NEIGHBORS` defines the L2 broadcast address. The function `fish_getl2address` returns the node's L2 address once it has joined the network.

For debugging purposes the function `fnl2_ntoa` will convert a `fn_l2addr_t` into a string. Do not call this function more than once within a single printf() call, otherwise the additional calls will produce incorrect results.

Other Important Fishnet Functions

The following functions will be of interest to you while you are implementing your fishnode. These functions are documented more fully in the `fish.h` header file. `fish.h` contains additional functions that are not necessary to implement a fishnode, but may be useful for debugging, etc.

- `fish_scheduleevent` – Schedule an "event" to take place sometime in the future. An event is simply a callback. This function allows you to register a function that `fish_main` will call `msec_delay` milliseconds in the future.
- `in_cksum` – The same `in_cksum` function you used in program 1. It is built in to libfish.
- `fish_debugframe` – Parses and prints a frame to standard error.

Sending, Receiving, and Processing Fishnet Packets

Refer to the documentation in `fish.h` for information on how to send, receive, and process fishnet packets.

Testing your Fishnode

We have provided you with a handful of programs to help you develop and test your own fishnode. These include: fishhead, goldentrout, fishdump, ping, and ocean.

Running your very own network

During development and testing you will find it very useful to run your own fishnet. Some of the provided programs make this possible.

Fishhead is the master program that controls the entire network. It is responsible for determining the topology and generally keeping things in order. There must be one fishhead running per network. Once the fishhead initializes it will print out the port it is running on:

```
pcp028651pcs:~/poly/464/fishnet/bin bellardo$ ./fishhead-Darwin-i386
fishhead started successfully on pcp028651pcs.wireless.calpoly.edu:56012
Terminate fishhead when done with ^C.
```

The complete list of command line options for fishhead are:

```
pcp028748pcs:~/poly/464/fishnet/release bellardo$ ./fishhead-Darwin-i386 -h
Usage: fishhead [options]
Run the Fishnet network manager.

options can include the following, where single character
options are a shorthand for the longer ones..

 -h, --help                print this message
 -p, --port <port>         run fishhead on the specified port
                           (default: 0)
 -i, --ip <IP or host name> run fishhead on the specified IP
 -d, --debug <debuglevel>  use the specified debug level
                           (default: 7)
 -l, --loss <loss rate>    add loss to all links.
                           (default: 0.000000)
 -t, --topology <topology> use a specified topology:
                           chain, clique, ring, pseudo-random, hierarchy
                           (default: pseudo-random)
```

The topology option allows you to specify the type of topology the fishhead creates. This is useful during early development because you can debug using a known topology.

Goldentrout is an implementation of a fishnode. You can start two instances of these connected to the fishhead:

```
pcp028651pcs:~/poly/464/fishnet bellardo$ ./goldentrout-Darwin-i386
                                                pcp028651pcs.wireless.calpoly.edu:56012
fishnode operating on Unix domain socket: F7FTIx
fish: joined network pcp028651pcs.wireless.calpoly.edu:56012 as F-0.0.0.1

dragonfly:~/poly/464/fishnet bellardo$ ./goldentrout-Darwin-i386
                                                pcp028651pcs.wireless.calpoly.edu:56012
fishnode operating on Unix domain socket: awysiK
fish: joined network pcp028651pcs.wireless.calpoly.edu:56012 as F-0.0.0.2
```

and then ping between them:

```
pcp028651pcs:~/poly/464/fishnet/bin bellardo$ ./ping-Darwin-i386 awysiK F-0.0.0.1
PING F-0.0.0.1: 26 bytes
26 bytes from F-0.0.0.1: echo_seq=1 ttl=63 time=0.597 ms
26 bytes from F-0.0.0.1: echo_seq=2 ttl=63 time=1.893 ms
26 bytes from F-0.0.0.1: echo_seq=3 ttl=63 time=2.404 ms
26 bytes from F-0.0.0.1: echo_seq=4 ttl=63 time=1.644 ms
26 bytes from F-0.0.0.1: echo_seq=5 ttl=63 time=2.007 ms
26 bytes from F-0.0.0.1: echo_seq=6 ttl=63 time=1.878 ms
26 bytes from F-0.0.0.1: echo_seq=7 ttl=63 time=1.046 ms
26 bytes from F-0.0.0.1: echo_seq=8 ttl=63 time=1.964 ms
26 bytes from F-0.0.0.1: echo_seq=9 ttl=63 time=1.929 ms
```

```
26 bytes from F-0.0.0.1: echo_seq=10 ttl=63 time=1.976 ms
26 bytes from F-0.0.0.1: echo_seq=11 ttl=63 time=2.008 ms
26 bytes from F-0.0.0.1: echo_seq=12 ttl=63 time=1.978 ms
26 bytes from F-0.0.0.1: echo_seq=13 ttl=63 time=1.884 ms
26 bytes from F-0.0.0.1: echo_seq=14 ttl=63 time=1.940 ms
26 bytes from F-0.0.0.1: echo_seq=15 ttl=63 time=1.901 ms
26 bytes from F-0.0.0.1: echo_seq=16 ttl=63 time=1.883 ms
26 bytes from F-0.0.0.1: echo_seq=17 ttl=63 time=1.943 ms
26 bytes from F-0.0.0.1: echo_seq=18 ttl=63 time=1.772 ms
26 bytes from F-0.0.0.1: echo_seq=19 ttl=63 time=2.300 ms
26 bytes from F-0.0.0.1: echo_seq=20 ttl=63 time=2.041 ms
26 bytes from F-0.0.0.1: echo_seq=21 ttl=63 time=1.438 ms
26 bytes from F-0.0.0.1: echo_seq=22 ttl=63 time=2.032 ms
^C
--- F-0.0.0.1 ping statistics ---
22 packets transmitted, 22 packets received, 0% packet loss
round-trip min/avg/max = 0.597/1.839/2.404 ms
pcp028651pcs:~/poly/464/fishnet/bin bellardo$
```

You can also connect to the fishhead using a web browser (the URL from the previous example would be `http://pcp028651pcs.wireless.calpoly.edu:56012/` to see a graphical layout of the network.

Collaboration

This is an individual assignment. Only collaboration as described in the syllabus for programming assignments is allowed. Refer to the syllabus for the details. Remember that all cheating and unauthorized collaboration will result in a F in this course and referral to the Judicial Affairs Office.

Example Output

There is no example output for this program.