Test the Simple Router

Disclaimer

The procedure described here and the associated script are provided for your information. This is what I plan to test your router, but I reserve the rights to test your routers in any other way without notice. You should make sure that your router implements all the required functionality correctly. Also note that the script is a simple Perl script to automate the tests. You can always test your routers manually by issuing ping, traceroute, wget etc. from command-line.

Files

The following files should present in your working directory:

- test1.pl, the test script
- Makefile and source code
- Topology files from your assignment email, e.g., vnltopoXX.iplist, where XX is the topology number.
- rtable, the routing table file, also part of your assignment email.

Tests

1. re-compile binary:

```
make clean;
make;
```

2. start the router:

```
./sr -t XX  -r rtable;
```

3. open another terminal, test ping, traceroute and web.

```
./test1.pl vnltopoXX.iplist ping all;

./test1.pl vnltopoXX.iplist tr all;

./test1.pl vnltopoXX.iplist web all;
```

The above tests ping/traceroute/web all the servers behind your router. A working router should be able to breeze through these tests.

In most cases the ping test should return 100%. If your router has persistent packet loss, there's something wrong with the router.

To manually run the web test, remember that the web server is running on port16280. So you can run a web browser from a CS machine, or use command-line tool such as lynx (which provides text-based interactive browsing), or wget (which downloads the web page) to access the URL:

```
http://ServerIP:16280/
```

Once you see the web page, you can follow the instructions to download other files or access other services on the server. In particular, I plan to test your router by downloading a 64MB file. You can try that on command-line as follows:

```
wget http://ServerIP:16280/64MB.bin -O /dev/null
```

With a correctly implemented router, this download should be done within a minute or two.

4. Test other behaviors.

4.1 Stop your router, then start it again with logging:

```
    ./sr -t XX -r rtable -l logfile;
```

4.2 Bring down the link between the router and server 1.

```
./vnltopoXX.sh vrhost setlossy eth1 100
./vnltopoXX.sh server1 setlossy eth0 100
```

The above commands set the loss rate of the two interfaces to be 100%.

4.3. in another terminal,

```
    ./test1.pl vnltopoXX.iplist unreach;
```

This will ping server 1 for 30 times, and then use wget to access the web port on the router.

The ping packets will cause unanswered ARP requests to server 1 because the link is down. Your router should time out after 5 ARP requests and return host unreachable message.

The web request will cause a TCP packet destined to the router. Your router should drop the packet and return port unreachable message.

4.4. stop the router, and bring back up the links.

```
./vnltopoXX.sh vrhost setlossy eth1 0
./vnltopoXX.sh server1 setlossy eth0 0
```

4.5. examine the log file:

```
    tcpdump -r logfile;
```

Look for correct behavior according to the required functionality: your router should send 5 ARP requests before timing it out and sending back a Host Unreachable message; not losing packets while waiting for the ARP reply; caching the Ethernet address of the default gateway (to internet) for 15s; responding with a Port Unreachable message to the TCP packet.