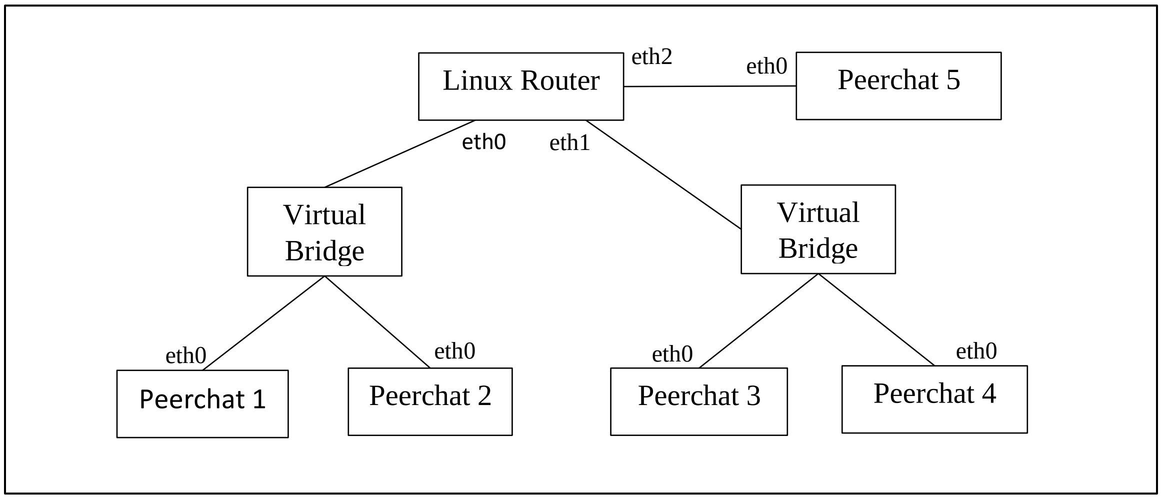
Assignment 4

Due: April 28th, 11:59 pm
Weight: 10% of total class grade

Overview

Convert Assignment 3 to utilize UDP multicast, retain the ability to forward to specific nodes, and provide a script that creates the networking configuration below using Linux Network Namespaces.



Configuration

The peerchat nodes have the following configuration:

Name	Address on eth0	Default Gateway	Listening	Forwarding
Peerchat 1	10.0.0.1/24	10.0.0.250	Port 10,000	10.0.1.1:10,000 10.0.2.1:10,000
Peerchat 2	10.0.0.2/24	10.0.0.250		
Peerchat 3	10.0.1.1/24	10.0.1.250	Port 10,000	10.0.0.1:10,000
Peerchat 4	10.0.1.2/24	10.0.1.250		
Peerchat 5	10.0.2.1/24	10.0.2.250	Port 10,000	10.0.0.1:10,000

The Linux router has the following configuration, with IP forwarding enabled:

Interface Name	IP Address
eth0	10.0.0.250/24
eth1	10.0.1.250/24
eth2	10.0.2.250/24

The Virtual Bridges can have any name (suggest “br0”, and “br1”), and the names of the Virtual Ethernet interfaces attached to the bridges can be anything (suggest “bpc1”, “bpc2”, “bpc3”, “bpc4”).

Starting Program

peerchat [-p <forwarding listen port>] [-f <forwarding client IP>:<port>]+ <username> <zip code> <age>

The optional port specifies the port number on which the “peerchat forwarder” will listen for peerchats forwarded to it from other peerchat forwarders. The optional forwarding client IP specifies a peer chat forwarding client to which peerchats should be forwarded. The”-f” option may be repeated up to 4 times. The username, zip code, and age represent information about the chat user. The developer should specify any necessary constraints on these fields, but should strive for flexibility and ease of use.

Example: peerchat –p 7001 –f 10.0.1.1:6001 –f 10.0.2.1:5003 PhineasFlynn 91501 10

Commands

```
/join [-p port] <multicast IP address>
/leave
/who
/zip <zip code>
/age <age>
/exit
```

Command Details

Join Command

This command joins the multicast IP address, and the optional port number, multicasting to other peers the username, zip code, and age of the user. It should also somehow receive information about all other active peerchat sessions (periodic multicasts, unicast replies to multicast, etc).

The output of the /join command on the joining session shall be one of:

[joined chat] -

On a successful join, all other sessions in the chat network will display the following:

[member joined: <username>@<ip address> <zip code> <age>]

Leave Command

This command disconnects the host from the chat network. The output of the /leave command on the departing host is:

[left chat]

On a leave, all other sessions remaining in the chat network will display the following:

[<username>@<ip address> left the chat]

Who Command

The who command shows all members on the chat network. A typical output might show:

/who

```
[PhineasFlynn@<10.0.0.1> 10 91501]
[PerryThePlatypus@<10.0.0.2> 5 99999]
[CandiceFlynn@<10.0.0.3> 15 51504]
[InternCarl@<10.0.0.4> 24 99999]
[DrDoofenschmirtz@<10.0.0.5> 42 96701]
```

Zip Command

The zip command shows all members on the chat network in the specified zip code. A typical output might show:

/zip 99999

```
[PerryThePlatypus 5 99999]
[InternCarl 24 99999]
```

Age Command

The age command shows all members on the chat network with the specified age. A typical output might show:

```
/age 5

[PerryThePlatypus 5 99999]
```

Exit Command

The exit command disconnects any active session (equivalent to a /leave), and then terminates the program.

Typed Text (Updated 2/23)

Non-command text entered on standard input (typed) will be echoed as normal in the local session. This text should also appear on the output of all peer members of the session, but the output should be prefaced by the username in angle brackets (<username>) and a single space. User "PhineasFlynn" typing "Where's Perry" would show the following in the local session:

```
Where's Perry
```

And would show the following on all peer members of the session:

```
<PhineasFlynn> Where's Perry
```

Forwarding

For peers acting as forwarding agents (to cross the broadcast domain), they will listen on a port specified with the "-p" option, and forward all peerchat text to the specified list of forwarding agents, specified with one or more "-f" options (one per each forwarding instance). Forwarding agents receiving such text should then forward it to the multicast address.

All multicast packets should be sent with a **TTL of 1**.

Script

The "setup_chats" script should launch 6 xterm windows, 5 to be used for PeerChat <x>, with a label of "Peerchat <x>". The sixth xterm will contain the Linux router. None of these should exist in the default Namespace, and the script should create and use separate namespaces for each of these xterms. The virtual bridges can exist wherever makes sense to the student.

Submission Details

A single zip file named "<rit_user_name>.zip" should be submitted to the assignment dropbox. The zip file should contain a single directory corresponding to your RIT username. Inside that directory should be C source file(s), headers, a script named "setup_chats", and a file "myname.txt" which contains the first & last name of the student, and a Makefile that will produce an executable named "peerchat". Hint – be sure to use tabs, not spaces in front of the makefile compile rules. For example:

```
bob1234.zip:
<bob1234>\
        peerchat.c
        peerchat.h
```

Makefile
setup_chats
myname.txt (contains the single line “Tony Dal Santo”)

Grading

Programs may be graded on style, comments, or entirely on the correctness of the output. Programs that crash, fail to produce the correct output, or don’t compile may lose up to 100% of the points, depending upon severity of the error. Some effort may be made during grading to correct errors, but students should not depend upon this. For grading, programs will be compiled and run on Ubuntu.

Resources

http://wiki.treck.com/Introduction_to_BSD_Sockets
http://www.cs.put.poznan.pl/wswitala/download/pdf/B2355-90136.pdf
https://mycourses.rit.edu/d2l/le/content/686309/viewContent/5004087/View
https://mycourses.rit.edu/d2l/le/content/686309/viewContent/5028873/View

The use of Wireshark is highly recommended. The gnu debugger “gdb” can be used to debug. The graphical front end “ddd” provides a friendlier user interface to gdb. Packages can be installed on Ubuntu via the command “apt-get install <package name>”. The command “netstat” is also useful to identify what UDP sockets are open and/or connected.

The following man pages will likely prove useful: ip(7), tcp(7), accept(2), connect(2), listen(2), select(2)