CSC/ECE573 Internet Protocol Project #2

Relay server

Due date : 11:45 PM Mar 19

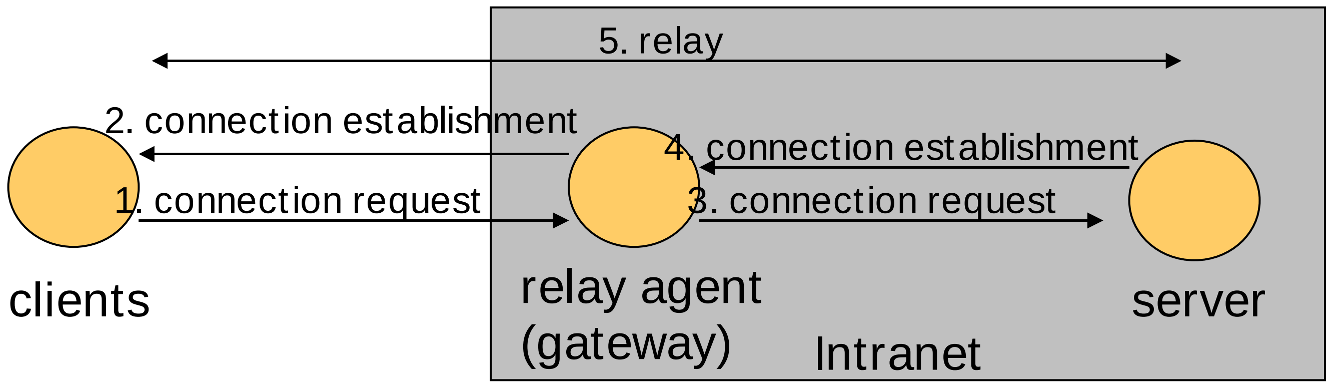
This assignment is an individual assignment. This project implements simplified TCP, UDP delay server operating as a kernel module as a preparation of final project. In this project, performance measurement of data packets relay will not be made, instead your module must guarantee "stability" issues.

Implementing TCP relay server

In this assignment, we build a simplified tcp relay server that relay packets between a server and clients to enhance the understanding of socket programming in kernel environment.

The basic operation of the TCP relay server is :

1. Listens for a TCP connection request from a client
2. Establishes a "client connection" with that client and makes a connection request to a server to which the connection is bound.
3. Forward the packets from the client to the server and forward the packets from the server to the client.
4. Closes the both peer connections in case of close request from one of the connections (graceful disconnection)



The transferred packets , both client to server and server to client, will flow through your module. TCP relay server must handle multiple , simultaneous connections with a minimum delay. Sender's packets delivered to the application layer are copied in application layer (TCP relay server) and transferred to receiver through the established channel.

Implementing UDP relay server (NAT on netfilter)

Your relay module will also relay UDP packets between clients and servers. To do this, register your translate functions in NF_IP_PRE_ROUTING , NF_IP_POST_ROUTING hooking points. Hooking functions probe the UDP packets which flows through the hooking points , and change the client (source) address to the relay server address. And then, the address pair should be registered (or managed in your module) to find the requester when the reply arrives in relay server. As UDP uses NAT , the packets are not delivered to the application layer.

Stability & service daemon requirements.

As Relay server provides network service in kernel module , it must fulfill several essential requirements.

1. Allocated resource s(Memory, socket , ..) should be released when they are not used any more.
2. Your OS must be completely free from the malfunction of your module.
3. Your service module should maintain the consistency.

Implementation & Test.

As in Project A, you will use C programming language on a linux OS (Kernel ver >= 2.4). Your web-browser, nslookup in client machine connects to the TCP relay server in a standard service port. Your clients should act in a same way when they connect to the httpd,DNS in a real server. To make virtual network environment, each relay server machine and real server machine should be installed as guest-OS in Vmware.

Notes

Before implementing in kernel level , thorough understanding of socket programming is required to the students who do not have enough experience. To do this, prototype development in user level will greatly enhance your understanding of what you do.

Simple preparation List.

1. Compose virtual network environment with Vmware or UML. You need at least two guest-OS running on two Vmware instances, each for relay server and real server machine.
2. Module service environment: Service start/stop , thread termination , /proc configuration , so forth
3. Kernel thread and thread synchronization or (multiplexing – select , poll)
4. Understanding of socket channel multiplexing.

References.

1. Project A references
   http://courses.ncsu.edu/csc573/lec/001/prog1.html

2. CSC573_2008Spring_Project_QA.ppt references

http://courses.ncsu.edu/csc573/lec/001/wrap/CSC573_2008Spring_Project_QA.ppt