College of Computer and Information Science Fall 2011
Northeastern University
CS4700 & CS5700: Fundamentals of Computer Networks

Problem Set 4 (due November 13, 2011 @ 11:59pm).

Important Notes:

1. Assignment to be done in teams of two.
2. Preferred programming languages C or Java. They grade will also be determined by the quality of the code/comments.
3. Late submissions will result in a 10% penalty per day (e.g., 2.5 days late result in 25% penalty).
4. You can use the Internet to get some help, but you should use your own words, examples, and code when answering the questions. Reference Internet materiel when appropriate.

Reverse SSH and NAT Traversal for Mobile Hosts

The goal of this assignment is to design and implement a solution to allow users to reverse SSH to their computers (e.g., laptops) even when they are left behind a NAT box.

Network Address Translation is a widely used mechanism to mitigate the scarcity of IPv4 addresses. It is commonly used in residential gateways (also called home routers, NAT boxes, or wireless access points). One of its positive side effects is that it partially hides the existence of home devices and prevents external scanning and access to such devices. It however limits users from remotely connecting to their home devices while they are outside the home network.

Question 1 [5 points]: Manual Setup – Consider a laptop running an ssh server. Briefly describe a simple and commonly used solution to allow users to remote access their laptop from anywhere in the Internet. The solution requires the configuration of the access points. Describe how this is done to be most convenient and what are the limitations. You don't need to implement this.

In the following, we want to design a system that allows reverse ssh to this laptop no matter where it is connected from.

Question 2 [5 points]: SSH Setup – Install or setup an SSH server on your laptop/home computer. Linux and MAC OS systems only need the SSH access to be enabled. For a Microsoft Windows system you can use cygwin or OpenSSH. Alternatively, you can also setup a minimal linux virtual machine using VirtualBox (a minimal debian installation can be less than 1GB in size). Provide a screenshot of how you SSH to your laptop.

Question 3 [35 points]: Basic Reverse SSH – Design and implement a client/server pair of programs with the following functionalities:

1. The client *CP1* runs on the laptop. It periodically (e.g., every 10 seconds) sends a UDP heartbeat message to a pre-defined UDP port on a remote server machine with a public IP address (e.g., login.ccs.neu.edu).

Example:

```
$ CP1 login.ccs.neu.edu portX
```

2. The server program *SP* runs on a server machine with a public IP address on the pre-defined port. Anytime the laptop changes IP address/port, the server responds with a command to the client requesting it to open an SSH tunnel between the laptop and the server machine. This should allow users to remote connect to their laptop by SSHing to the server on some port. To be able to do this part carefully read the man page for SSH (specifically the –R option). The details are left out for you to figure out how to achieve this.

   Example:

   ```
   $ SP portX portY
   $ ssh login.ccs.neu.edu –p portY
   ```

   Note that *portX* and *portY* should be replaced by specific numbers. *portX* corresponds to the port on which the server should be listening and *portY* the port on which the user should ssh to reach the laptop.

   Submit your programs in a folder named Q3. In your report submit screenshots of running the two programs.

   Can you remote connect to your laptop from any machine or do you need to be connected on the server? Why and if no what do you need to change in the config of your SSH server to allow remote connection from any third machine?

The solution you have implemented requires an SSH tunnel to always be present between the laptop and server which can be wasteful of resources. In the following, you will develop a system that creates the SSH tunnels on-demand.

Question 4 [35 points]: On-Demand Reverse SSH – Modify your design to allow a user to use an additional client program *CP2* running on the server to on-demand establish the SSH tunnels and enable the reverse SSH. You need three programs:

1. The client *CP1* running on the laptop does not have to be modified.

   Example:

   ```
   $ CP1 login.ccs.neu.edu portX
   ```

2. The server program *SP* should also expect requests coming from a third program running on the server requesting the establishment of the reverse SSH capability.

   Example:

   ```
   $ SP portX
   ```

3. Whenever the user wants to reverse SSH to its laptop, it runs the second client *CP2* (on the server machine). *CP2* sends a command (UDP packet) to the server program which then instructs the client program *CP1* to setup the SSH tunnel and prints the port number on which the user should SSH (on the server machine) to connect to the laptop (obviously you should make sure that the port is not already taken, etc.).

Example:
`$ CP2 login.ccs.neu.edu` *`portX`*

Submit your programs in a folder named Q4. In your report submit screenshots of running the three programs.

Find the largest value for the heartbeat period such that your home router (NAT box) does not forget the mapping that allows the reverse access to *CP1*.

Additional Required Problem for graduate students (CS5700).

Question 5 [20 points]: Multi-Device Support – Assume that you need to access multiple mobile devices that are sometimes behind NAT boxes. Assume that each device has a unique identifier known to the user. Extend your system to support multiple devices. Minimize the steps required for the user and simplify the system design.

Submit your programs in a folder named Q5. In your report submit screenshots of running the programs.