Fall 2020 CS120 Project 3. Gateway

Due Date: Nov. 29, 2020

(8 points + 11 points)

Suggested workload: 2~4 FULL days

Please read the following instructions carefully:

- This project is to be completed by each group **individually**.
- Submit your code through Blackboard. The submission is performed by one of the group members.
- Each group needs to submit the code **once and only once**. Immediately after TAs' checking.

Overview

The goal of this project is to build a gateway for the Athernet. In this way, Athernet devices are able to connect to the Internet. A network gateway may contain multiple network functions, including DHCP server, routing, switching and so on. This project focuses on the aspect of interconnecting different protocols, i.e., how to translate Athernet traffic to run on existing Internet infrastructure, and vice versa. After finishing this project, the TCP/IP traffic should be able to run over the Athernet network that you have built in Project2.

We consider the situation in Figure 1, where Athernet nodes connect to an Athernet Gateway, and then get access to the Internet (the Athernet Gateway gets access to the Internet through the WiFi Access point, which is the gateway for the Athernet Gateway. It's not uncommon for such hierarchical gateways in today's Internet).

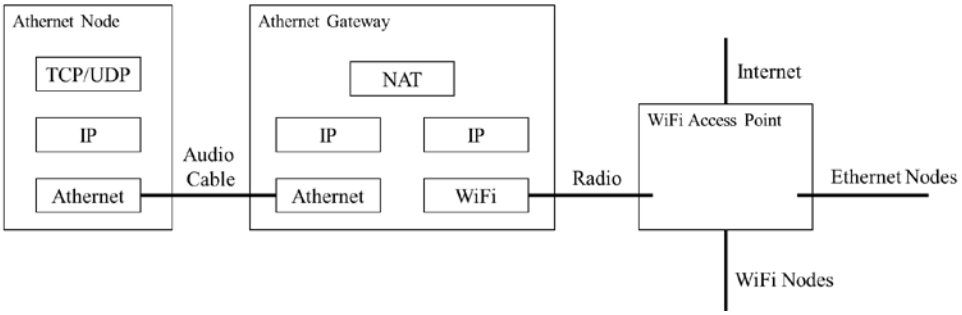

Figure 1 Project3 Overview

The functionality of the Athernet Gateway can be as simple as a bridge, i.e., decoding the Athernet payload and repacking it into WiFi packets, and vice versa, but the problem is that the Athernet Node must be able to handle WiFi authentication, DHCP and many subtle details, which may complicate our project.

Another design choice is shown in Figure1. On the one hand, the Athernet Gateway handles the WiFi connection with existing utilities from OS. On the other hand, it handles Athernet connections from Athernet Nodes. Note that in this situation, Athernet nodes are out of the domain of the WiFi network, and thus require a different IP subnet for addressing. Since we lack independent IP addresses, NAT is required in the Athernet Gateway to translate the Athernet traffic.

(Hierarchy of Tasks. Tasks are graded according to their hierarchy. A full score of one part

automatically guarantees the full score of its subparts. In this project, the hierarchy is denoted as Part1<Part2; Part1<Part3<Part5; Part4<Part5.)

Part 1. (2 points) Socket Programming

The first step to build a gateway is to manipulate the TCP/IP traffic. Although different operating systems handle network traffic differently, they provide a similar interface to send/receive TCP/IP packets: the Network Socket. Refer to the textbook for more information about socket programming in C/C++. You can also use Python wrapper for socket API. If you are using Java, be noted that native Java socket API lacks the ability to manipulate raw IP packets and thus you might need to either find third party Java packages for low level socket or integrate Java with other language to finish Part3.

To get you familiar with socket programming, this part is to set up a UDP server/client pair through socket API. The network topology is shown in Figure2. You can ignore NODE1 in this part. Make sure the WiFi network connection is successfully established in both nodes before programming. You can use system's network utility to connect NODE2 and NODE3 to the campus Wi-Fi network "ShanghaiTech". Use `ipconfig/ifconfig` command to find out the IP address of the node in the WiFi network. Also, make sure the two nodes can hear (`ping`) each other.

Socket API provides a convenient interface to send and receive UDP packets. Write a program to send a UDP packet with 20-byte random payload from NODE2 to NODE3 every 1 second. NODE3 should be able to display the received content.

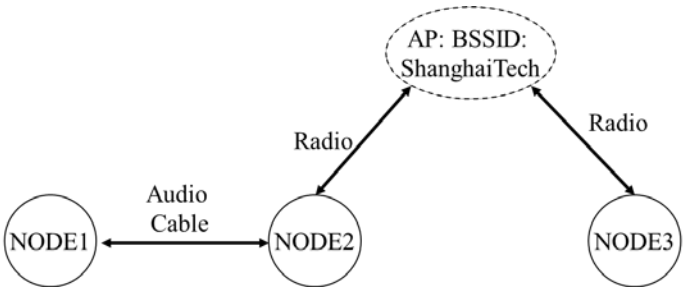

Figure 2 Network Topology

Checkpoints:
The group provides two devices: NODE2 and NODE3

CK (2 points).
NODE2: transmits UDP packets to NODE3 once per second for 10 seconds
NODE3: displays IP, ports and payload of the received UDP packets.

Tips:
a. Be careful about your binding IP address, especially when you have multiple network interface cards.
b. Be careful about your firewall settings. You may want to disable or add whitelist in the system firewall and firewall applications when necessary.

Part 2. (4 points) NAT

Network Address Translation (NAT) is a design choice to connect the Athernet to the Internet. In Figure 2. NODE2 has two network interfaces. One is the WiFi interface which is used to connect to the wireless LAN. The other consists of the audio card and the previous two projects you have finished. At a high level, the two network interfaces have very similar functionalities. From the operating system's viewpoint, however, they are different. Normally, the network traffic of the OS passes through the TCP/IP stack. The WiFi interface functioning below the TCP/IP stack is one of the standard I/O devices for TCP/IP segments. However, the I/O of Project 2 hasn't been hooked to the TCP/IP stack and thus the operating system cannot recognize the Athernet interface. Thus, existing NAT tools based on TCP/IP stack cannot be directly used to realize NAT in the Athernet Gateway.

Therefore, this part is to manually build NAT for the Athernet Gateway. Check more information about NAT on Lecture Slides. In order to talk with the Internet, the traffic over Athernet must "speak" TCP/IP, i.e., the payload of the Athernet MAC frame must contain a TCP/IP header. Specifically, if the protocol is UDP, the frame structure is shown in Figure 3. After receiving an Athernet frame like Figure 3, NODE2 is responsible for extracting <NODE1_IP, NODE1_port#> as the key to construct the NAT table. Also, NODE2 is responsible for repacking the UDP Payload of the Athernet frame to standard network socket with <NODE2_IP, NODE2_port#> and then sends it to the Internet through the WiFi interface. On the reverse direction, NODE2 should listen on the specific port# and translate the received packets to Athernet frames. After that, it transmits the frame to NODE1 via the acoustic link.

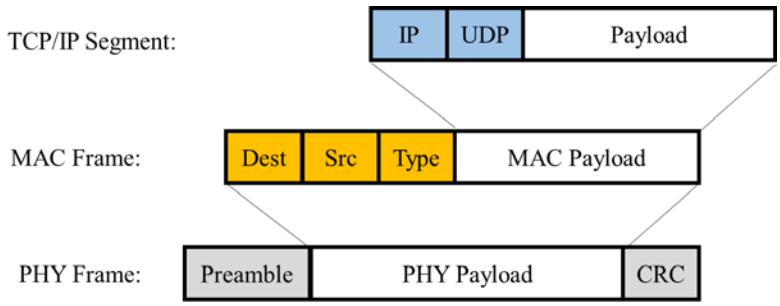

Figure 3 Frame Structure

Tips:

a. If you use different programming languages to implement the Athernet transceiver and the NAT table, the network packets between them can be shared through a temporary file.

Checkpoints:

The group provides three devices: NODE1, NODE2 and NODE3.

The network topology is shown in Figure 2.

The IP of the Athernet interface of NODE1 is 192.168.1.2, the default gateway is 192.168.1.1

The IP of the Athernet interface of NODE2 is 192.168.1.1

The TA provides a TXT file "INPUT.txt" which contains 30 lines. Each line is a message. The length of a message is less than 40 bytes (40 characters). A sample "INPUT.txt" with two messages is provided.

CK1(2 points).
NODE1: transmits messages in “INPUT.txt” to NODE3
NODE2: NAT
NODE3: displays IP, ports and payload of the received UDP packets.
The transmission must be correctly finished within 30 seconds.

<30s	-0%
>30s	-100%

CK2(2 points).
NODE1: displays IP, ports and payload of the received UDP packets.
NODE2: NAT
NODE3: transmits messages in “INPUT.txt” to NODE1

The transmission must be correctly finished within 30 seconds.

<30s	-0%
>30s	-100%

Part 3. (2 points) ICMP Echo

Internet Control Message Protocol (ICMP) is one of the most import protocols running over IP. As the name suggests, ICMP targets "control" rather than transmitting data. ICMP defines several useful messages to indicate and debug network errors. ICMP echo is the most basic one. ICMP echo defines a send and reply protocol. Whenever a receiver receives an `ICMP Echo Request` message, the receiver is responsible for replying an `ICMP Echo Reply` message to the request sender as soon as possible.

This task is to support ICMP echo in Athernet nodes, you may want to consider the following three aspects:

First, NODE1 should be able to send correct `ICMP Echo Request` to the Internet. In order to be able to receive replies from standard receivers, the ICMP packets must be in the correct format. Check the wiki [1] to see the ICMP format that is not covered in lectures. As `Ping` is based on ICMP echo, you can use `Wireshark` with icmp filter to capture `Ping` packets as the reference to create correct ICMP echo packets.

Second, NODE2 should be able to translate ICMP echo packets. As the ICMP is a protocol in IP layer (i.e., parallel to TCP/UDP), its packets have no port number, which is used as the translation key in the NAT table. However, NAT can use the identification field [1] in the ICMP payload as the identifier [2]. In other words, the <IP, ID> pair in ICMP packets has the same function as the <IP, port#> pair in TCP/UDP packets for the NAT.

Third, NODE2 should be able to send and receive ICMP echo packets to and from NODE3. ICMP echo packets, especially `ICMP Echo Reply`, are normally handled by the OS and are hidden to the normal socket. In order to capture the `ICMP Echo Reply` from NODE3, you may want to use the raw socket [4]. When forwarding `ICMP Echo Request` from NODE1, the raw socket is also a good choice to fully customize your packets.

Tips:
1. Some network interface cards do not support raw socket.

Checkpoints:
The group provides two devices: NODE1, NODE2
TAs provide NODE3 with Ubuntu
The network topology is shown in Figure 2.
The IP of the Athernet interface of NODE1 is 192.168.1.2, the default gateway is 192.168.1.1
The IP of the Athernet interface of NODE2 is 192.168.1.1

CK1(1 points).
NODE1: sends `ICMP Echo Request` to NODE 3 once per second for 10 seconds; at the same time, it displays IP, payload and latency of the received ICMP packets.
NODE2: NAT

CK2(1 points).
NODE1: sends `ICMP Echo Request` to 119.75.217.26 (www.baidu.com) once per second for 10 seconds; at the same time, it displays IP, payload and latency of the received ICMP packets.
NODE2: NAT

(Tasks with “Optional” tag are optional tasks. The instructor is responsible for checking and grading the optional tasks. Contact the instructor to check if you have finished one or more of them. In this project, Part 4 and Part 5 are checked by TAs.)

Part 4. (Optional + 2 points) Ping from External Network

Part 3 enables ICMP echo from the network behind NAT to the Internet. Normally, ICMP echo is not able to go through a NAT gateway to the internal network from the external network. However, the payload of the ICMP packets can be leveraged to piggyback information for NAT to indicate the internal destination [5]. Suppose the NAT gateway uses static mapping, i.e., <NODE1_IP, NODE1_port#, NODE2_IP, NODE2_port#> is static. If <NODE2IP, NODE2port#> is known to the public, e.g., NODE3. NODE3 can modify the `ICMP Echo Request` payload through `ping -p` option. Define the special ICMP payload and modify the NAT gateway to support ping from the external network.

Checkpoints:
The group provides two devices: NODE1, NODE2
TAs provide NODE3
The network topology is shown in Figure 2.
The IP of the Athernet interface of NODE1 is 192.168.1.2, the default gateway is 192.168.1.1
The IP of the Athernet interface of NODE2 is 192.168.1.1

CK(2 points).
NODE1: receives and replies ICMP packets
NODE2: NAT
NODE3: ping NODE2_IP -p XXXX
X is specified by the group.
NODE3 should be able to receive the ping reply from NODE1.

Part 5. (Optional + 4 points) ToyNet

The current Athernet is developed by each group independently, but a real network is about opening, sharing, and cooperating. This task is to reward the groups who are able to interconnect to each other. Figure 4 is the network topology of three connected groups.

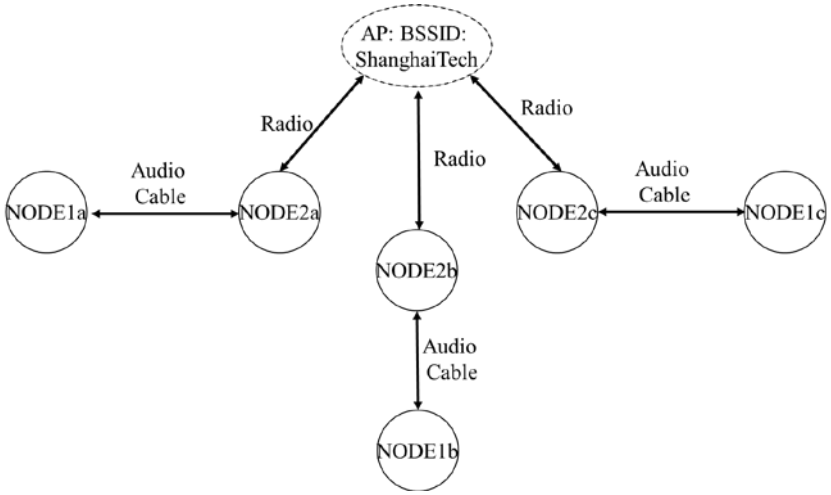

Figure 4 Network Topology for Three Connected Groups

Tips:

1. You can design your own ping utility, not necessarily rely on the ping from your system.

Checkpoints:

Each group provides two devices: NODE1x, NODE2x

The network topology for three connected groups is shown in Figure 4.

CK(4 points).

Each group should be able to ping other groups:

NODE1x: ping NODE2y_IP -p XXXX

The reward is proportional to the number of connected groups (can ping each other).

Connected Groups	
2	-50%
3	-25%
4	-0%

Part 6. (Optional + 5 points) TCP over Athernet

TCP is much more complex than UDP. TCP has flow control, sliding window, and congestion control components. Directly implementing TCP from scratch is not an easy task. The TCP/IP stack in current OS is implemented in the kernel, and it is also not easy to port to the Athernet. However, there are some open source TCP implantations in the user space. Refer to [4] for more information.

Checkpoints:
The check routine is similar to Part2. The network traffic is changed to the TCP protocol.

Reference and Useful Links

[1] ICMP Echo https://en.wikipedia.org/wiki/Ping_(networking_utility)
[2] NAT RFC https://tools.ietf.org/html/rfc3022
[3] Raw Socket https://tangentsoft.net/wskfaq/examples/rawping.html
[4] User Space TCP http://savannah.nongnu.org/projects/lwip/
[5] Ping through NAT https://stackoverflow.com/questions/4769587/is-it-possible-pinging-through-nat-from-outside-the-nat-inside