CSC358H5: Principles of Computer Networking — Winter 2025

Programming Assignment 2: Routing Packets in the Router

Due Date: Sunday, March 16, 11:59 PM

In PA1, you implemented a simplified networking stack for hosts. That implementation take care of encapsulating Datagram packets inside appropriate IP and Ethernet packets and perform necessary tasks such as sending ARP requests and replying back to ARP requests. In this assignment, you will implement a simplified routing that performs the fundamental task of a router: determining the next hop of packet in its journey to the destination host. This includes finding the best route for the packet from the routes installed in the routing table. This assignment is also based on one of Stanford's CS 144 lab assignments.

Collaboration

All programming assignments should be done individually. All codes that you use in your submissions should be written by you. Do NOT simply copy-and-paste code from websites such as Stack Overflow, GitHub, or even AI generating agents. It is OK to see the sample code snippets on such places, but you should write the code yourself and give the credit to them by citing the source that you used (*e.g.*, by mentioning the page URL).

You should not share your code with others or see other student codes. It is OK to discuss the assignment or potential solutions for them with another student. However, in addition to writing the whole code yourself, you should also mention this (the discussion with other students in finding the answer to the assignments) in the submissions.

You can also ask your questions on the course discussion board publicly, as long as you do not reveal your code. If there is a specific question related to a part of your code that needs revealing it, either ask it in a tutorial session or an office hour from a TA or via a private question on the discussion website.

1 Background

The Internet is created from a series of nodes that are connected to each other. A main design feature of this network is that there is no specific configuration defining how these different nodes are connected to each other. Therefore, one of the main tasks in the network is to find a path for each packet to deliver to the destination host. This problem is known as the routing problem, which we study during the semester in the class. Inside the network, specialized devices known as routers are used for this purpose. In this assignment, you will implement a simplified version of it. In other terms, you will implement a very simple router.

A router is usually located at the edge of a network connected to multiple networks. Its main job is to find the next hop for each packet that arrives at one of its ports. To do this, a series of rules (known as route entries) will be installed in it and it will search through them to find the best route for each packet.

The best route is defined as the route that has the best match with the packet's destination. This is usually referred to as "Longest prefix matching." What it means is that the router starts to compare destination address of the packet with installed routes. Each route consists of an address and a prefix length. The address is a 32-bit IP address and the prefix length is a number between 0 and 32. A route entry will match the destination IP address if the prefix most-significant bits of its matches the IP address in the route.

For example, if the prefix length is 8, it means that the 8 most-significant bits of the destination IP address (*i.e.*, the first byte of it) should matches the route IP address. So if the IP address is 123.0.0.0, then destination

IP addresses 123.123.123.123, 123.0.12.54, 123.255.255.255, and 123.100.200.50 will all match this rule, but destination IP addresses 124.0.0.0 or 8.0.0.0 will not match it.

The router will search through its routing table to find all matching routes for the destination IP address of the packet. If there is no such route, the packet will be dropped. If there are more than one such route, it will select the route with the longest prefix (hence the name longest prefix matching).

Therefore, you should be able to perform three tasks to route a packet in a router:

- **Installing a new route:** The router should have a routing table and be able to install new routes in it.
- **Finding the best matching routes:** The router should then search in the routing table and find all routes that match the destination IP address of a packet. If more than one route matches the packet, then select the one with the longest prefix.
- **Send the packet out of the router:** After finding the best route, it should send the packet to the next hop. This involves looking at the route rule to see what is the interface that this packet should be sent on. Furthermore, it should check to see if there is a specific next hop defined in the route. If the destination is not in a network that is directly connected to the router, then this router should send this packet to another router to continue routing the packet. This next router is usually referred to as the next hop. In this case, the next hop will also be defined in the route rule that is installed in the routing table.

2 Getting Started

This assignment depends on PA1 and you need a working solution for PA1 to be able to implement this assignment.

2.1 Starter code

You will continue to use the same starter code directory from pa0.

The following commands remain the same from pa1, the only difference is that the target now is "pa2".
Run the following command to create directory for compiling the code.

Linux Command: Configuring the build system and generating the build files in the "build" directory

```
cmake -S . -B build
```

Enter the build directory. You can build the source code by running the following command **from the build directory**:

Linux Command: Building the source code

```
cmake --build .
```

You are ready to start working on the source code to complete the assignment. Whenever you want to run build your solution and run tests on it, you can run:

Linux Command: PA1 automated tests

```
cmake --build . --target pa2
```

2.2 TODO List!

There are two functions inside NetworkInterface class (router.cc file) that you need to complete:

1. void Router::add_route(const uint32_t route_prefix,
   const uint8_t prefix_length,
   const optional<Address> next_hop,
   const size_t interface_num)

   This function is called to install a new route rule in the routing table of the router. This method just adds the route to the routing table. The actual routing will happen in another function. Each routing rule has four parts:

   - **prefix_length:** This is a number between 0 and 32 that defined how many most-significant bits of the packet's destination IP address should match the rule's IP address.
   - **route_prefix:** This is prefix IP address of the rule, which is a 32-bit IP number.
   - **interface_num:** If this route is the best matching route for a packet, then the packet should be sent on this specific interface number. You can get the actual network interface object that represents this in the class by calling the *interface(interface_num)* method of the *Router* class. Refer to the *Router* class definition in *router.hh* file for more information.
   - **next_hop:** This part is optional. What it means is that if the destination host is in the network that is connected to the router's interface number *interface_num*, then the packet can be directly sent to it on that interface. Otherwise, the packet should be sent to another router (as the next hop) that has IP address *next_hop*.

2. void Router::route()

   This is the function that will process and handle all packets that arrived at the router (through various interfaces) and are waiting to be route. It will process all such packets and send them appropriately. Here are the tasks that should be done in this function:

   - It should iterate through all network interfaces and process any packet that arrived at that interface and awaits processing.
   - For each packet that it found, it should find the best route. This means searching through the routing table to find the matching route with the longest prefix.
   - If there is no matching route for a packet, the router should drop the packet. This means that simply do nothing more and move on to the next packet.
   - Otherwise, the packet should be prepared for sending out. This means decrementing the TTL field. If the TTL was 0 before doing this or becomes 0 as a result of this, then the packet should be dropped as well.
   - If the TTL is higher than zero, then the packet should be sent out on the interface that is specified in the route. This will be done by calling the *send_datagram* of the outgoing network interface.

   In addition to completing these two functions, you can add any extra helper function or property to the Router class (*e.g.*, the routing table!) Therefore, you may need modify both *router.cc* and *router.hh* files. Our sample solution has less than 100 extra lines of code.

A Few Hints

Here are a few hints that maybe useful for you in writing:

- You do not need to worry about optimizing your routing table lookup. Although it is a very important aspect of designing routers (and includes designing special hardware to speed up such lookups), we will have relatively small routing tables in our test. The goal of this assignment is for you to understand and implement the main idea of longest prefix routing and not optimizing it.

- You may benefit from creating a "struct" for routes. You should do some reading to find out how to correctly create and initialize structs in c++. You will need to add a constructor to the struct that outlines the default values of it's members.
- You may need to convert the IP addresses between the *uint32_t* and Address format. You can use the specially defined functions inside the Address class for this (*Address::ipv4 numeric()* and *Address::from ipv4 numeric()* methods).
- You can use arithmetic shift to remove least significant bits that should not be matched from the destination IP address and the prefix IP address before comparing them together.
- Note the special case of prefix length 0 and the fact that you cannot shift a 32-bit integer by 32 bits.
- When you drop a packet, there is no need to generate any ICMP message. Although this is done in real life to inform the sender that a packet was dropped, you are not responsible to do it in this simplified assignment.
- Note again that you should have a working solution for PA1 (*i.e.*, *NetworkInterface* class) to be able to pass all tests. Note that we will NOT publish the solution to PA1. So you may want to run the tests for PA1 again before focusing on PA2 tests just to be sure. If you do not have a working solution for PA1, you can approach your TAs during the office hour and they will try to help you in-person. Otherwise, you can ask private questions on Piazza, so that your TAs can help you to finish PA1 part as well. However, the latter would be less efficient and may take more time.

Submitting The Assignment

You can access the MarkUs submission website of the course at `https://markus.teach.cs.toronto.edu/utm-2025-01/main/login_remote_auth`.

We prepared the submission assignment for Programming Assignment 2 there (PA2). You need to submit the four source code files for the *NetworkInterface* class (*network_interface.cc*, *network_interface.hh*, *router.cc*, and *router.hh*). You should also submit the writeups/pa2.md file that contains a few questions that you should answer about the assignment. Your submission will be graded only by the test that were given to you. A2 is graded with a total of 25 marks and it contributes 10% towards your course grade. The first test (*i.e.*, Test #1) is worth 1 mark and each of the remaining six tests (*i.e.*, Test #9-14) is worth 4 marks.

The writeup file is for us to receive feedback from you and to make sure that the submitted work is the student's own work. It's main functionality is to detect plagiarism and cheating would be reported the department and further actions will be taken by them.

Note that you are allowed to use up to a maximum of two grace tokens for this assignment.

Using Git

You are welcome to store your code in a private repository on GitHub, GitLab, Bitbucket, etc., but please make sure your code is not publicly accessible.