Programming Assignment 2: Routing Simulation

In this assignment, you'll learn how routers route traffic through the Internet by simulating a network of routers via socket programming.

Your task is to develop a small network of routers that route packets to each other via sockets. What follows is a high-level overview of the assignment. Each router will first create connections to neighboring routers based on a given topology, thereby creating the network of routers. One router will read an input file containing packets that need to be routed across this network. This router will then need to send each packet to the appropriate next hop router which is determined by its forwarding table. The next hop router will receive the packet and send it off to its appropriate next hop. This process will continue until each packet reaches its final hop router.

There will be 3 inputs for this assignment:

1. A CSV file that contains packets that need to be routed.
2. A CSV file for each router that contains that router's forwarding table.
3. The router network topology diagram.

<u>A CSV file that contains packets that need to be routed (*packets.csv):*</u>
Each line in this CSV file represents a simplified version of a packet, and includes Network Layer (Layer 3) information, payload, and the TTL field. Specifically, the 4 fields for each packet are source IP, destination IP, payload, and TTL. Each field is delimited by a comma. For example:

173.121.170.136,10.2.4.82,Hello,7

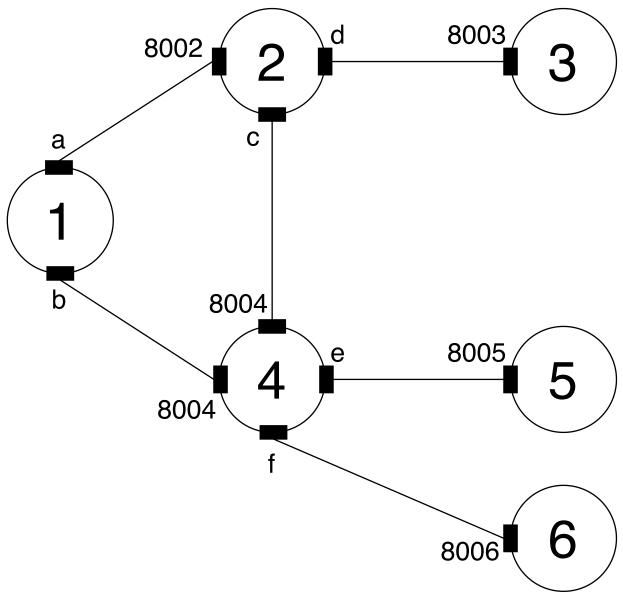
- where the source IP is 173.121.170
- the destination IP is 10.2.4.82
- the payload is Hello
- and the TTL is 7

<u>A CSV file for each router that contains that router's forwarding table (*router_#_table.csv*):</u>
Each router will have its own forwarding table. Each table will include 4 fields: network destination, netmask, gateway, and interface. Each field is delimited by a comma. For example:

10.0.0.33,255.255.255.224,127.0.0.1,8003

- where the network destination is 10.0.0.33
- the netmask is 255.255.255.224
- the gateway is 127.0.0.1
- and the interface is 8003

<u>The router network topology diagram:</u>
The diagram below shows each router (1, 2, 3, 4, 5, 6), and its interface(s).



In this assignment, an interface is simulated using ports – for example, if Router #4 wants to communicate with Router #5, it would create a socket and connect the socket to IP address 127.0.0.1 and port 8005 (destination port). Before this connection can be established, Router #5 needs to have created a socket, bound the socket to IP address 127.0.0.1 and port 8005, and be listening. Because the source ports are selected by the operating system, we substitute variables (a, b, c, d, e, and f) for these ports.

You can manually set up the connections from the diagram above. There are a few things to keep in mind.

- First, Router #1 is responsible for reading in the packets and beginning the routing process.
- Second, each router is a separate program on the same machine (127.0.0.1).
- Third, since we are using sockets to connect the routers, it is up to you to decide which router(s) should act as clients, which router(s) should act as servers, and which router(s) should act as both (hint: study the forwarding tables before you start writing any code and decide which way the packets will flow starting from Router #1).

<u>Abstractions to keep in mind:</u>
To reiterate, there are a few abstractions going on in this assignment (don't worry if the following is confusing – as you begin working on this assignment, these abstractions should become a clearer):

- Each interface represents a port that the router's socket will bind to (for this assignment, software ports simulate network interfaces).
- In the forwarding tables, each gateway is 127.0.0.1 since you are running each router instance on your local machine.

- The network destination IP addresses in the forwarding table are the simulated IP addresses of the routers – the actual IP address of each router is 127.0.0.1 (localhost).
- Each interface in practice would correspond to a unique IP address (think of these routers being connected by an ethernet cable – you can't plug two different cables into the same physical port). However, because we are using software ports to simulate network interfaces, and since a socket can listen to multiple connections on the same port, a router can use the same port to represent two different interfaces. The only example of this happening in this assignment is on Router #4's port 8004.

Instructions and Hints:

1. Study the input files carefully. Make sure you understand the purpose of each field. Understanding the forwarding table can be especially difficult initially – Read chapter 4 of the textbook and look for helpful resources online (e.g., https://en.wikipedia.org/wiki/Routing_table#Forwarding_table and https://www.youtube.com/watch?v=rYodcvhh7b8)

2. Set up the connections. First figure out which routers need to act as clients, servers, or both. Then create the sockets and bind the sockets to the correct ports (as seen in the diagram). For servers, enable them to accept connections by listening. Once a connection is established, **create a new thread for handling the message (receiving, processing, and forwarding).** Multithreading allows a server to communicate with more than one client simultaneously.

3. Read in and store the forwarding tables. Each router should be able to parse its forwarding table to determine over which socket to send any given message it receives. It's up to you what data structure you would like to use to store this information.

4. Router 1 should read in the packets. Router #1 is responsible for parsing the packets CSV file and sending each packet to its appropriate next hop. Router #1 should send one packet every second. This way, we can get a better view of the entire routing process.

5. Each router should be able to perform the routing process. Several steps need to be performed in the routing process:
(a) Each router should be able to receive packets. This should be taken care of in step 2. Once a router receives a packet, append that packet to a file called **received_by_router_#.txt**, where # is the router number (1, 2, 3, 4, 5, or 6).
(b) When a router receives a packet, it should first parse the packet for the destination IP and TTL. It should then check its forwarding table to see which socket that packet should be sent over. In order to do so, the router will need to check each row in its forwarding table to find a match between the packet's destination IP and the combination of the network destination and netmask fields. *If a match is found on a particular row, the socket over which the packet should be sent will be the one that is bounded to the port that matches the row's interface field*. If no match is found, then the router will send this packet out on its default gateway's interface (the default gateway's interface is represented by the interface tied to the network IP 0.0.0.0). If a match is found, but the interface is listed as 127.0.0.1 in the forwarding table's interface field,

then it means that this router is the final hop router (no next hop). If so, the router should append the payload ONLY to a file called **out_router_#.txt**.
(c) Before sending a packet to its next hop, the router should decrement the TTL value by 1 (including Router #1). If the TTL value becomes 0, then the packet should NOT be forwarded, but instead discarded. If so, append the packet (with a TTL of 0) to a file called **discarded_by_router_#.txt**. If the TTL value is greater than 0, then the router will first append the packet (with the new TTL) plus the router that it is sending this packet to (e.g., out_file.write(packet_to_write + " " + "to Router " + send_to_router + "\n")) in a file called **sent_by_router_#.txt**, and then send the packet out over the appropriate socket.

* While you can use any language you want, Python is highly recommended because of its ease of use when it comes to socket programming and threading.
* You are only allowed to use the Python libraries that are already imported in the skeleton code (and only equivalent libraries if you decide to use another language).

<u>How to run the network:</u>
The easiest way to run the network is to first open 6 separate terminal windows. Given the network topology diagram and how connections between the routers should be established, then run the router programs in reverse order, each in a separate terminal (i.e., first run router6.py, then router5.py, then router4.py, etc.).

<u>What to turn in:</u>
1. Code for each router (router#.py): total of 6 files – one for each router
2. Output files (received_by_router_#.txt, discarded_by_router_#.txt, sent_by_router_#.txt, and out_router_#.txt): there will be a total of 17 files if the assignment is coded correctly – note, some routers may not produce all types of output files.