

CSE123 Computer Networks
Spring 2021 Project 3
DESIGNING A SIMPLE ROUTER

Project 3 TA - Aritra Basu

Acknowledgement: Roger Chen

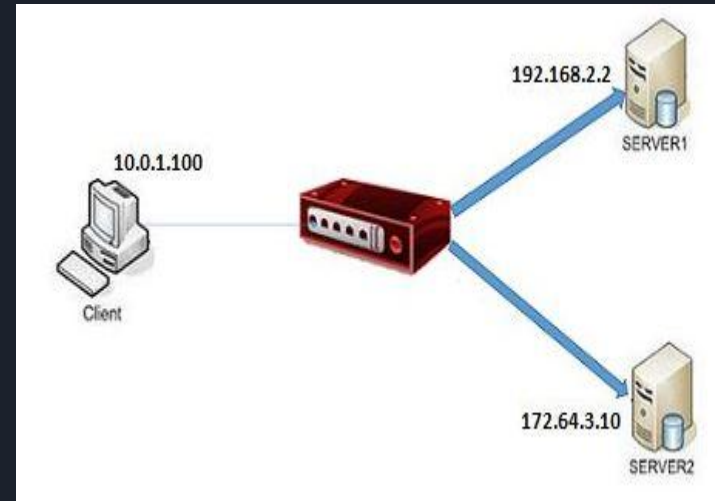


Project Setup

- Get the invitation to access the skeleton code.
- The link will also be shared on a Piazza post and a Google Doc with the project description.
- Once you accept the invitation, you will get a private repo.
- Clone the repo and do the project yourself - this is an individual activity.

Overview

- In this assignment you will write a simple router for a given static network topology and routing table.
- Your router will receive raw Ethernet frames and needs to process these packets by forwarding these packets to the correct outgoing interface.
- You are responsible for implementing the logic for handling the incoming Ethernet frames.
- A VM will also be provided to you with all the suitable environment settings.





Mininet

- This assignment runs on top of Mininet, which was developed at Stanford.
- Mininet is a network emulation tool which allows you to emulate a network topology on a single Linux kernel by providing the necessary isolation between the emulated hosts and routers.
- Mininet uses lightweight virtualization to make a single system look like a complete network, running on the same kernel.
- You don't have to know how Mininet works to complete this assignment, but more information about Mininet (if you're curious) is available [here](#).



Getting Started

- Follow the instructions provided in the Google Doc to install VirtualBox on your system.
- Download the virtual machine image ubuntu1404.ova following the descriptions in the GDoc.
- Configure VirtualBox as explained in the instructions and boot the VM image.
- Run the VM and login using username “mininet” and password “mininet”.
- After logging in, remember to run the command “sudo dhclient eth1” to get an IP address on the host-only network.



Getting Started

- You might find it more convenient to ssh into your VM instead of working inside of it.
- To do this, run “ifconfig”.
- Look for the interface that has an IP address that starts with “192” and note it down.
- Logout of your vm by typing “exit”. This should take you back to the login screen.
- Minimize your VM window. Do not shutdown the machine.
- In a terminal, run “ssh -X mininet@<the ip address you wrote down>”



Getting Started

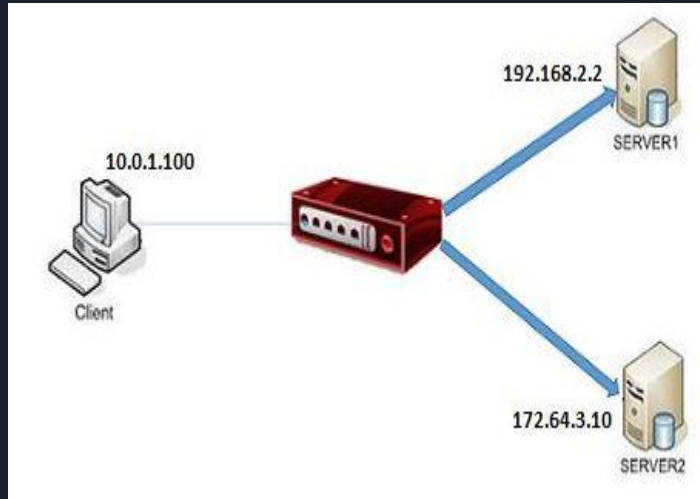
- You will need 3 terminals for this project, one each for the following (please follow the instructions in the Google Doc)
 - Mininet
 - POX controller
 - Running your code
- You can use [screen](#) to run multiple shell sessions on the same window. For a cheat sheet of the keyboard shortcuts available, consult the link or use manpages.
- All the coding will be done in C.



Getting Started

- In the third terminal, “cd” into the starter code directory.
- Execute “make clean”, followed by “make”.
- ~/cse123_p2/sr_solution is a reference solution for you to see what the expected behavior is.
NOTE: This is not the exact solution for project 3, but you can at least use it to test if your setup is working fine. We will make a Piazza post when we share the solution binary of project 3.
- Execute ping commands from the first terminal where mininet is running (please follow the instructions in the Google Doc). You will be able to view the expected output.
- Next, run the starter code from ~/cse123_p2/sr (please follow the instructions in the Google Doc). Since the skeleton code cannot handle ping commands, you will get unexpected outputs. Now, you are ready to start coding!

Topology



- The topology is setup using two files:
 - IP_CONFIG
 - rtable

- `> cat ~/cse123-p2/IP_CONFIG`

```
server1 192.168.2.2
server2 172.64.3.10
client 10.0.1.100
sw0-eth1 192.168.2.1
sw0-eth2 172.64.3.1
sw0-eth3 10.0.1.1
```

- `> cat ~/cse123-p2/rtable`

```
192.168.2.2 192.168.2.2 255.255.255.255 eth1
172.64.3.10 172.64.3.10 255.255.255.255 eth2
10.0.1.100 10.0.1.100 255.255.255.255 eth3
```



What should your router support?

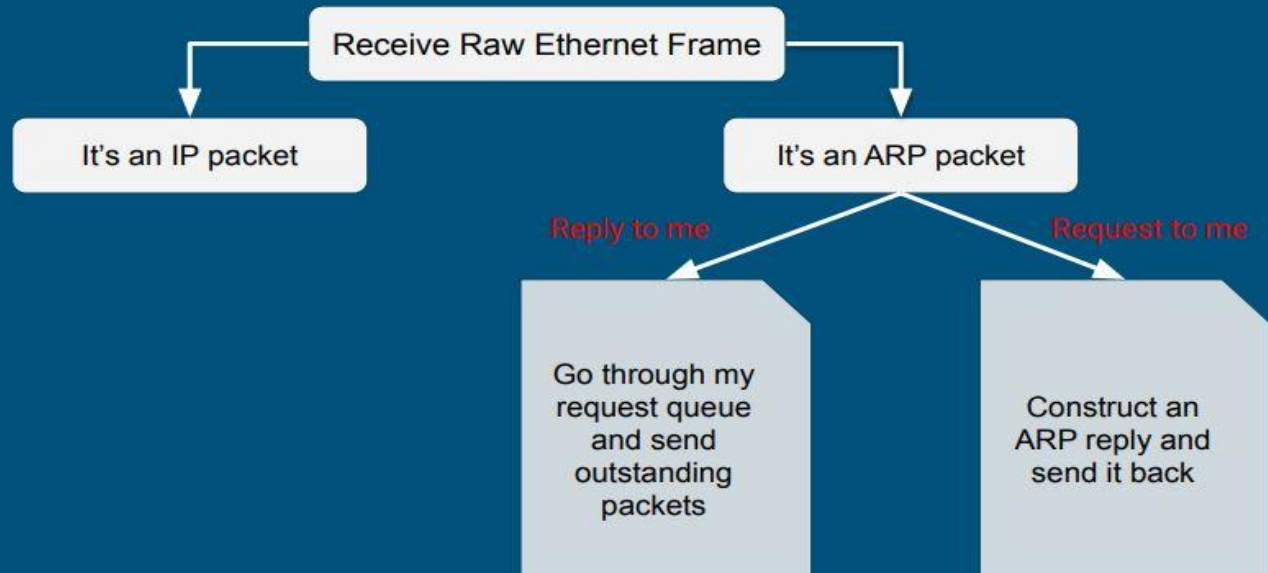
- Route Ethernet frames between the client (10.0.1.100) and the two HTTP servers (192.168.2.2 & 172.64.3.10).
- Your router should be able to handle two types of packets:
 - ARP packets: requests & replies
 - IP Packets: ICMP & TCP/UDP
- Where do you start coding?
 - `sr_handlepacket()` method in the `sr_router.c` file



How do you differentiate packets?

- “ether_type” to your rescue!
- Check the packet received in the `sr_handlepacket()` function using “sr_ethertype” defined in the `sr_utils.c` file
- You can find these definitions in the `sr_protocol.h` file.

ARP FLOW

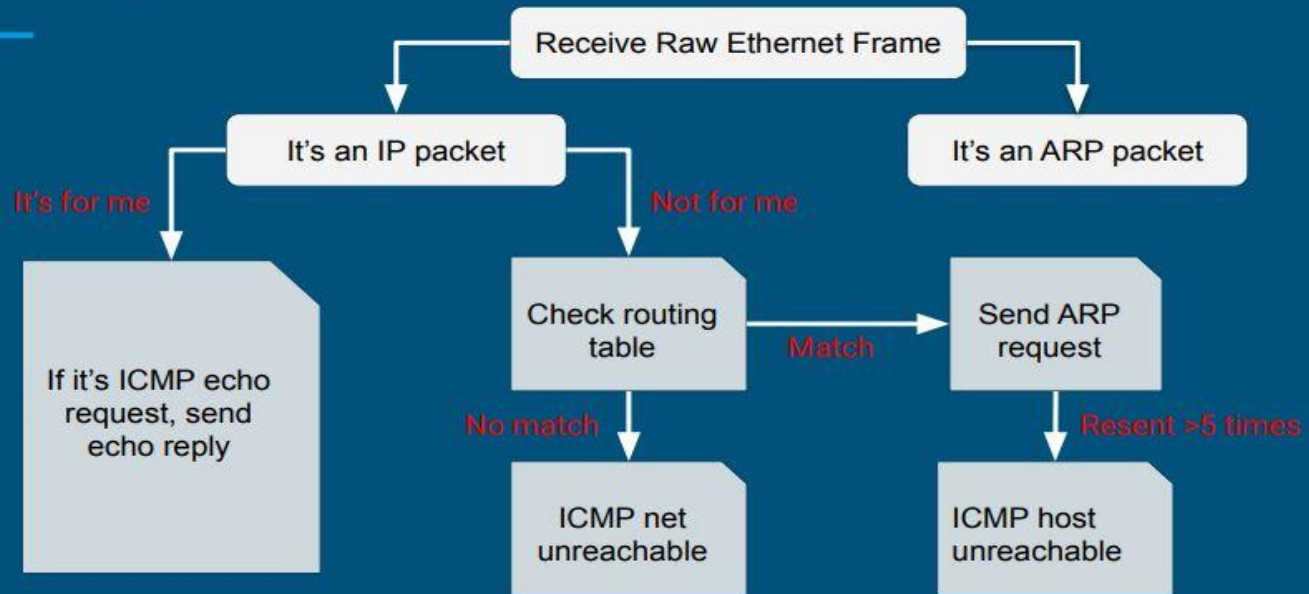




If it is an ARP packet

- Verify the length of the packet.
- Use “sr_get_interface” (defined in the sr_if.c file) to get the interface record.
- Get the MAC address of the destination IP address / interface IP address from the record.
- Check the “opcode” variable of the ARP header and see if it is and “arp_op_request” or an “arp_op_reply”.
- If it is an ARP Request, update all the fields of the packet and use “sr_send_packet()” to send an ARP reply.
- If it is an ARP Reply, send all the packets in the queue to the destination.

IP FLOW





If it is an IP packet

- **Sanity check the packet by validating the IP header:**
 - **Should not be IPv6.**
 - **Check the IP header length - “ip_hl”.**
 - **Check IP packet length - “ip_len”.**
 - **Check the checksum.**



If it is an IP packet (destined to you)

- Check the IP protocol - “ip_p”.
- If it is ICMP echo request (type 8), then generate ICMP echo reply (type 0).
- Otherwise, ignore the packet.
- NOTE: The data field of an ICMP echo request does not have a fixed length. Its length is determined by the total length field of the IP header. The router should copy the complete data field from an echo request to the corresponding echo reply.



If it is an IP packet (not destined to you)

- Check TTL value - “ip_ttl”. If $TTL \leq 1$, send ICMP time exceeded (type 11, code 0).
- Look up the next-hop address by doing exact comparisons to the IP addresses in the routing table using the packet’s destination address (do not perform longest-prefix matching).
- If an entry does not exist, send ICMP net unreachable (type 3, code 0).
- Otherwise, decrement TTL by 1 and recompute the packet checksum over the modified header.
- Send ARP request to determine MAC address.
- Encapsulate IP datagram in Ethernet packet.
- Forward packet to outgoing interface.
- Do not store the response in an ARP cache for this project.



Tips

- Take advantage of the print functions available in the “sr_utils.c” file for printing out the network header information from your packets.
- Don't get mixed up with endianness - Linux is little endian while the network is big endian.
- Make sure to push your code to your Git repo frequently as your VM image can get corrupted.
- Set milestones so that you can keep track of how much work is pending.



Tips

- In the file “`sr_router.h`”, struct “`sr_instance`” defines the context of the router.
- In the file “`sr_router.c`”, “`sr_handlepacket()`” is called for every packet that goes through the router (this is where you have to add most of your code).
- In the file “`sr_protocol.h`”, you will find all the convenient structs for accessing fields in packets.
- In the file “`sr_if.h`”, you will find all the methods for getting information about the router’s interfaces.
- The file “`sr_utils.c`” has quite a few utility functions, such as `cksum()`, `ethertype()`, `ip_protocol()`, etc, that you will find to be very useful.



Tips

- What to do when “Routing table not consistent with hardware” appear?
 - Restart mininet, and run `./sr` or `./sr_solution`



Tips

- **What are the Endianness for Networks Protocols vs Linux?**
 - **Linux uses Little Endian.**
 - **Network Protocols use Big Endian.**
 - **Convert the endianness using `ntohs()` / `ntohl()` / `htons()` / `htonl()`.**