An Pham
9477698118

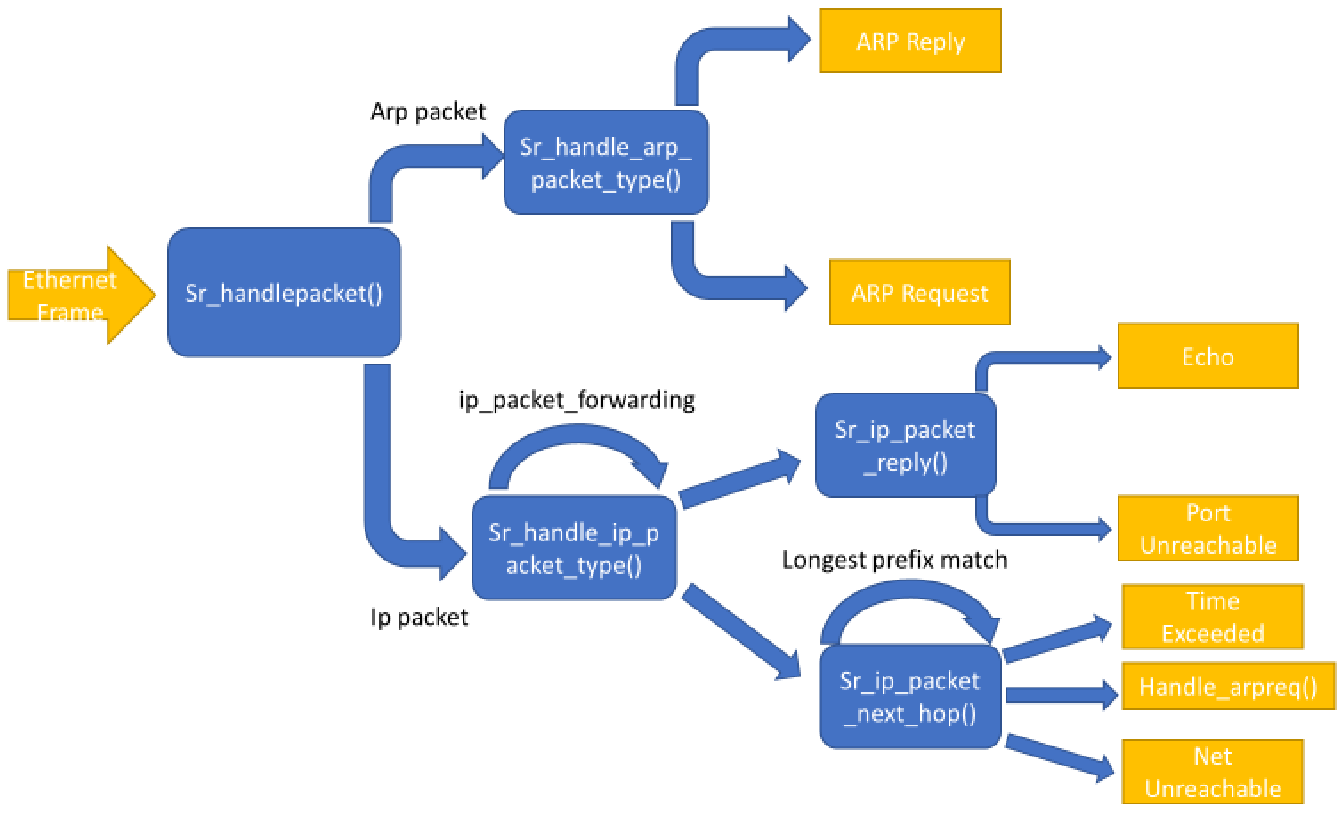
CSCI 551 Lab 1 Report

Overview: The various files within this Lab implements the functionality and protocols that would be found in an everyday internet router. These protocols include: Ethernet, IP, ICMP, ARP. In a nutshell, our router receives and ethernet frame as byte data, parses the data to determine where to route the packet, and finally repackets the byte stream with the correct ip/MAC address so that the packet would reach the correct destination.

Environment:

- Ubuntu Linux Virtual Box: Environment to test our code
- Mininet: To set up our network topology
- POX: SDN Controller, allows router to communicate with our code
- Visual Studio Code:
    - Text editor with various functionalities, providing ease of development
    - Ssh-remote: to avoid using VM's GUI

Program Flow:



Changed Files:

Sr_router.h: Overall added function definitions & added MACROs that would be used in sr_router.c
Sr_router.c: Implemented the logic and code flow for both arp packets and ip packets.

- **sr_handlepacket():** Entry point of the Ethernet frame. From here the code determines whether we've received an arp packet or an ip packet. It then forwards the function parameters to either sr_handle_arp_packet_type or sr_handle_ip_packet_type.
- **sr_handle_arp_packet_type():** This function determines whether the arp packet operation is to request or reply.
- **sr_handle_ip_packet_type():** This function determines if the ip packet is meant for someone else or if it's in the router's routing table
- **sr_ip_packet_reply():** This function is called if the ip packet was destination was meant for one of the router's interfaces. It checks if the packet is valid (check sum) then it either sends and echo or a port unreachable message.
- **sr_ip_packet_next_hop():** This function is called if the ip packet doesn't match the interface list and needs to be forwarded. If the destination ip does not exist in any of the subnet, then it returns a net unreachable message. If the TTL 0 after being decremented, it sends a time exceeded error code. After passing both test, it looks within the arp cache to see if it's there. If it exists, it sends the packet to the lpm's interface else we call handle_arpreq().
- **ip_packet_forwarding():** This is a helper function that loops through the interface list, looking for a matching destination ip that was found in the receiving ethernet packet. It either returns the interface with the matching ip or returns null
- **lpm():** This is a helper function that loops the routing table entries. It does 2 checks. The first check determines if the destination ip exists within the subset, while the second check is to keep track of the longest prefix match.
- **send_icmp_echo_packet():** This is a helper function that builds an icmp echo packet and sends it.
- **send_icmp_error_packet():** This is a helper function that builds an icmp error packet and sends it.

Sr_arpcache.h: Contains the function prototype, constants, and structure definition related to arpcache
Sr_arpcache.c: Implemented the logic and code flow for sweeping through arp cache and handling them based on number of request and time.

- **sr_arpcache_sweepreqs():** Loops through the arpache's list of arp requests. Before forwarding the current request to handle_arpreq(), we save the next request incase it gets destroyed.

- **handle_arpreq():** Takes care of two cases: either sends a broadcast arp request message or sends an icmp_error_packet and destroys it.

Limitation: Our design and implementation follow very closely to the lab manual's description. Future development should provide more generality and support for other ip protocol or icmp types.

Result: Passed every test case besides "tracerouteip"

Error Message:

```
INFO:root:*** Stopping tcpdump at client
*** client : ('pkill -9 -f tcpdump',)
INFO:root:*** Checking if traceroute matches 10.0.1.1
INFO:root:*** Found traceroute hop 192.168.2.1
INFO:root:*** Found path 192.168.2.1
```