SIMPLE ROUTER – PROJECT 2

- Balachander Padmanabha
- TA – CSE 123 (FALL 2017)
- OH (Wed 9-11am B240A)
MININET

- It is a network emulation orchestration system which runs a collection of end-hosts, switches, routers, and links on a single Linux kernel. It uses lightweight virtualization to make a single system look like a complete network, running the same kernel, system, and user code.

- Please use “#mininet> help” command or refer the documentation online at “http://mininet.org/walkthrough/” for more information.
3,2,1.. GETTING STARTED

- Boot the VM image provided on Vmware fusion and use **mininet/mininet** (username/password).

- You need 3 terminals, One each for **mininet**, **pox controller** and another one for running the code. Use **screen** or **byobu**. Refer online for how to use these commands.

- In the third terminal, “cd” into the router directory, execute “**make clean**” -> “**make**” -> **./sr**

- ~/cse123_p2/sr_solution is a reference solution for you to see what the expected behavior is.

- Execute ping/traceroute commands from first terminal where mininet is running.
TOPOLOGY

• Github repository has been setup with skeleton code for this project and a VM will be provided with suitable environment settings.

• Coding will done in C (all hail pointers!)

• The topology is setup using IP_CONFIG and rtable files. Please check them out.
• Route Ethernet frames between the client (10.0.1.100) and the HTTP servers (192.168.2.2 & 172.64.3.10).

• Router should handle two types of packets:
  • ARP packets: requests & replies
  • IP Packets: ICMP & TCP/UDP

• Where do you start coding?
  • sr_handlepacket() method in sr_router.c
HOW TO DIFFERENTIATE PACKETS?

• “ether_type” is your friend. Check the ‘packet’ received in `sr_handlepacket()` using `ethertype` function defined in `sr_utils.c`.

```c
enum sr_ethertype {
    ethertype_arp = 0x0806,
    ethertype_ip = 0x0800,
};

struct sr_ethernet_hdr {
    #ifdef ETHER_ADDR_LEN
    #define ETHER_ADDR_LEN 6
    #endif

    uint8_t ether_dhost[ETHER_ADDR_LEN];
    uint8_t ether_shost[ETHER_ADDR_LEN];
    uint16_t ether_type;
} __attribute__((packed));

typedef struct sr_ethernet_hdr sr_ethernet_hdr_t;
```
Receive Raw Ethernet Frame

- It's an IP packet
  - Cache it, go through my request queue and send outstanding packets
  - Reply to me

- It's an ARP packet
  - Construct an ARP reply and send it back
  - Request to me
IF IT’S A ARP PACKET..

• Verify the length of the packet.
• Use `sr_get_interface` (in sr_if.c) to get the interface record. Get the mac address of the destination IP address/interacce IPaddress from the record.
• Check the ‘opcode’ variable of the ARP header and see if it is `arp_op_request` or `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 a Reply, update ARP cache and ARP queue. Send all the packets in the queue to the destination.
IP FLOW

Receive Raw Ethernet Frame

It's an IP packet
- It's for me
  - If it's ICMP echo req, send echo reply
  - Or if it's TCP/UDP, send ICMP port unreachable
- Not for me
  - Check routing table, perform LPM
  - No match
    - ICMP net unreachable

It's an ARP packet
- Check ARP cache
  - Hit
    - Send frame to next hop
  - Miss
    - Send ARP request
    - Resent >5 times
    - ICMP host unreachable
IF IT’S AN IP PACKET..

- Check the length.
- Validate the IP header.
  - Should not be IPv6
  - Check ip_hl
  - Check ip_len
  - Checksum
- Check if it is destined to you, the router.
  - Check ip_p.
  - Router should not handle non-ICMP packets (tcp or udp). Otherwise generate ICMP port unreachable (type 3, code 3).
  - If it is ICMP echo request (type 8), then generate ICMP echo reply (type 0).
IF IP PACKET IS NOT DESTINED TO THE ROUTER.. 

• Check ip_ttl. If TTL <= 1, send ICMP time exceeded (type 11, code 0).

• Look up next-hop address by doing a LPM on the routing table using the packet’s destination address. If it does not exist, send ICMP host unreachable (type 3, code 0).

• If it does exist, then reduce ttl and update checksum.

• From next-hop address, determine outgoing interface and next-hop MAC address

• If necessary, send ARP request to determine MAC address

• Encapsulate IP datagram in Ethernet packet

• Forward packet to outgoing interface
Consult the working binary shared in your VM (./sr_solution) using Wireshark, to get an idea of what packets are being sent/received.

GDB and Wireshark – your best buddies for next few weeks.

Use the Print functions available in sr_utils.c for printing out network header information from your packets.

Don’t get mixed up with endianness: Linux - little endian, network - big endian. Take a look at the print functions to get a clearer picture.

Make sure to push your code to your Git repo. VM image could get corrupted.

Set milestones so that you can keep track of how much work is pending.
QUESTIONS?