Computer Networks Assignment 2: Router ***Tutorial***

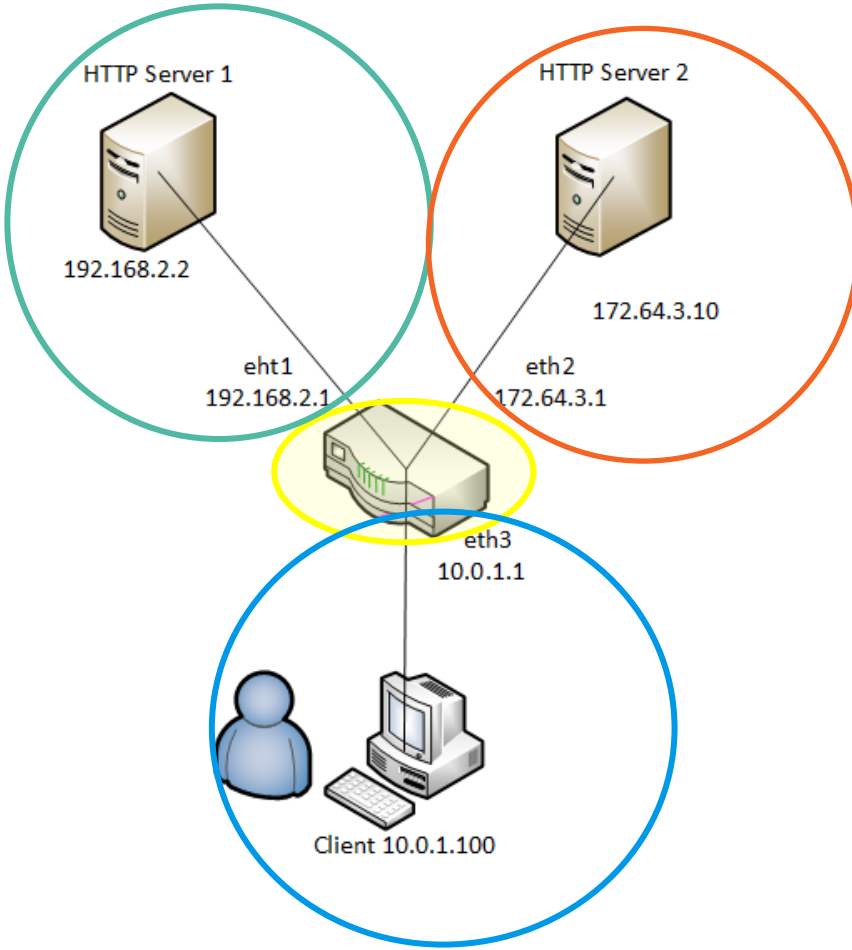
Tutorial: Oct 12th, 2022
Due Date: 5pm ***Nov 1st , 2022***

PART 1: ASSIGNMENT DESCRIPTION

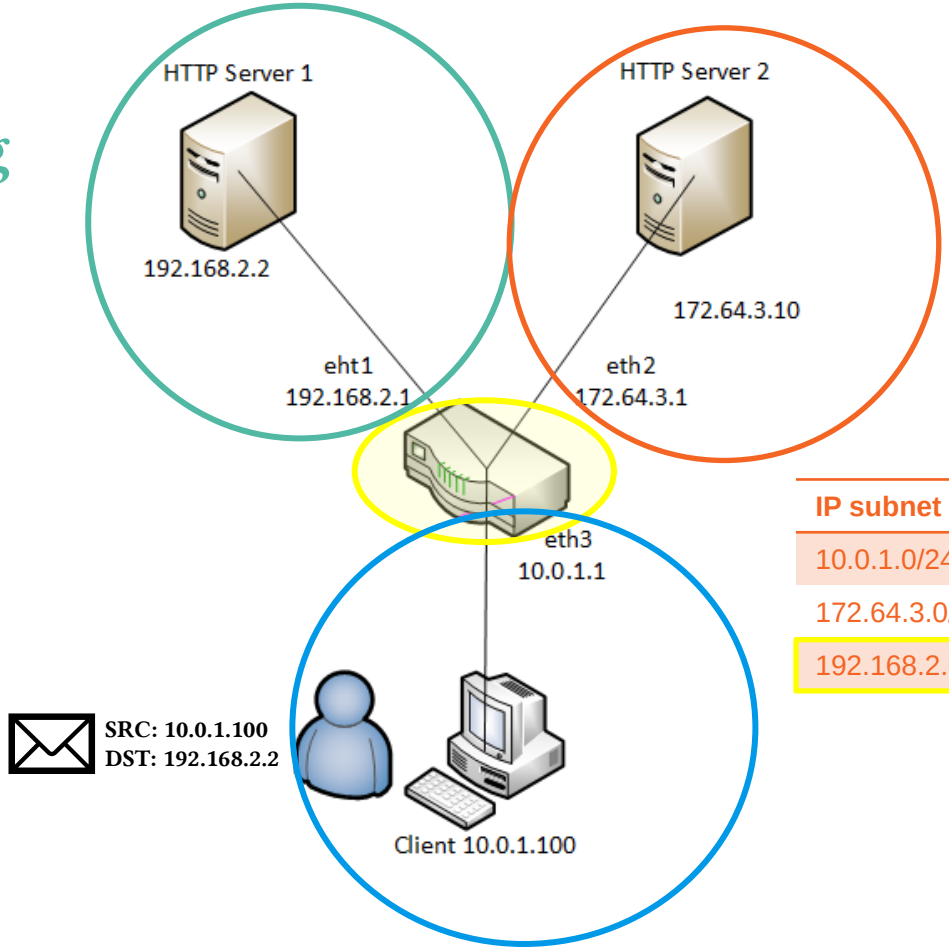
Implementing a simple router

Overview

- You will build a **simplified router** which will route packets from an emulated client to two emulated servers.
- Emulation environment: **Mininet**
- Three main parts to the assignment:
  - Preparing the emulation environment
  - Implementing IP forwarding
  - Handling address resolution (ARP)

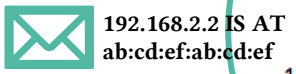


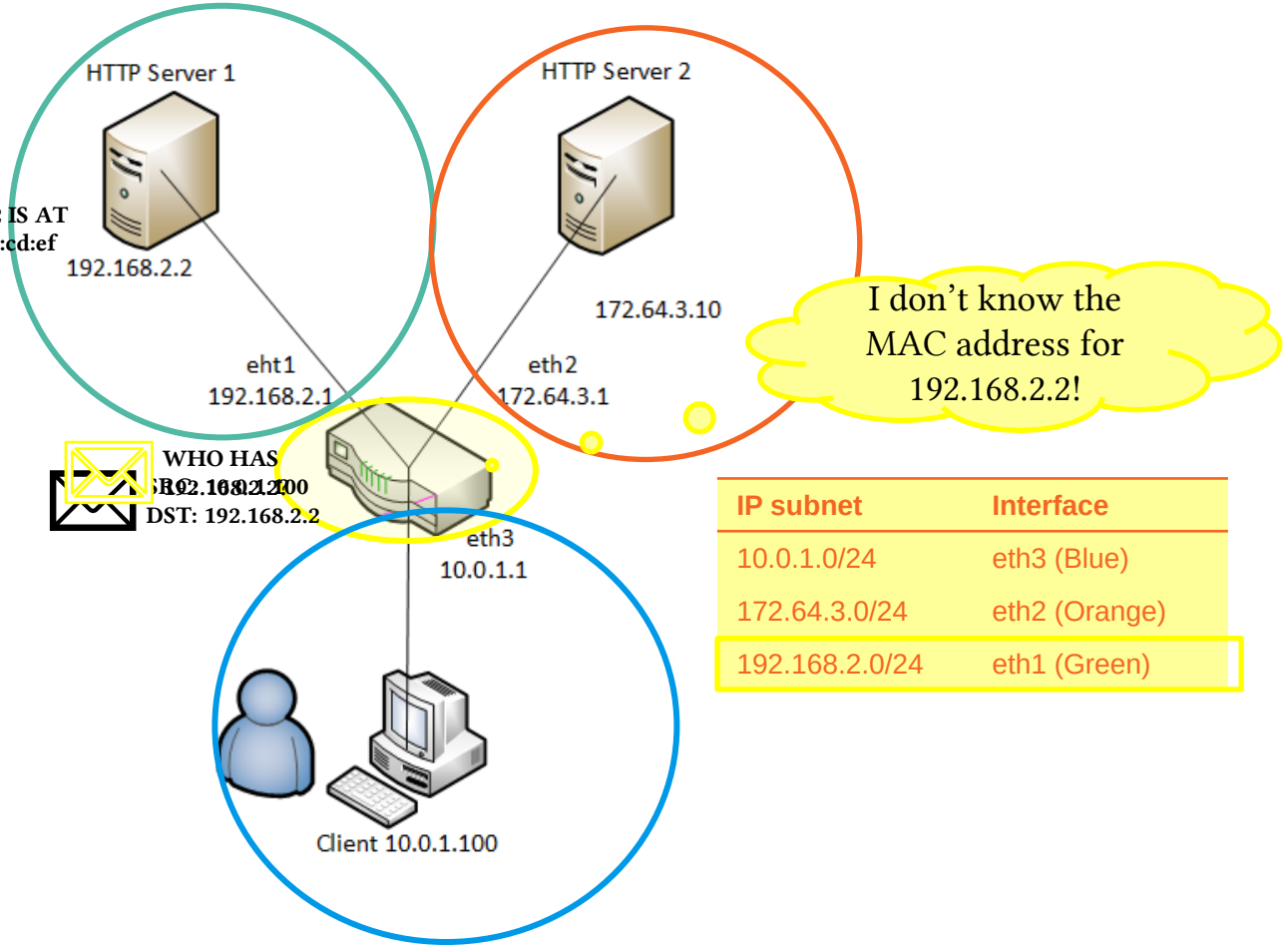
Overview:
IP Forwarding



IP subnet	Interface
10.0.1.0/24	eth3 (Blue)
172.64.3.0/24	eth2 (Orange)
192.168.2.0/24	eth1 (Green)

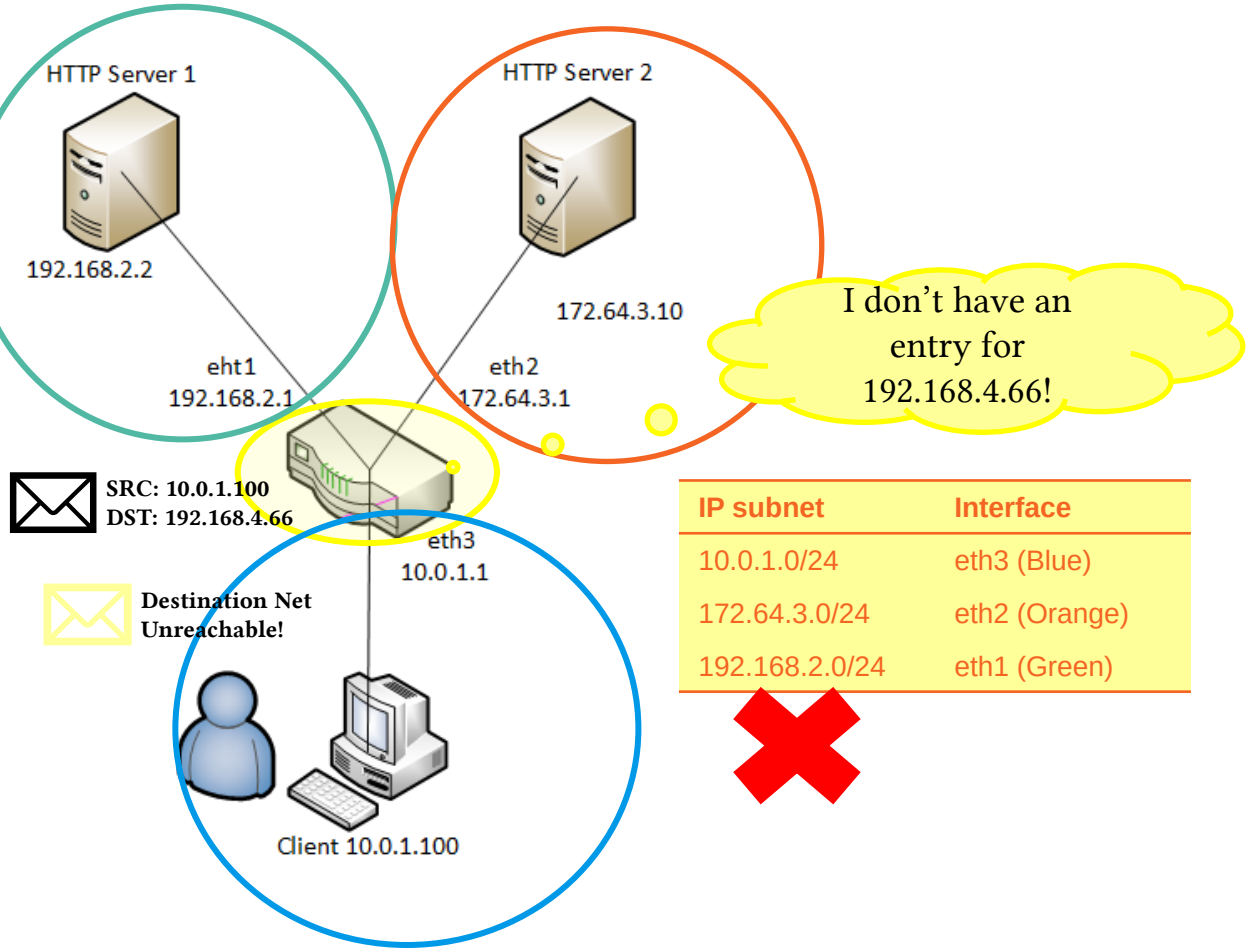
Overview:
ARP Handling





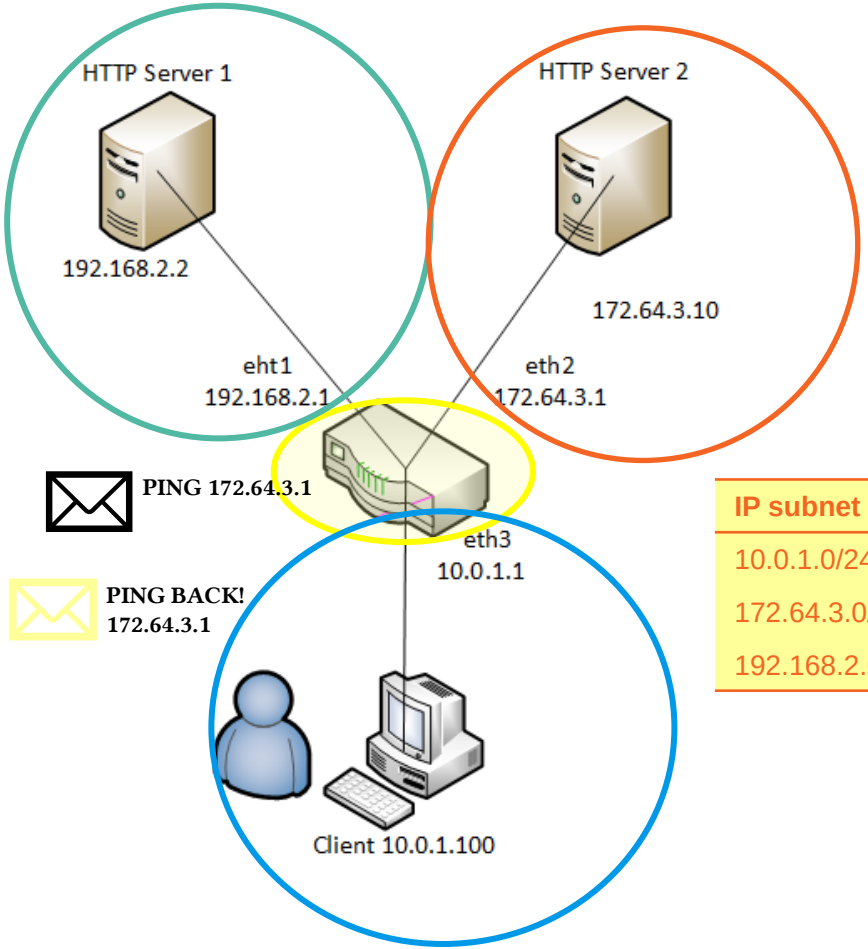
IP subnet	Interface
10.0.1.0/24	eth3 (Blue)
172.64.3.0/24	eth2 (Orange)
192.168.2.0/24	eth1 (Green)

Overview:
ICMP handling



IP subnet	Interface
10.0.1.0/24	eth3 (Blue)
172.64.3.0/24	eth2 (Orange)
192.168.2.0/24	eth1 (Green)

Overview:
ICMP handling



IP subnet	Interface
10.0.1.0/24	eth3 (Blue)
172.64.3.0/24	eth2 (Orange)
192.168.2.0/24	eth1 (Green)

Raw Ethernet Frames

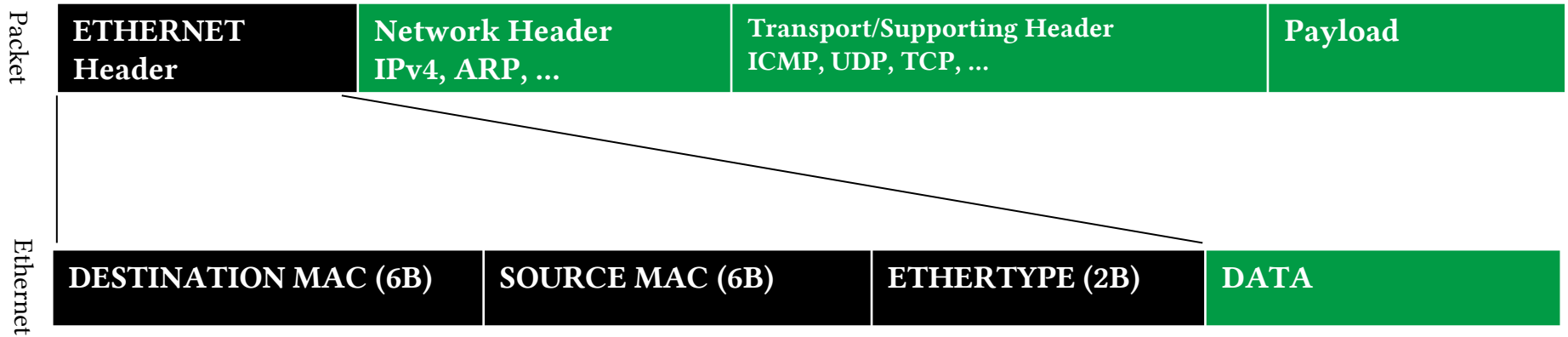
- Your router will receive and send raw Ethernet frames.

What the router software receives

Ethernet

- Your router will receive and send raw Ethernet frames.
- Ethernet frames have headers which contain source and destination MAC addresses.
  - To forward a packet one hop, we must change the destination MAC address of the forwarded packet to the MAC address of the next hop's incoming interface.
- Your router must process packets & forward them to the correct interface.
- When your router receives a packet
  - First, identify the packet (IP, ARP)
  - Then, extract appropriate header

Ethernet Header

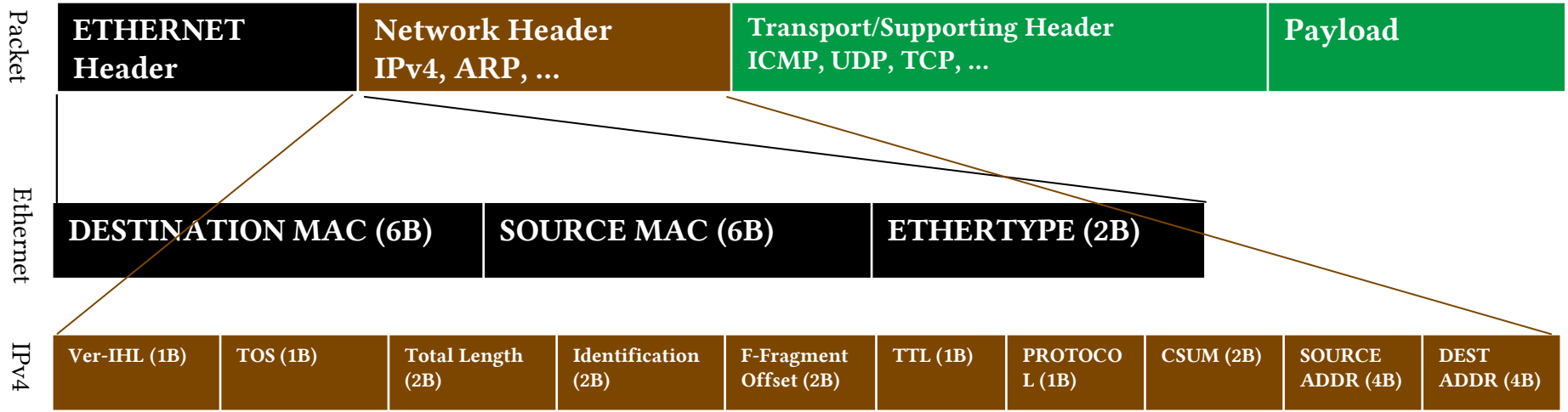


Internet Protocol (IPv4)

If your router receives an IPv4 packet:

1. Perform basic checks
    a. Verify its checksum
    b. Check that it meets minimum length
2. Determine if it is destined for one of our interfaces:
    a. If yes, check that it is an ICMP echo request (type 8)
        i. If yes, send **ICMP echo reply (type 0)** back
        ii. Else, drop it. If not even an ICMP packet, send **ICMP port unreachable (type 3, code 3)**
    b. If not destined for our interfaces, IPv4 packet **must be forwarded**.

IPv4 Header


Packet
ETHERNET Header
Network Header IPv4, ARP, ...
Transport/Supporting Header ICMP, UDP, TCP, ...
Payload
Ethernet
DESTINATION MAC (6B)
SOURCE MAC (6B)
ETHERTYPE (2B)
IPv4
Ver-IHL (1B)
TOS (1B)
Total Length (2B)
Identification (2B)
F-Fragment Offset (2B)
TTL (1B)
PROTOCOL (1B)
CSUM (2B)
SOURCE ADDR (4B)
DEST ADDR (4B)

IP Forwarding

If packet must be forwarded:

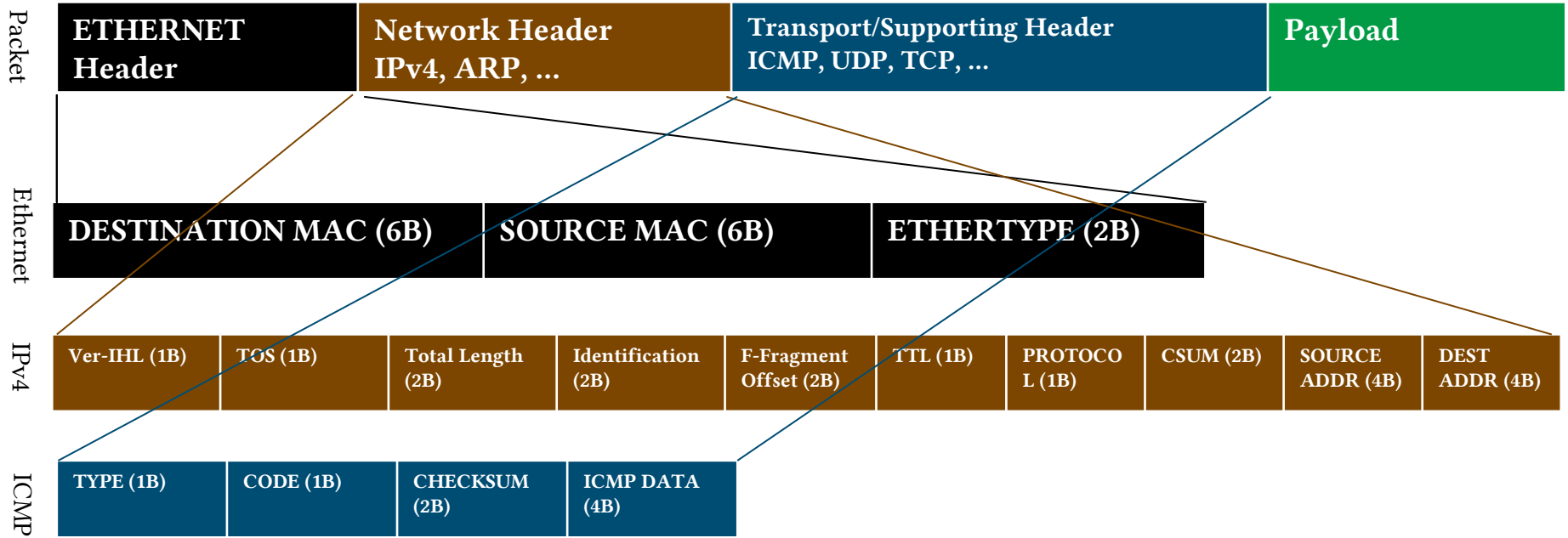
1. Decrement TTL by 1, recompute packet checksum
    a. If TTL is expired, send **ICMP time exceeded (type 11)**
2. Find match for the destination IP address in the routing table
    a. If no match, send **ICMP destination net unreachable (type 3, code 0)**
    b. If match, check ARP cache for next-hop MAC address corresponding to next-hop IP
        i. If ARP entry is found, **send IP packet to its next hop**
        ii. If no ARP entry is found:
            1. Send ARP request for next-hop IP
            2. Put **packet onto a queue** as it waits for an ARP reply.

ICMP

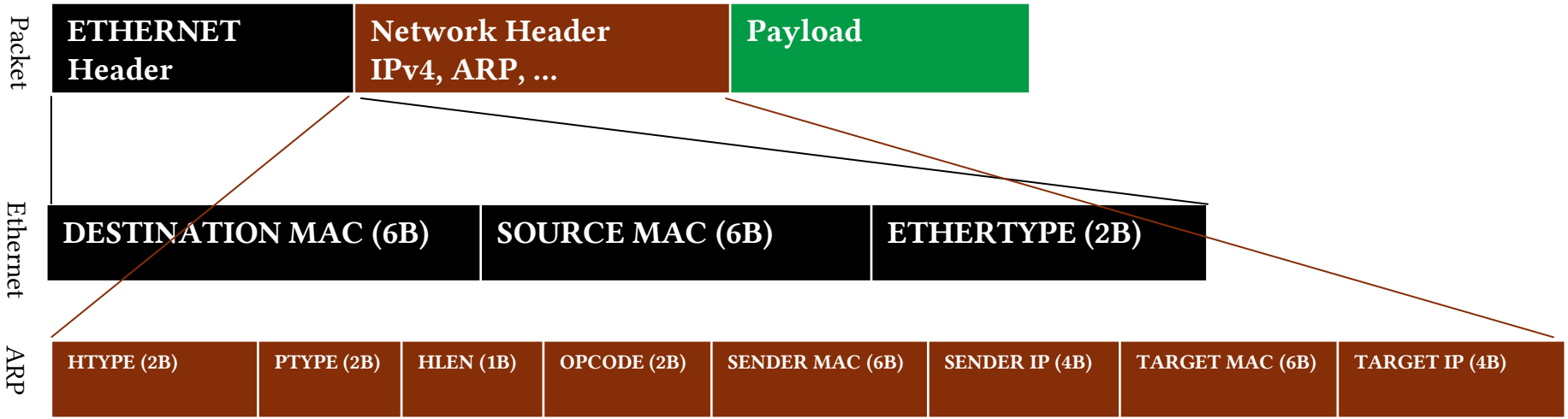
Use ICMP messages to send control information to host:

- **Echo reply (type 0)**
  - Send in response to echo request (type 8). Used for PING.
- **Destination net unreachable (type 3, code 0)**
  - Sent if there is a non-existent route to the destination IP.
- **Destination host unreachable (type 3, code 1)**
  - Sent if five ARP requests were sent to the next-hop IP without a response.
- **Port unreachable (type3, code 3)**
  - Sent if an IP packet containing a UDP or TCP payload is sent to one of the router's interfaces.
- **Time exceeded (type 11, code 0)**
  - Sent if an IP packet is discarded because the TTL field is 0.

ICMP Header


Packet
ETHERNET Header
Network Header IPv4, ARP, ...
Transport/Supporting Header ICMP, UDP, TCP, ...
Payload
Ethernet
DESTINATION MAC (6B)
SOURCE MAC (6B)
ETHERTYPE (2B)
IPv4
Ver-IHL (1B)
TOS (1B)
Total Length (2B)
Identification (2B)
F-Fragment Offset (2B)
TTL (1B)
PROTOCOL (1B)
CSUM (2B)
SOURCE ADDR (4B)
DEST ADDR (4B)
ICMP
TYPE (1B)
CODE (1B)
CHECKSUM (2B)
ICMP DATA (4B)

ARP Header


Packet
ETHERNET Header
Network Header IPv4, ARP, ...
Payload
Ethernet
DESTINATION MAC (6B)
SOURCE MAC (6B)
ETHERTYPE (2B)
ARP
HTYPE (2B)
PTYPE (2B)
HLEN (1B)
OPCODE (2B)
SENDER MAC (6B)
SENDER IP (4B)
TARGET MAC (6B)
TARGET IP (4B)

Address Resolution Protocol (ARP)

- ARP determines the next-hop MAC that corresponds to the next-hop IP
- Your router needs to generate and process ARP requests/replies.
  - You are required to cache ARP replies that your router receives
  - We handle the cache timeout for you
- During the packet forwarding process, the router should first check the ARP cache for the MAC address before sending an ARP request

Address Resolution Protocol (ARP)

If your router receives an ARP packet:

- Check if it's an ARP request or ARP reply:
  - If ARP request:
    - Check if its target IP address is one of our router's IP addresses.
      - If yes, **send ARP reply**
      - Else, drop it
  - If ARP reply:
    - **Cache the ARP reply** as long as the target IP address is one of your router's IP addresses
    - If there is an ARP request waiting on the reply
      - **Send IP packets** waiting on this ARP reply
      - Remove corresponding ARP request from queue

END OF PART 1: Assignment Description

Q&A

PART 2: TESTING ENVIRONMENT

Deploying Mininet using Vagrant
&
Getting started

Mininet VM Setup

- Checkout the appropriate branch on Github
- Custom Vagrant VM setup (in ./Assignment2/ dir) → **must use this one**!
- Mininet emulator is *already set up* for you, so it should be painless :)
- Test connectivity using the reference binary provided.
  - **Do this BEFORE writing any code.**
  - **Do this early** so we can help resolve any setup issues.
  - If there are any connectivity issues, you will not be able to test your implementation.
- Step-by-step instructions are in the README

Getting Started: sr_router.c

- In `sr_router.c`:

```
void sr_handlepacket(struct sr_instance* sr, uint8_t *
packet, unsigned int len, char* interface)
```

- This method is called every time the router receives a raw Ethernet frame.
- You will have to implement the logic to process the packets and send them to the correct interface.
  - **Tip:** First check if it is an IP or an ARP packet using `ethertype()` method from `sr_utils.c/h`, then process accordingly
  - Use helper methods!

Helper functions: sending a packet

- In `sr_vns_comm.c`:

  ```
  int sr_send_packet(struct sr_instance* sr, uint8_t* buf,
  unsigned int len, const char* iface)
  ```

- Sends a packet of length `len` to the network out of `iface`
- Your `sr_router.c` should call this method
- You do not need to modify this method

Getting Started: sr_arpcache.c

- In `sr_arpcache.c`:

  ```
  void sr_arpcache_sweepreqs(struct sr_instance *sr)
  ```

- Called once per second
- You should add code to iterate through the ARP request queue and re-send outstanding requests
- Fill in the helper method called e.g. `handle_arpreq()` to track of how many times the request has been sent
  - If sent >= 5 times and no response, send **ICMP destination host unreachable (type 3, code 1)**
- **Tip:** there is helpful pseudocode for this in `sr_arpcache.h`

Summary

- Code to be written in `sr_router.c`:
  - Handle IP packet
    - Send ICMP echo replies
    - Send (forward) IP packets
    - Queue ARP requests
  - Handle ARP packet
    - Send ARP replies in response to requests
    - Cache incoming ARP reply
    - Send IP packets that are waiting on incoming ARP reply
    - Remove corresponding ARP request from queue
  - Send ICMP messages based on certain conditions

- Code to be written in sr_arpcache.c:
  - Iterate through ARP request queue
  - Send and re-send ARP requests
  - Check if we need to re-send or destroy ARP request

- There are some helpful methods to look up interfaces in `sr_if.c`.

- There are some generally helpful methods in `sr_utils.c`.

These are just my recommendations for where to place different functionality.
You may organize the functionality across the code base as you see fit.

Final Thoughts

- Please start early.
  - I repeat, **start early.**
- Come to office hours
- Ask questions on Canvas