Build Your Own Router

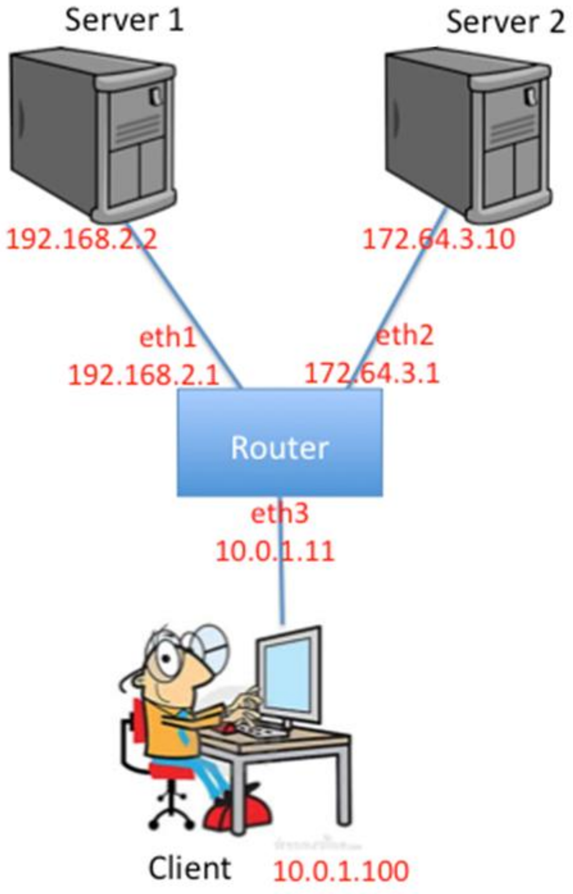
Computer Networks

Course Project

Project Overview

- Implement a simple router in a single router topology with static routing table (forwarding table).

- Your router will receive raw Ethernet frames, and handle/forward packets in correct logic.



LESSONS TO LEARN

What about Our Router?

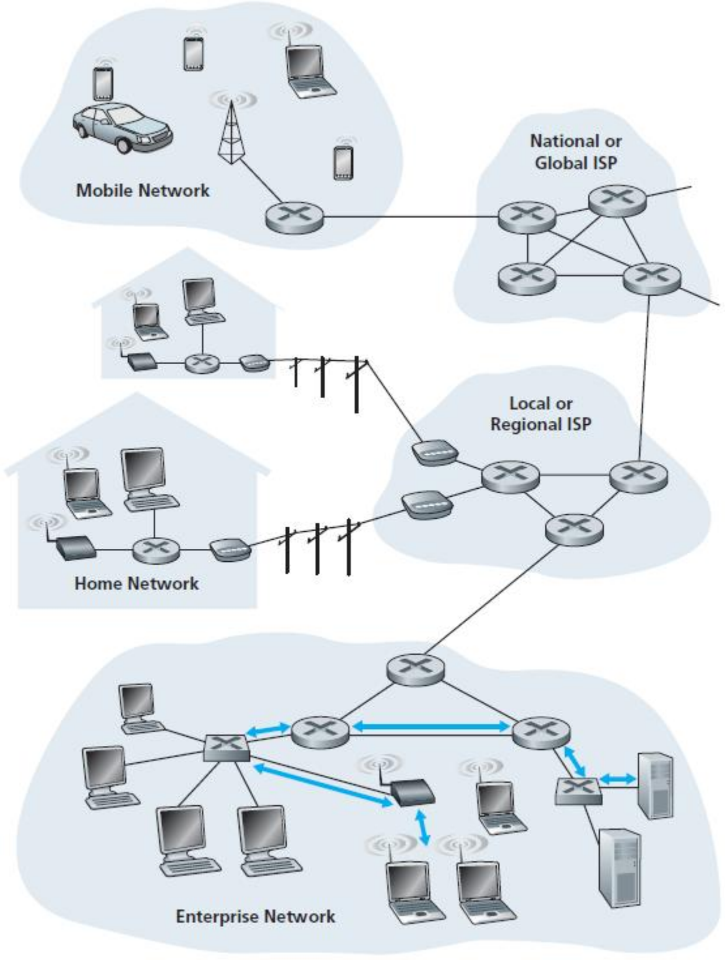
- **Forwarding**: move packets from router's input to appropriate router output
  - Load predefined routing table
  - Look up matching entry in routing table
- Handle ICMP
- Handle ARP request / reply
- ~~**Routing**: determine route taken by packets from source to dest.~~

Link Layer

- Transfer internet layer datagram from node to adjacent node over a link

- Encapsulate internet layer datagram into frame, add header & trailer

- Use "MAC" address in frame headers for source, dest

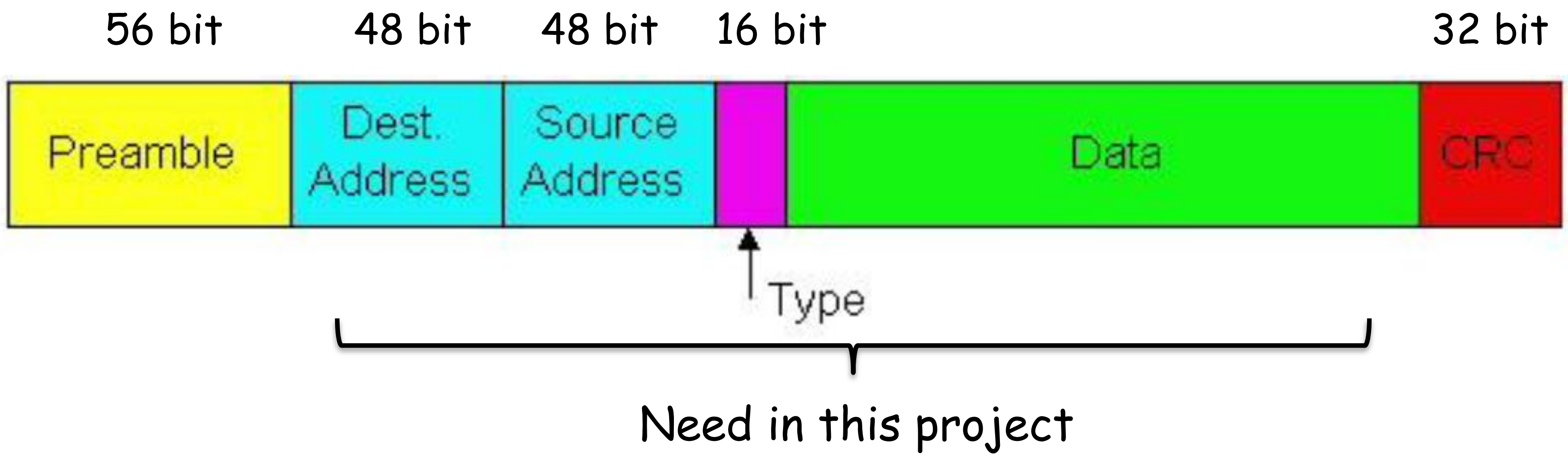


MAC address v.s. IP address

- 32-bit IP address:
  - network-layer address
  - used to get datagram from src to dest IP subnet

- 48-bit MAC (Ethernet) address:
  - link-layer address for network interfaces
  - get frame from one interface to another physically-connected interface (same subnet)
  - Broadcast address: “FF:FF:FF:FF:FF:FF”

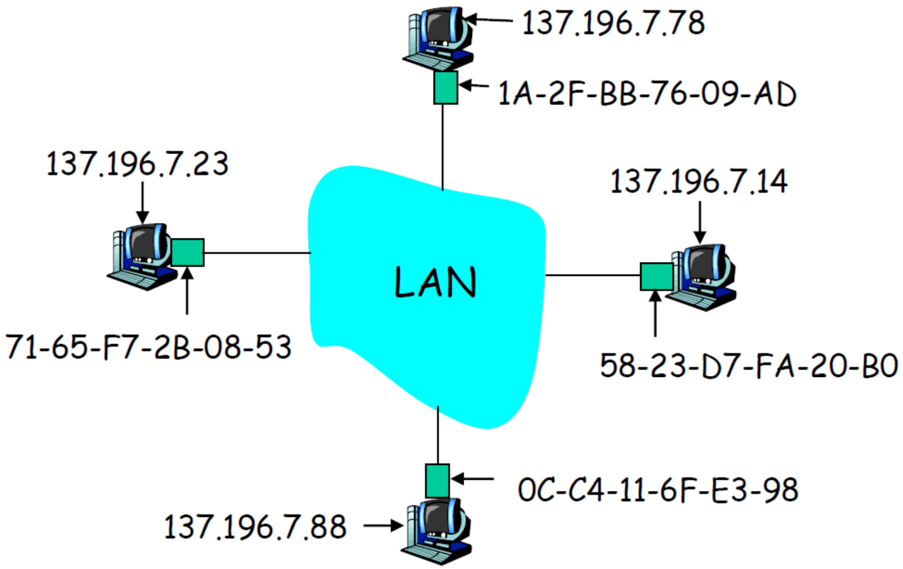
Ethernet Frame



- Encapsulates IP datagram (or other network layer protocol packet) in Ethernet frame
  - preamble and crc are handled transparently in this project

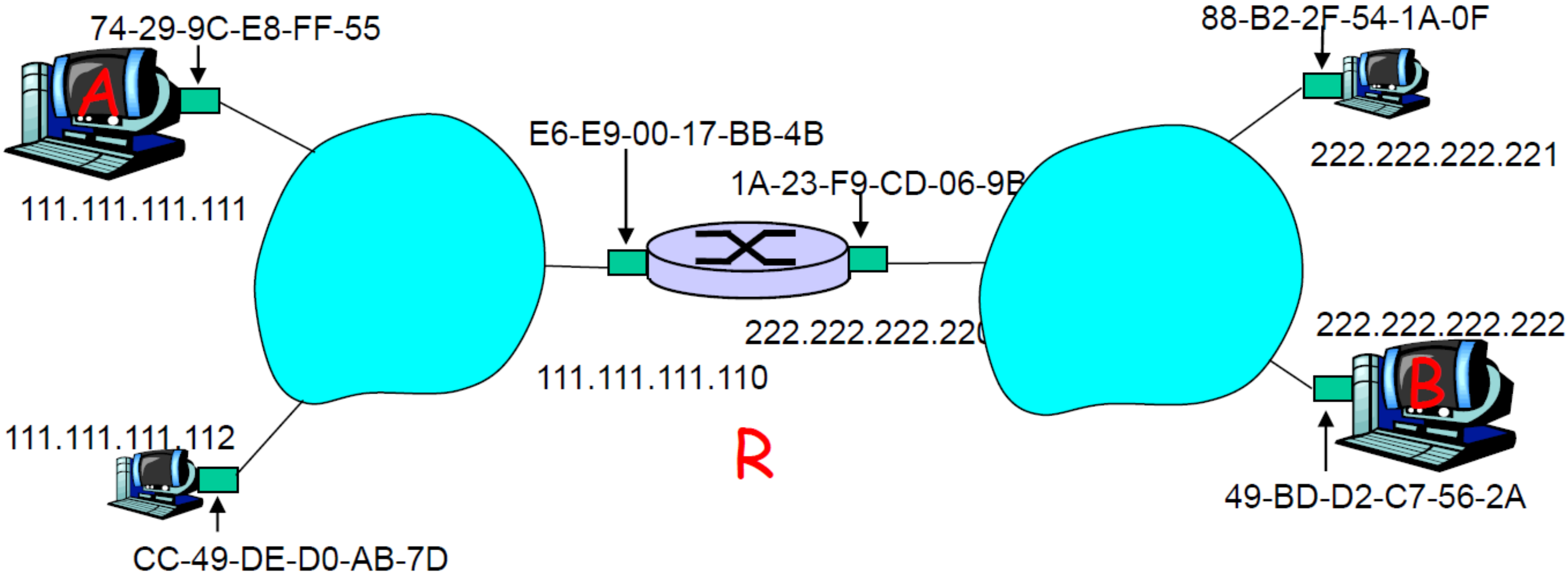
ARP: Address Resolution Protocol

Question: How to get MAC address of B from B's IP address



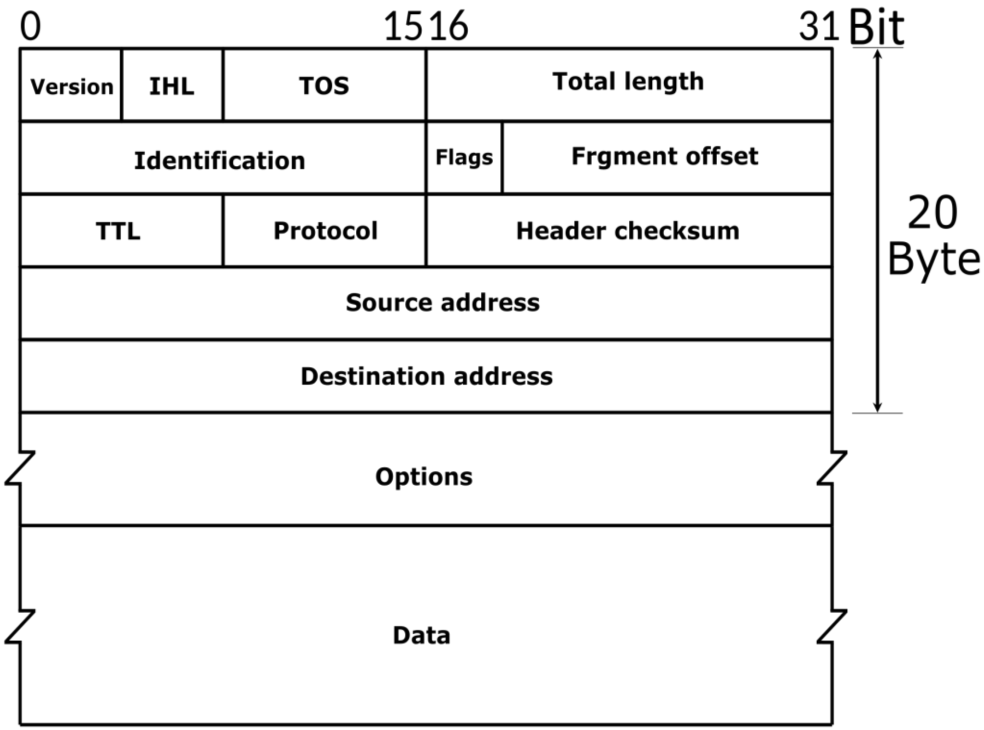
- ARP request: request IP-MAC mapping of next hop interface (send to broadcast address)
- ARP reply: send IP-MAC of current interface
- ARP cache: IP – MAC mapping for nodes (timeout after a 30s)

ARP: Address Resolution Protocol



Walkthrough: send datagram from A to B through R.

Internet Protocol (IPv4)



- Delivering packets from src to dest based on IP address

- Header checksum

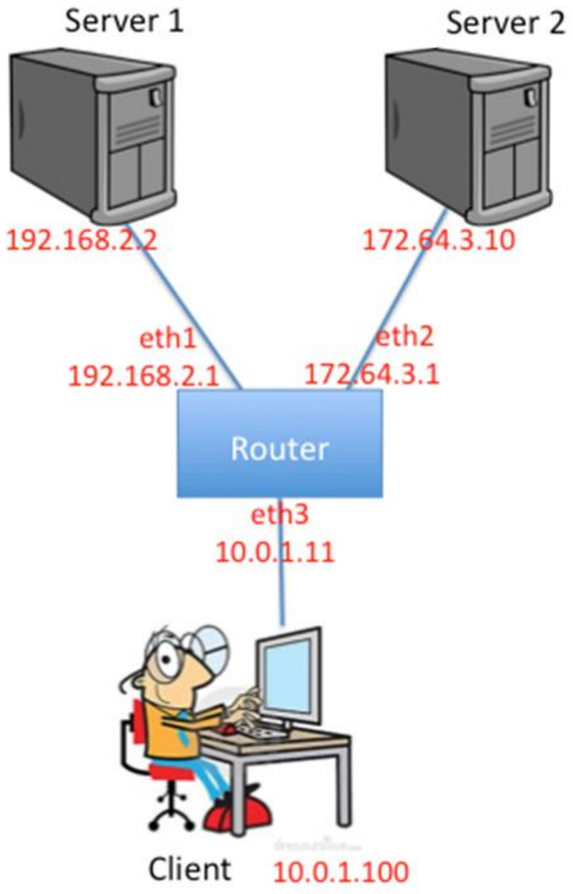
- Decrement TTL

Routing Table

- Destination & Netmask: subnet network ID
- Gateway: next hop IP address to destination
- Interface: name of network interface card connected to gateway
- Metric: routing metric of path to destination (omitted in this project)

Longest Prefix Match

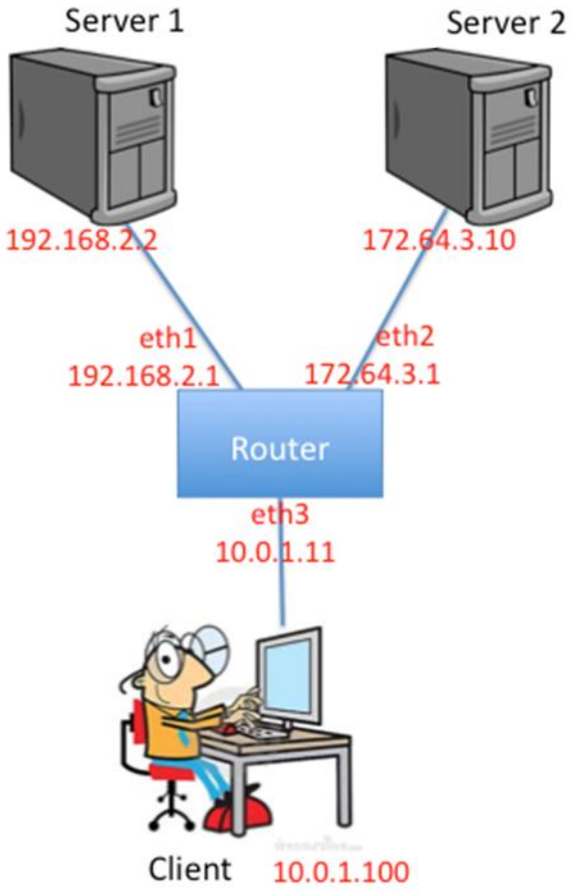
Destination	Netmask	Gateway	Interface
0.0.0.0	0.0.0.0	10.0.1.100	eth3
192.168.2.2	255.255.255.0	*	eth1
172.64.3.10	255.255.0.0	*	eth2



Where should packet dest at "192.168.1.1" go?

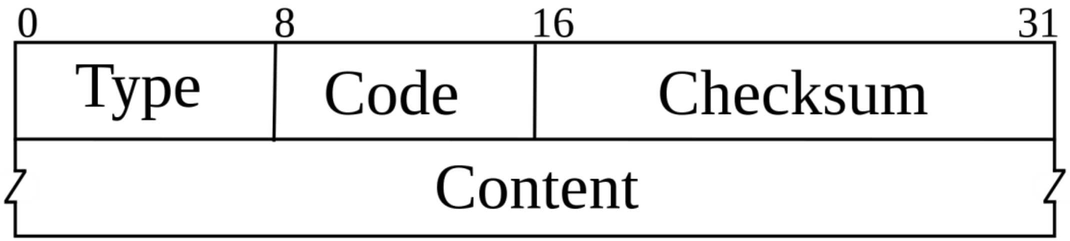
Longest Prefix Match

Destination	Netmask	Gateway	Interface
0.0.0.0	0.0.0.0	10.0.1.100	eth3
192.168.2.2	255.255.255.0	*	eth1
172.64.3.10	255.255.0.0	*	eth2



Where should packet dest at “192.168.2.1” go?

ICMP: Internet Control Message Protocol



0	8	16	31
Type	Code	Checksum	
Content			

- Used by hosts & routers to communicate network-level information
  - error reporting (unreachable host, network, port, protocol)
  - echo request/reply
- Sent as IPv4 payload
- Content: Internet header + 8 bytes of original datagram

Ping & Traceroute

- **Ping**: ICMP echo request/reply
  - reply with TTL = 64

Ping & Traceroute

- **Ping**: ICMP echo request/reply
  - reply with TTL = 64

- **Traceroute**: displaying possible routes and RTT in IP network.
  - sends UDP segments with TTL = 1,2,3, … with unlikely port number
  - when nodes receive datagram with TTL=0, return ICMP "Time Exceeded"
  - destination returns ICMP "Port Unreachable"

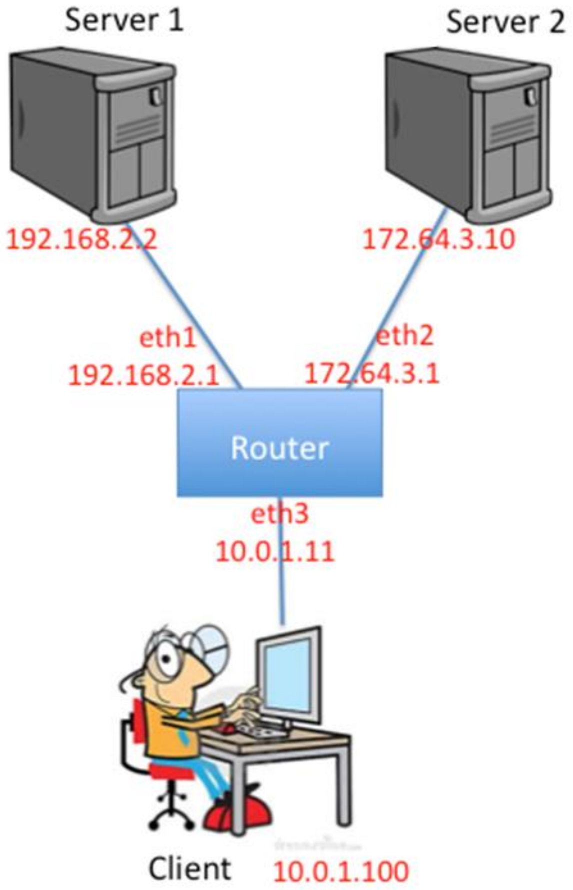
Useful Materials

- IPv4:
  - RFC 791: https://tools.ietf.org/html/rfc791
  - Text Book: section 4.4.1, 4.4.2
- ICMP:
  - RFC 792: https://tools.ietf.org/html/rfc792
  - Text Book: section 4.4.3
- ARP:
  - RFC 826: https://tools.ietf.org/html/rfc826
  - Text Book: section 5.4.1

RESULTS TO SHOW

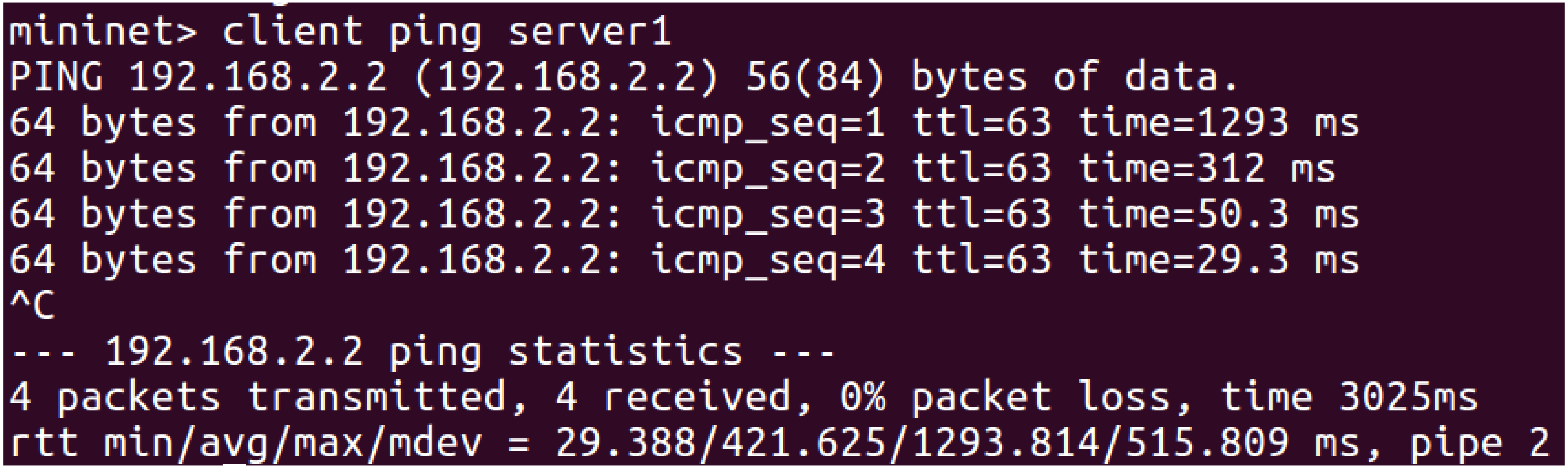
Expected Behaviors

- **ping** from client to any server & router interfaces
- **traceroute** from client to any server & router interfaces
- **wget** files from server
- Update ARP cache table



Ping

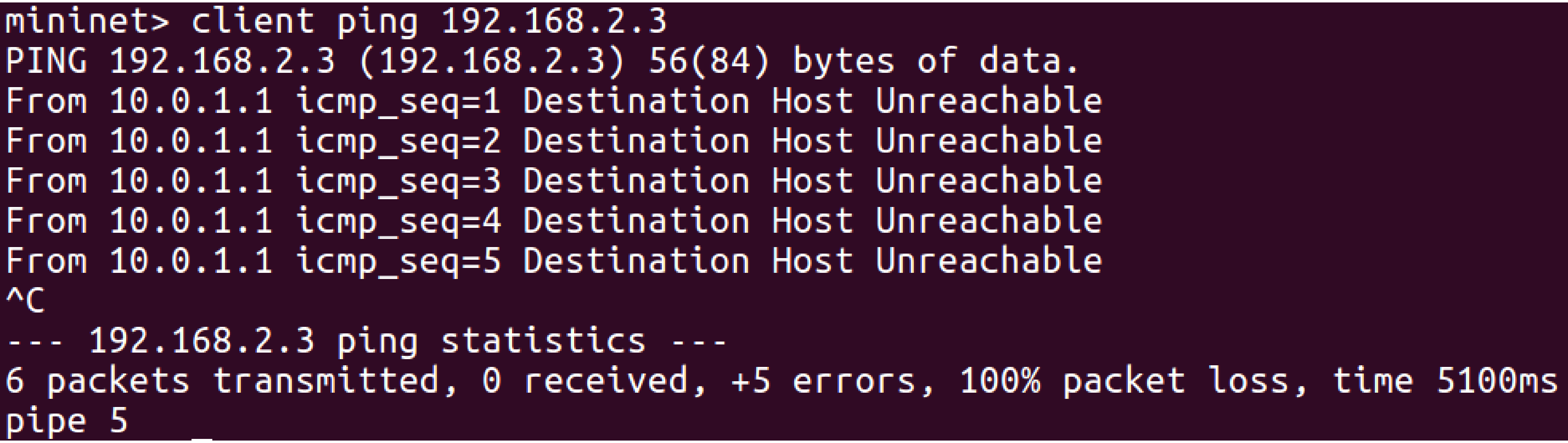
- ping from client to any http servers



```
mininet> client ping server1
PING 192.168.2.2 (192.168.2.2) 56(84) bytes of data.
64 bytes from 192.168.2.2: icmp_seq=1 ttl=63 time=1293 ms
64 bytes from 192.168.2.2: icmp_seq=2 ttl=63 time=312 ms
64 bytes from 192.168.2.2: icmp_seq=3 ttl=63 time=50.3 ms
64 bytes from 192.168.2.2: icmp_seq=4 ttl=63 time=29.3 ms
^C
--- 192.168.2.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3025ms
rtt min/avg/max/mdev = 29.388/421.625/1293.814/515.809 ms, pipe 2
```

Ping

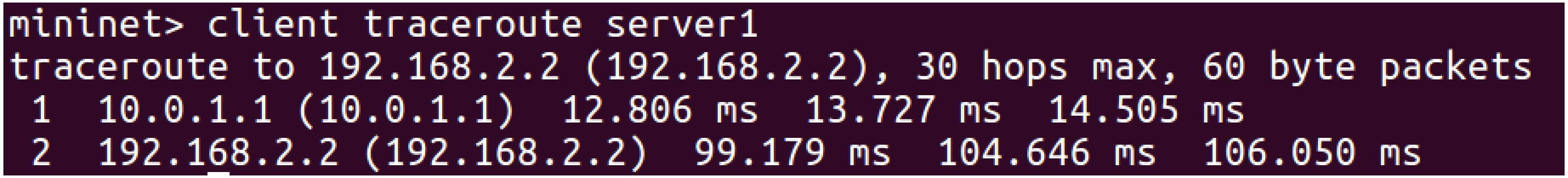
- ping wrong IP address



```
mininet> client ping 192.168.2.3
PING 192.168.2.3 (192.168.2.3) 56(84) bytes of data.
From 10.0.1.1 icmp_seq=1 Destination Host Unreachable
From 10.0.1.1 icmp_seq=2 Destination Host Unreachable
From 10.0.1.1 icmp_seq=3 Destination Host Unreachable
From 10.0.1.1 icmp_seq=4 Destination Host Unreachable
From 10.0.1.1 icmp_seq=5 Destination Host Unreachable
^C
--- 192.168.2.3 ping statistics ---
6 packets transmitted, 0 received, +5 errors, 100% packet loss, time 5100ms
pipe 5
```

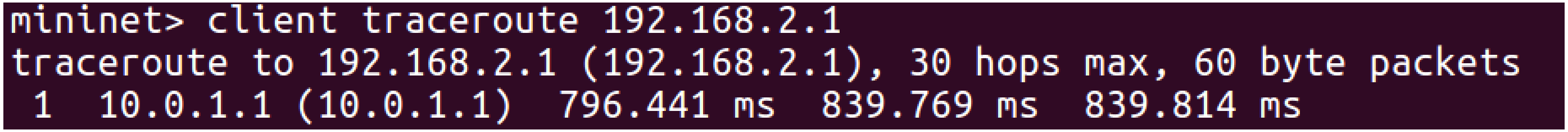
Traceroute

- traceroute to any http server



```
mininet> client traceroute server1
traceroute to 192.168.2.2 (192.168.2.2), 30 hops max, 60 byte packets
 1  10.0.1.1 (10.0.1.1)  12.806 ms  13.727 ms  14.505 ms
 2  192.168.2.2 (192.168.2.2)  99.179 ms  104.646 ms  106.050 ms
```

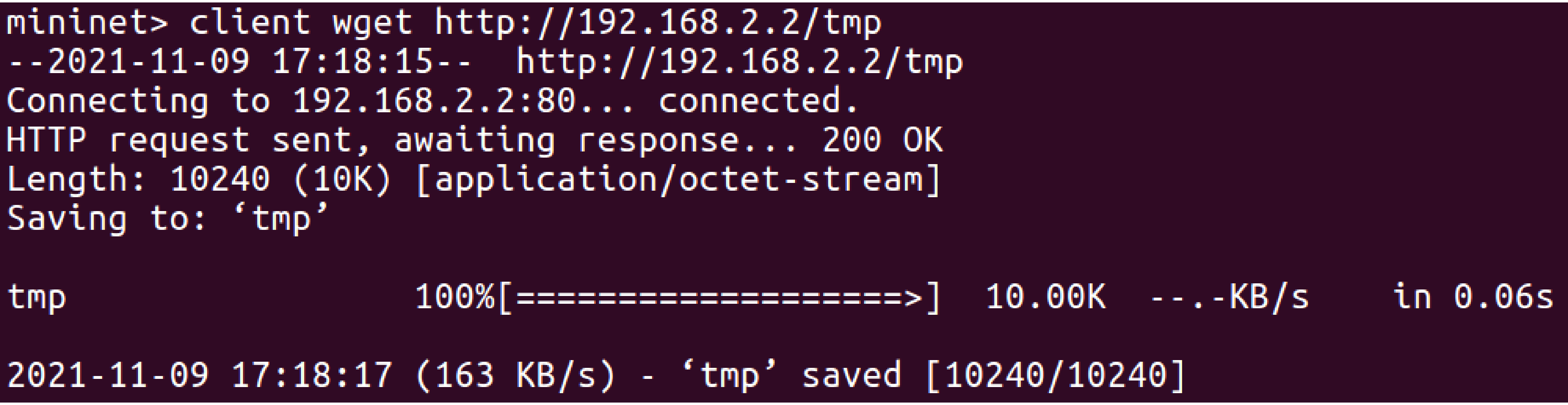
- traceroute to router interfaces



```
mininet> client traceroute 192.168.2.1
traceroute to 192.168.2.1 (192.168.2.1), 30 hops max, 60 byte packets
 1  10.0.1.1 (10.0.1.1)  796.441 ms  839.769 ms  839.814 ms
```

File Downloading

- wget from any of servers



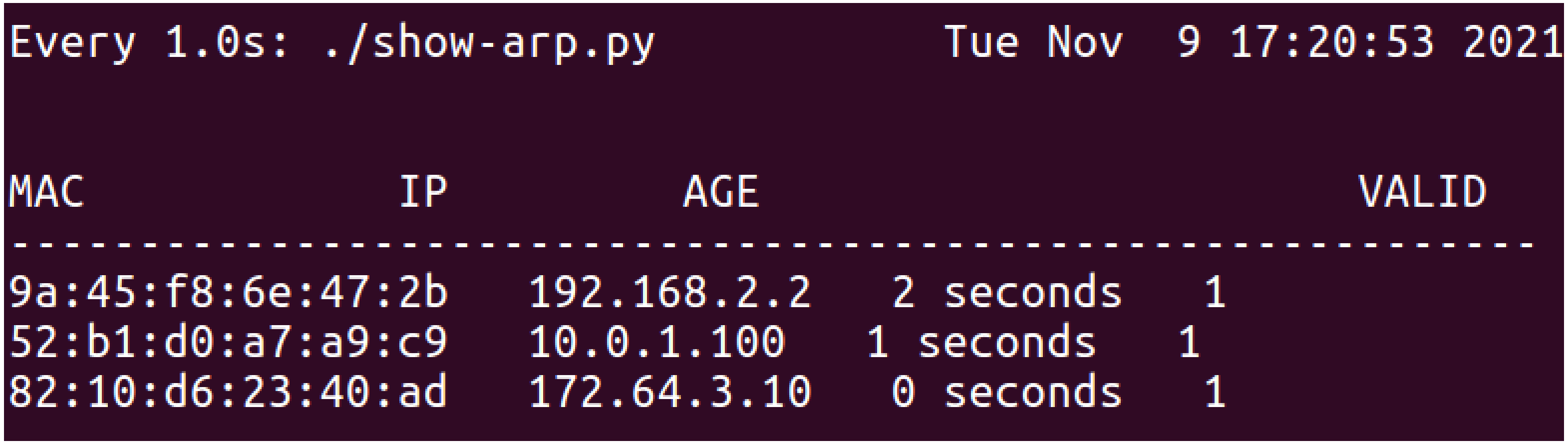
```
mininet> client wget http://192.168.2.2/tmp
--2021-11-09 17:18:15--  http://192.168.2.2/tmp
Connecting to 192.168.2.2:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 10240 (10K) [application/octet-stream]
Saving to: ‘tmp’

tmp                 100%[===================>]  10.00K  --.-KB/s    in 0.06s

2021-11-09 17:18:17 (163 KB/s) - ‘tmp’ saved [10240/10240]
```

ARP Cache table

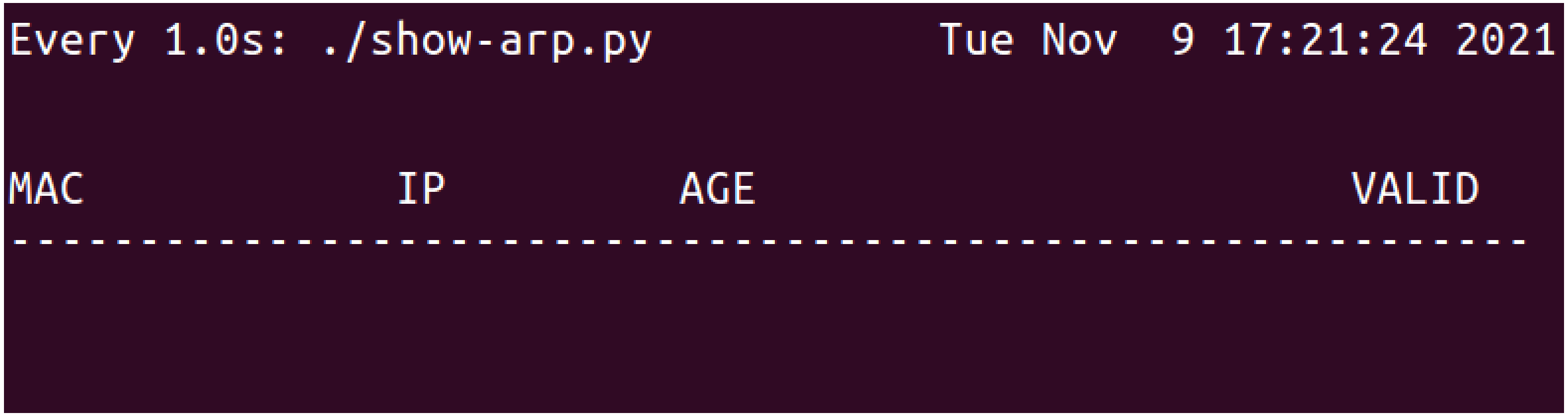
- “pingall”



```
Every 1.0s: ./show-arp.py           Tue Nov  9 17:20:53 2021


MAC            IP         AGE                       VALID
-----------------------------------------------------------
9a:45:f8:6e:47:2b   192.168.2.2   2 seconds   1
52:b1:d0:a7:a9:c9   10.0.1.100   1 seconds   1
82:10:d6:23:40:ad   172.64.3.10   0 seconds   1
```

- 30 seconds later

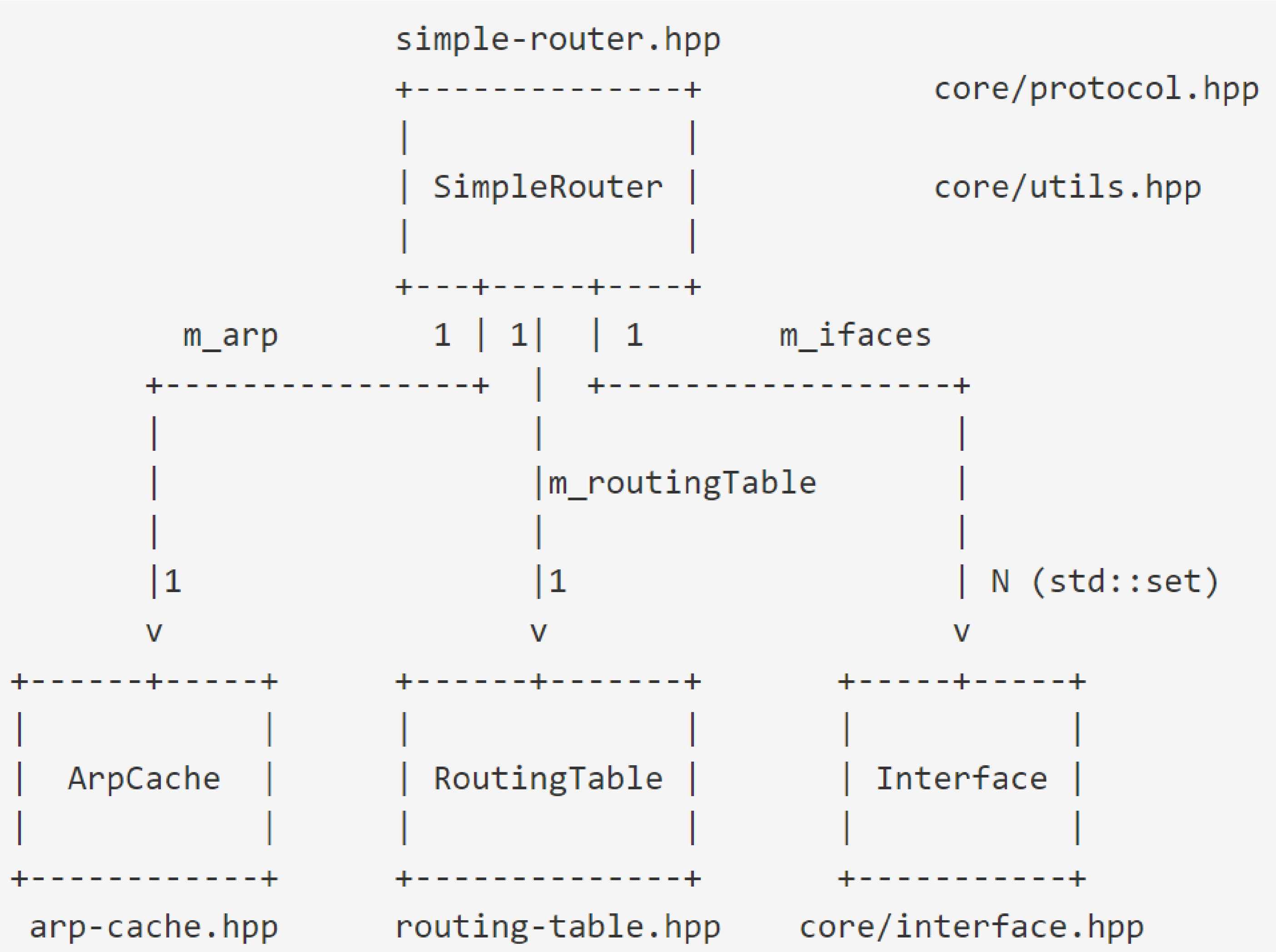


```
Every 1.0s: ./show-arp.py           Tue Nov  9 17:21:24 2021


MAC            IP         AGE                       VALID
-----------------------------------------------------------
```

CODE TO IMPLEMENT

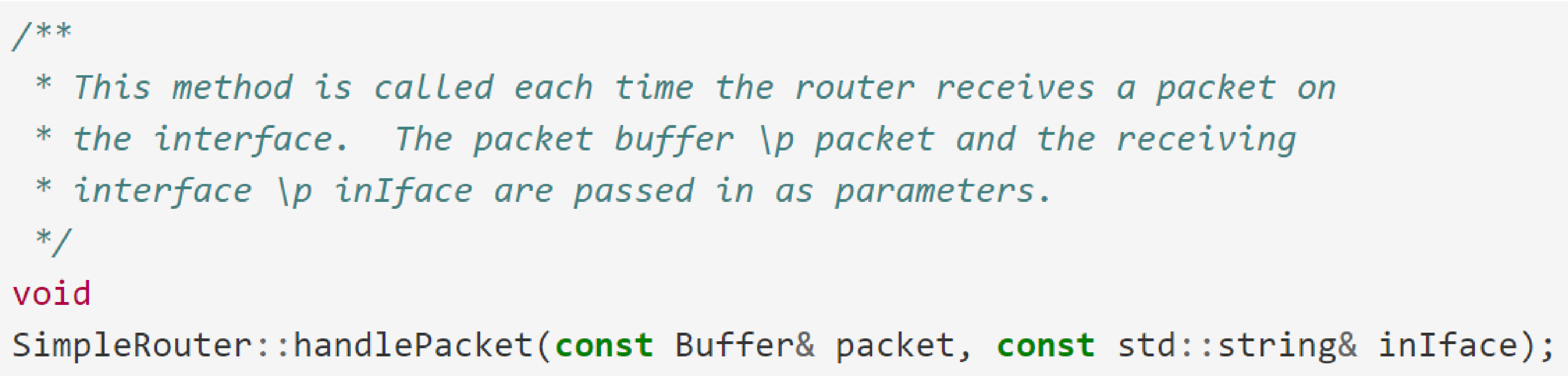
Code Structure



```
                    simple-router.hpp
                    +--------------+            core/protocol.hpp
                    |              |
                    | SimpleRouter |            core/utils.hpp
                    |              |
                    +---+-----+----+
         m_arp        1 | 1|  | 1       m_ifaces
       +----------------+  |  +------------------+
       |                   |                     |
       |                   |m_routingTable       |
       |                   |                     |
       |1                  |1                    | N (std::set)
       v                   v                     v
+------+-----+      +------+-------+       +-----+-----+
|            |      |              |       |           |
|  ArpCache  |      | RoutingTable |       | Interface |
|            |      |              |       |           |
+------------+      +--------------+       +-----------+
 arp-cache.hpp      routing-table.hpp    core/interface.hpp
```

Key Methods

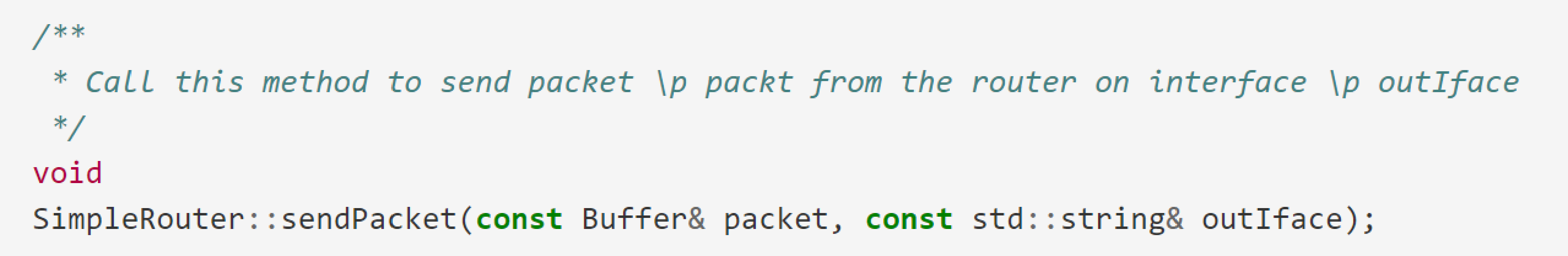
- NEED TO IMPLEMENT
- Method that receives a raw Ethernet frame (simple-router.hpp|cpp):



```
/**
 * This method is called each time the router receives a packet on
 * the interface.  The packet buffer \p packet and the receiving
 * interface \p inIface are passed in as parameters.
 */
void
SimpleRouter::handlePacket(const Buffer& packet, const std::string& inIface);
```

Key Methods

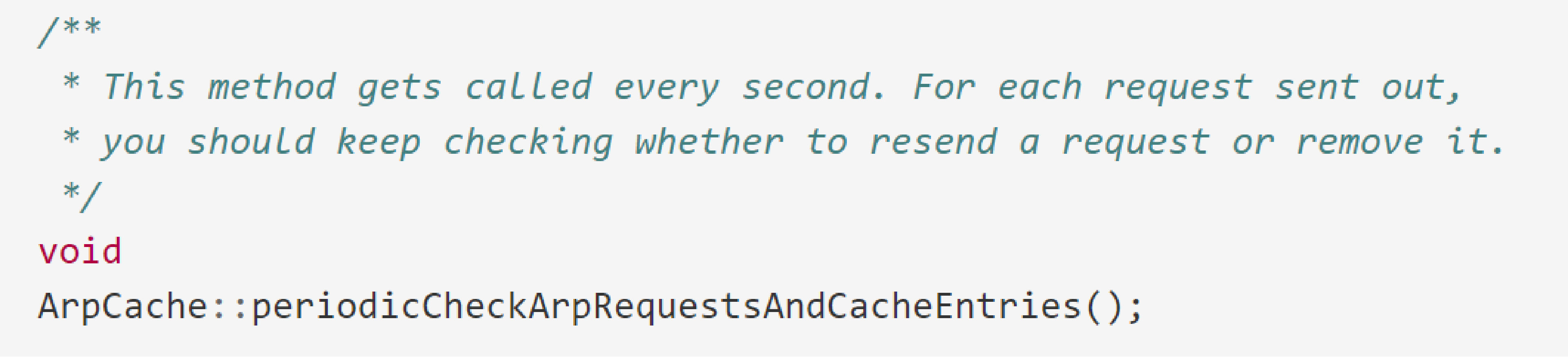
- IMPLEMENTED
- Method to send raw Ethernet frames (simple-router.hpp|cpp):



```
/**
 * Call this method to send packet \p packt from the router on interface \p outIface
 */
void
SimpleRouter::sendPacket(const Buffer& packet, const std::string& outIface);
```

Key Methods

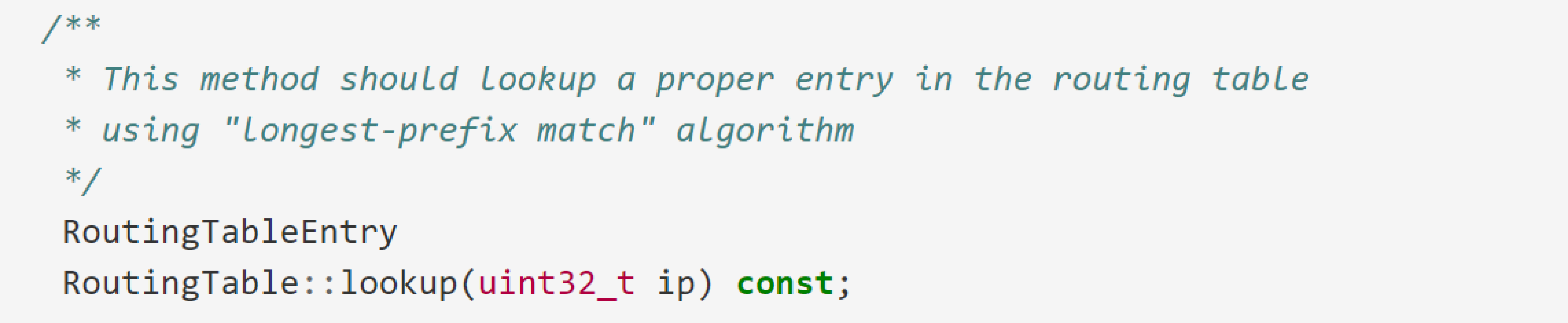
- NEED TO IMPLEMENT
- Method to handle ARP cache events (arp-cache.hpp|cpp):



```
/**
 * This method gets called every second. For each request sent out,
 * you should keep checking whether to resend a request or remove it.
 */
void
ArpCache::periodicCheckArpRequestsAndCacheEntries();
```

Key Methods

- NEED TO IMPLEMENT
- Method to lookup entry in the routing table (routing-table.hpp|cpp):



```
/**
 * This method should lookup a proper entry in the routing table
 * using "longest-prefix match" algorithm
 */
 RoutingTableEntry
 RoutingTable::lookup(uint32_t ip) const;
```

ISSUES TO NOTICE

环境配置

- 作业文档中提供了详细的环境配置指南

- 为了方便同学们快速配置环境，我们也提供了虚拟机文件
  - Windows VMware: https://cloud.tsinghua.edu.cn/d/05b4618dad6046b49438/
  - Mac(Apple Silicon) UTM: https://cloud.tsinghua.edu.cn/f/d09f8e35213540a5a2cf/

- 虚拟机的账号和密码都是 router

Some Important Issues

- Grading (up to 105%)
  - Router Implementation (85%=45%public + 40%private)
    - Ping tests (50%)
    - Traceroute tests (20%)
    - File Downloading tests (15%)
  - Project Report + Code Quality (20%)

- Individual work

Some Important Issues

- Submission
  - Source code ("*make tarball*")
  - Do not modify **existing data structures**
  - Report: no longer than THREE pages
- Evaluation
  - ping, traceroute, file downloading, details in project spec
  - Code quality
  - Project report

补交规则

- 正常提交: **在 DDL 前**提交作业正常计分
- 最迟提交期限和惩罚
  - 每次作业最迟在 DDL 后**一周（7天）内**提交，超出此期限**一律拒收**
  - 未超出此拒收期限的迟交作业得分***0.8**.
- 宽限期: 全部大作业共享 **7 天**宽限期。迟交**累计不超过**该期限的免于扣分惩罚

Some Important Issues

- Start up: today, after class
- Deadline: 12.21, 23:59 (3 weeks)
- Late Submission: 2025.12.22 ~ 2025.12.28, 23:59
  - Score * 0.8
  - No more submissions after 2025.12.28

Problems Emerged

- Improper TTL handling in traceroute / ping
  - when to do TTL--?
  - when to send ICMP time exceeded?
- Fail to maintain ARP cache entries / send ARP requests
- Imperfect longest prefix match
- Compilation & project structure problem
  - check before submission

Get to work

as soon as possible!

Q&A

Good luck

Acknowledgement

- This project is based on the CS118 class project by Alexander Afanasyev, UCLA.