# Lecture 20: Link Layer

COMP 332, Spring 2018

Victoria Manfredi

WESLEYAN
UNIVERSITY

**Acknowledgements**: materials adapted from Computer Networking: A Top Down Approach 7th edition: ©1996-2016, J.F Kurose and K.W. Ross, All Rights Reserved as well as from slides by Abraham Matta at Boston University, and some material from Computer Networks by Tannenbaum and Wetherall.

# Today

1. Announcements
   - hw7 written due today 11:59, hw7 programming due next Wed.11:59p
   - homework 8 out (no programming)
   - socket.inet_aton, socket.ntoa_inet()
   - what's a virtual machine?
2. Network programming
   - bit-wise operations in python
3. Link layer
   - overview
   - MAC addresses
   - Address Resolution Protocol (ARP)
   - switches

# NETWORK PROGRAMMING

# BIT-WISE OPERATIONS IN PYTHON

# Bit-wise operations on variables

x << y

- returns x with bits shifted to left by y places
  - new bits on right-hand-side are zeros
  - same as multiplying x by $2^y$

x >> y

- returns x with bits shifted to right by y places
  - same as dividing x by $2^y$

x & y

- does a bitwise and
  - each bit of output is 1 if corresponding bit of x AND of y is 1, otherwise 0

~ x

- returns complement of x
  - number you get by switching each 1 for 0 and each 0 for 1

E.g.,

- use to pack ip_version and ip header length into 8 bits

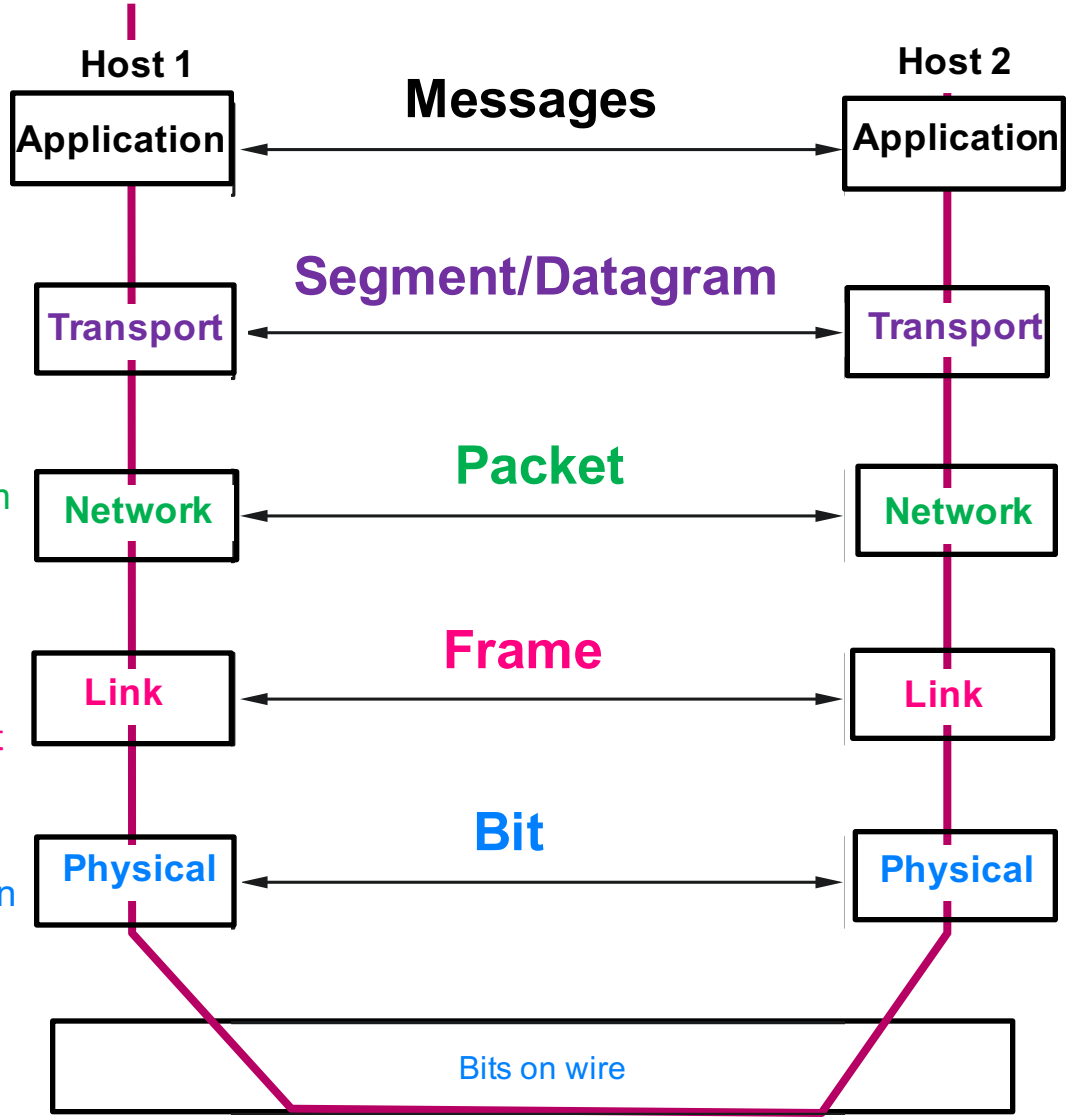
https://wiki.python.org/moin/BitwiseOperators
https://www.tutorialspoint.com/python3/bitwise_operators_example.htm

# Link Layer

## OVERVIEW

# Internet protocol stack

| Protocols | Service provided |
| --- | --- |
| • HTTP, DNS, DHCP, TLS, … | • Support network applications |
| • TCP, UDP, QUIC, … | • Deliver msgs to app endpoints, flow control, reliability |
| • IP, routing protocols (BGP, OSPF, RIP, …) | • Route segments from source to destination host (indirect) |
| • Ethernet, 802.11, … | • Move pkt over link from one host to next host (direct) |
| • Ethernet phy, 802.11 phy, … | • Move individual bits in frame from one host to next |

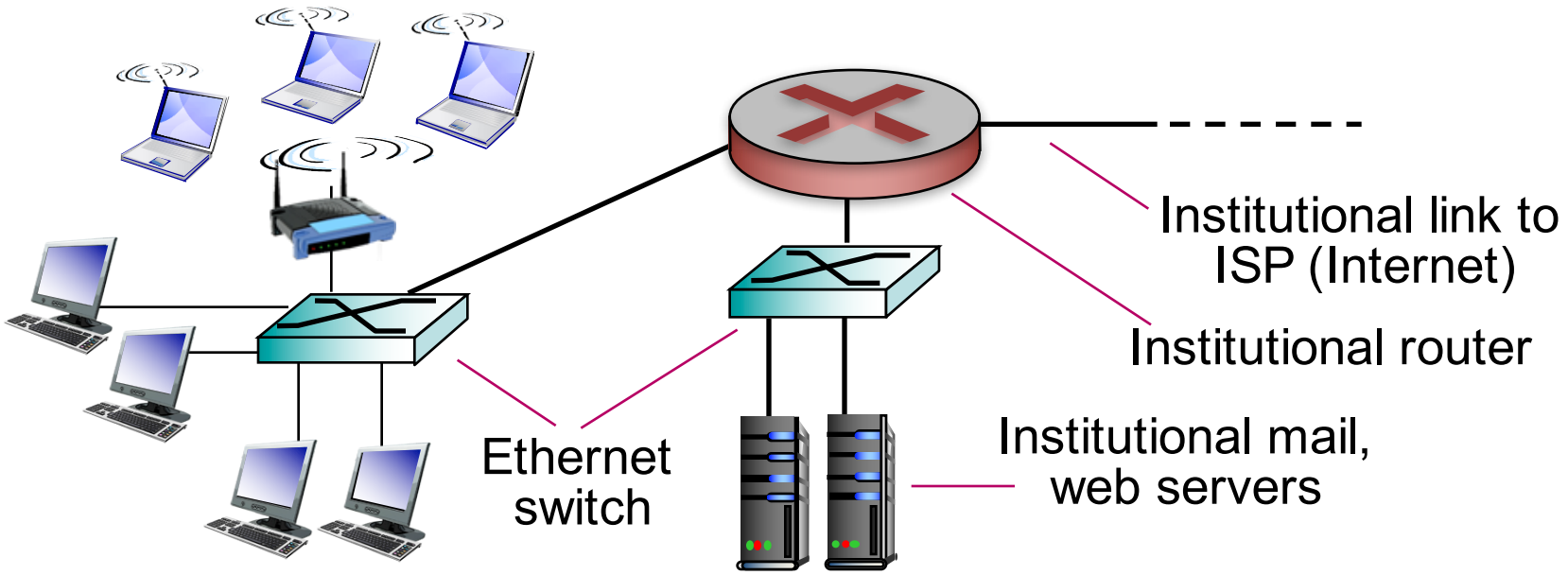


# Link layer

Divide packet up into frames, transfer frame across link

Link: communication channel that connects adjacent nodes

- wired vs. wireless
- local area network (LAN): frames broadcast to every device on LAN




Different link media have different characteristics, so will use different protocols with different services to transfer frame

# Link layer services

## Framing

- encapsulate packet into frame
- add header, trailer to detect start and end of frame

## Link access

- channel access if shared media like wireless link
  - How to share? Time/freq division, random access, …
- MAC addresses used in frame headers to identify src, dst
  - different from IP address, only used within network

## Reliable delivery between 2 end hosts of link

- like in transport layer
- low-bit error links rarely use: fiber, some twisted pair
- high bit error: wireless link

Q: why both link-level and end-end reliability?

What if error on last hop to dst?

# Other link layer services

## Flow control

- pacing between adjacent sending and receiving nodes

## Error detection

- errors caused by signal attenuation, noise, interference, …
- receiver detects presence of errors, e.g., via checksum
  - signals sender for retransmission or drops frame

## Error correction

- receiver identifies and corrects bit error(s)
- don't necessarily need to retransmit: instead use coding

## Half-duplex vs. full-duplex

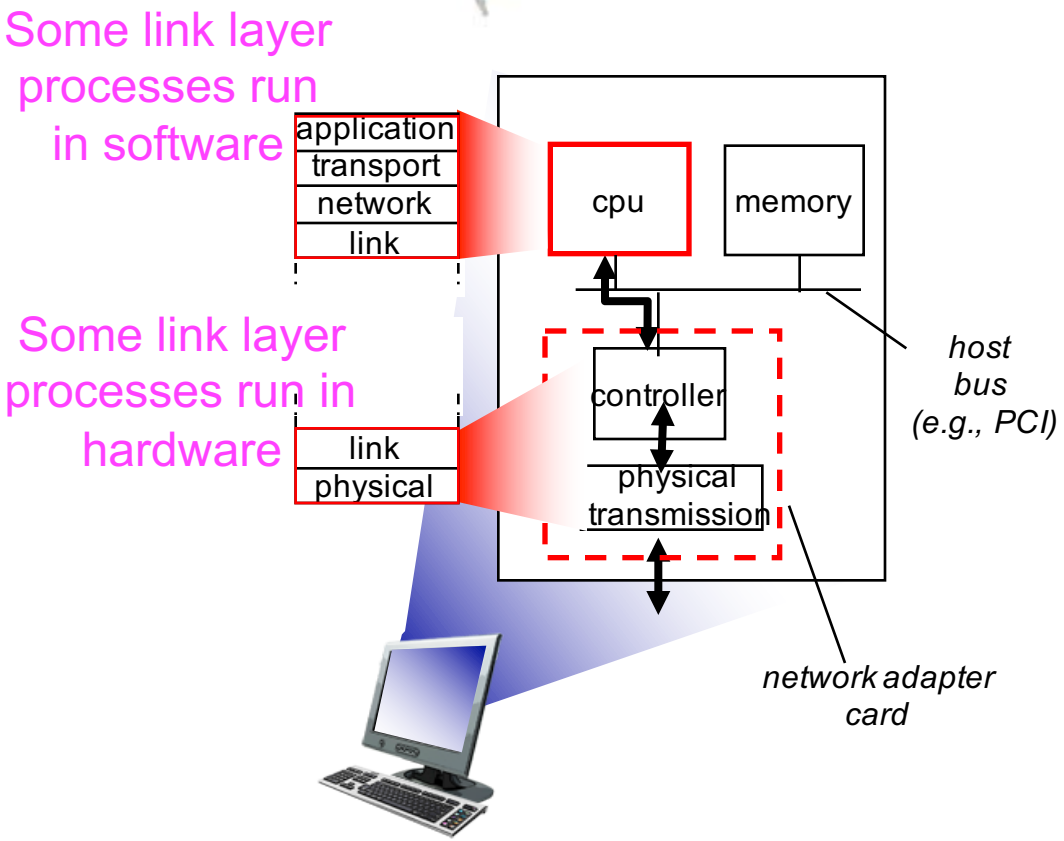
- half duplex
  - nodes at both ends of link can transmit, but not at same time
  - e.g., wireless link

# Where is the link layer implemented?

In network interface card (NIC) or on chip

- Ethernet card
- 802.11 card
- Ethernet chipset
- implements
  - link layer
  - physical layer: e.g., transmit radio wave

Attaches into system buses on host's system



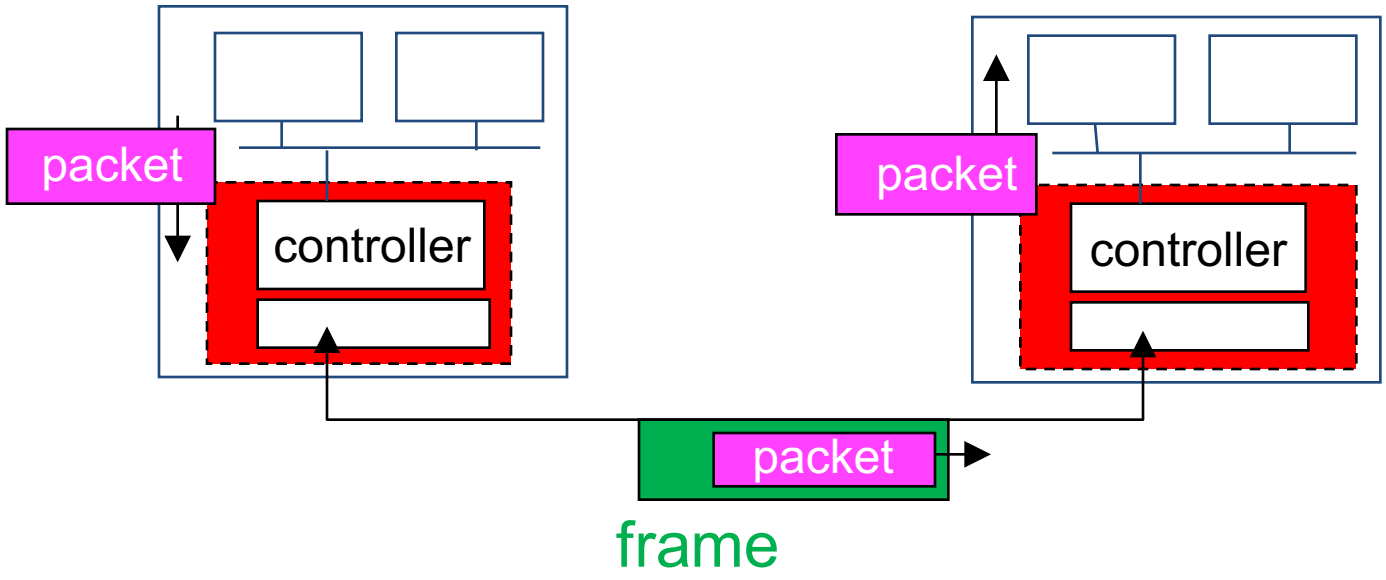
# Communication between network interfaces

## Sender

- encapsulates pkt in frame
- adds error checking bits, rdt, flow control, …

## Receiver

- looks for errors, rdt, flow control, …
- extracts pkt, passes to upper layer at receiving side



## Link Layer

# MAC ADDRESSES

# MAC addresses

Aka hardware, Ethernet, LAN, or physical address

## 32-bit IP address

- software address: network-layer address for interface
- used for layer 3 (network layer) forwarding

## 48-bit MAC address

- hardware address: link-layer address for interface
- used for layer 2 (link layer) forwarding
  - to get frame from one interface to another physically-connected interface
- burned in NIC read only memory, also sometimes software settable
- e.g., 1A-2F-BB-76-09-AD

# Why both MAC and IP addresses?

IP address: like postal address

- hierarchical address: not portable
- changes with location
  - address depends on IP subnet to which node is attached

MAC address: like SSN #

- flat address: portable
- does not change with location
  - can move LAN card from one LAN to another

LANs designed for arbitrary network layer protocol

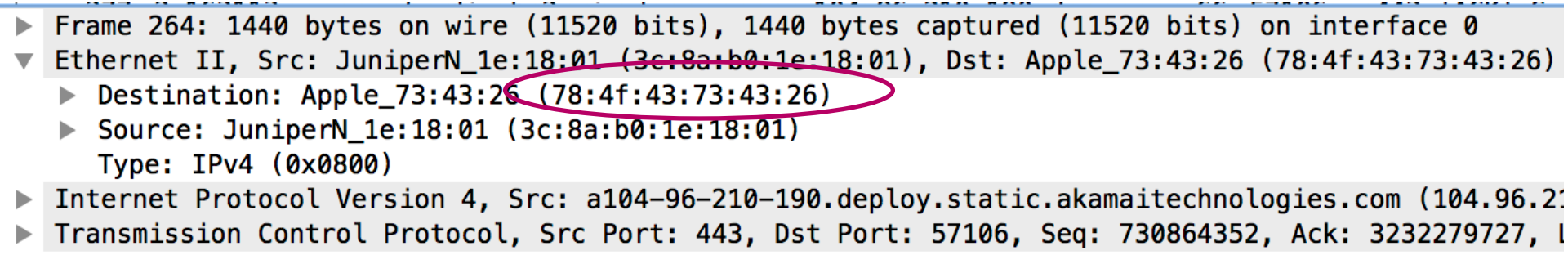
- not just IP

Don't want to pass frame up to network layer for ever frame

- faster, even if on same LAN to not go up to network layer

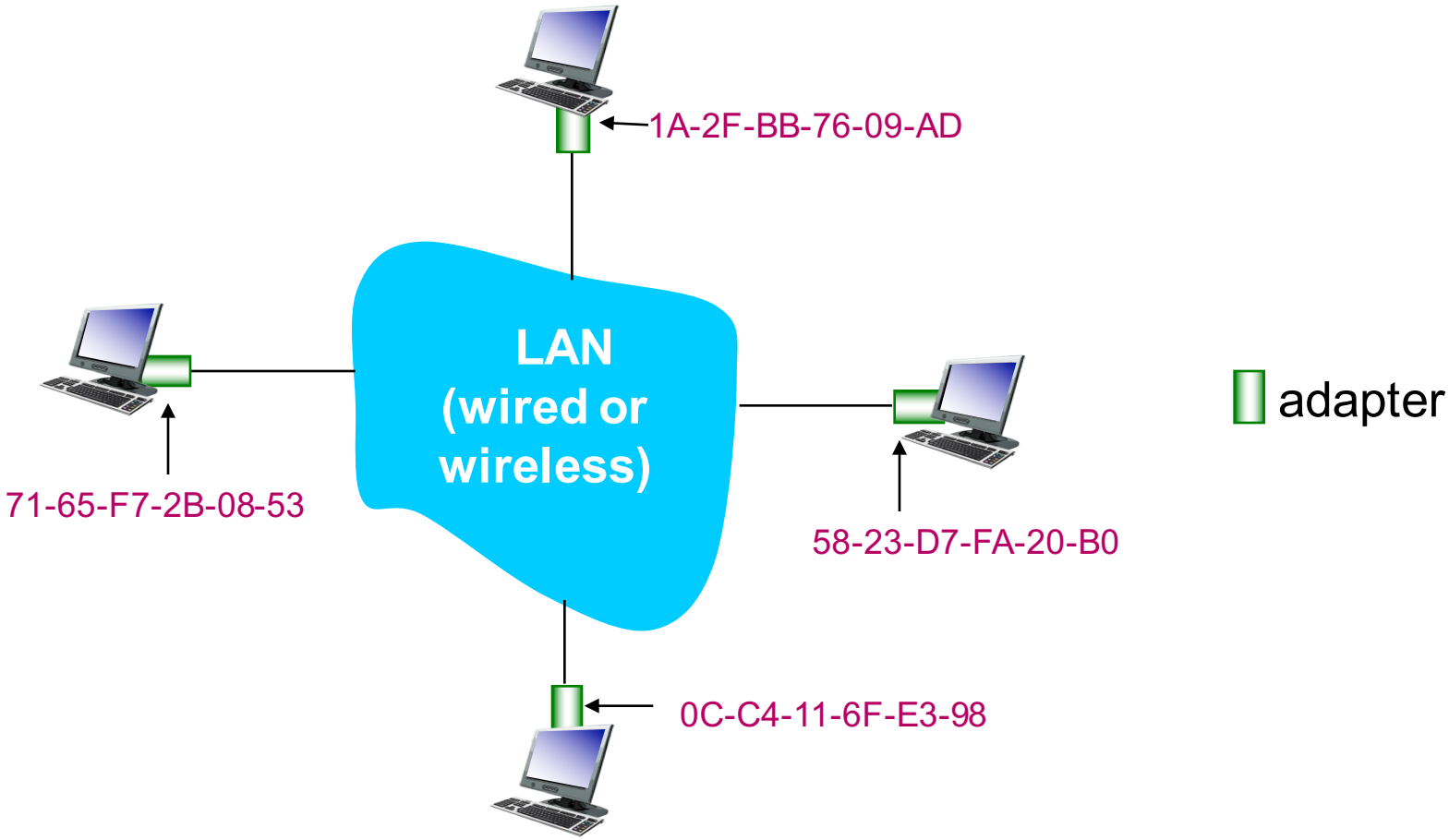
# Your MAC address

```
> ifconfig
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 16384
        options=1203<RXCSUM,TXCSUM,TXSTATUS,SW_TIMESTAMP>
        inet 127.0.0.1 netmask 0xff000000
        inet6 ::1 prefixlen 128
        inet6 fe80::1%lo0 prefixlen 64 scopeid 0x1
        nd6 options=201<PERFORMNUD,DAD>
gif0: flags=8010<POINTOPOINT,MULTICAST> mtu 1280
stf0: flags=0<> mtu 1280
en0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        ether 78:4f:43:73:43:26
        inet6 fe80::1c8a:4bcb:b52d:9d1d%en0 prefixlen 64 secured scopeid 0x5
        inet 129.133.187.174 netmask 0xfffff000 broadcast 129.133.191.255
```



# LAN addresses and ARP

Each adapter on LAN has unique LAN address

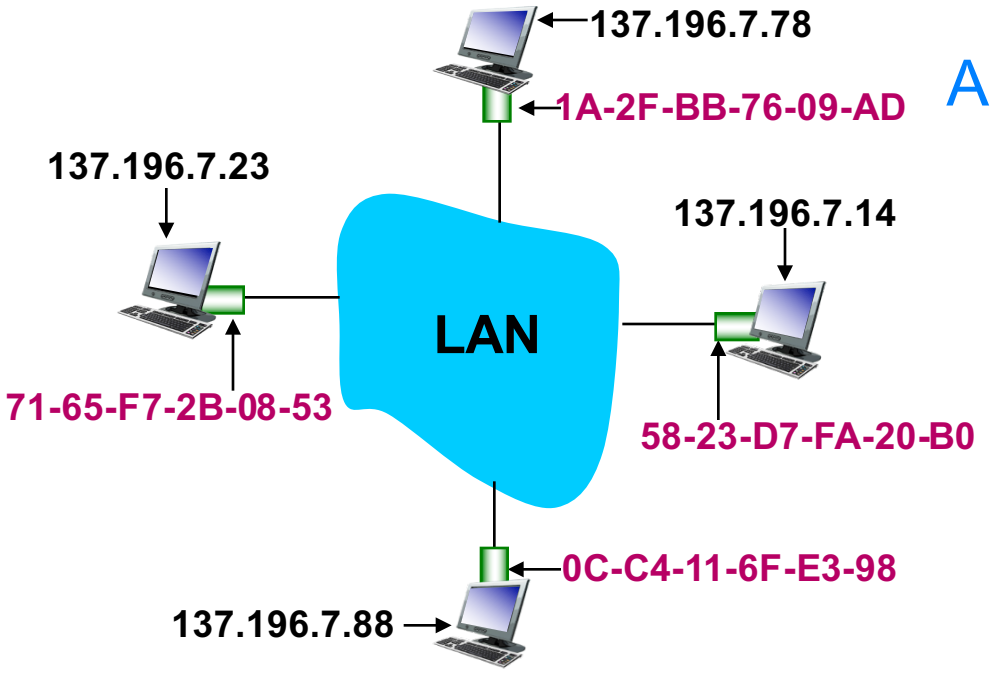


## Link Layer

# ADDRESS RESOLUTION PROTOCOL

# Address Resolution Protocol (ARP)

Link layer protocol that translates between MAC and IP addr




ARP table

- every IP device on LAN has table
- <IP addr; MAC addr; TTL>
- TTL (Time To Live)
  - time after which addr mapping forgotten
  - typically 20 min

# ARP in wireshark

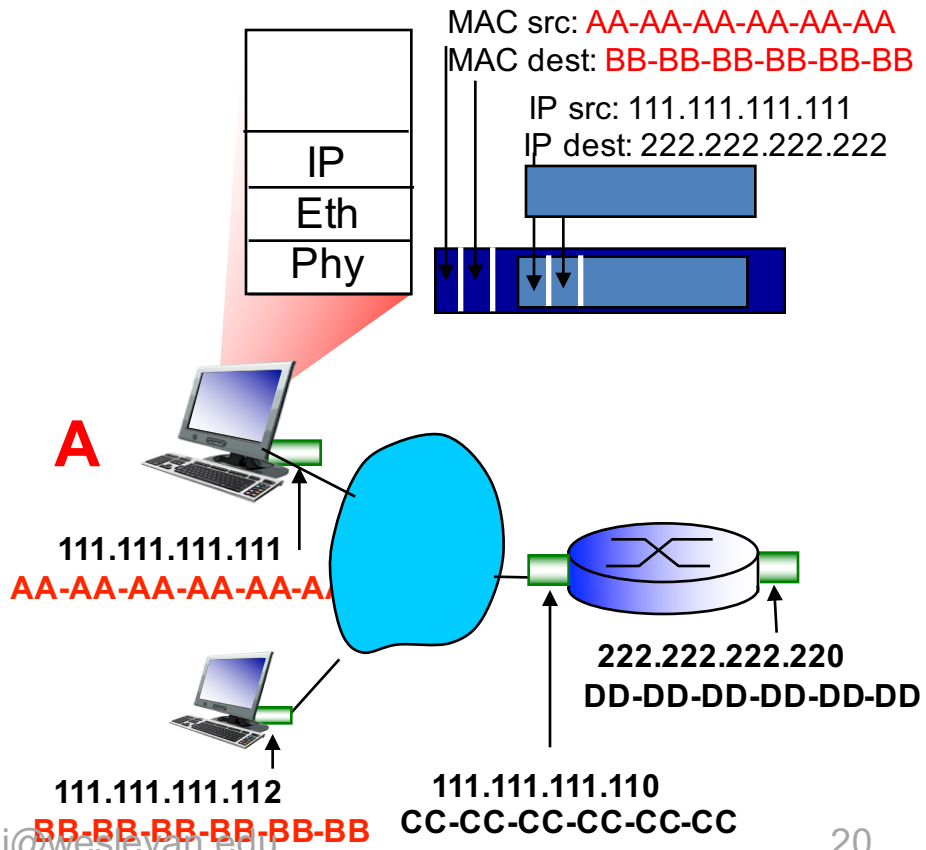
| No. | Time | Source | Destination | Length | Info |
|---|---|---|---|---|---|
| 5406 | 48.129810 | Apple_73:43:26 | 6e:57:ca:90:05:64 | 42 | Who has 172.20.10.1? Tell 172.20.10.11 |
| 5410 | 48.153207 | Apple_73:43:26 | 6e:57:ca:90:05:64 | 42 | Who has 172.20.10.1? Tell 172.20.10.11 |
| 5413 | 48.193996 | Apple_73:43:26 | 6e:57:ca:90:05:64 | 42 | Who has 172.20.10.1? Tell 172.20.10.11 |
| 5416 | 48.277611 | Apple_73:43:26 | 6e:57:ca:90:05:64 | 42 | Who has 172.20.10.1? Tell 172.20.10.11 |
| 5417 | 48.280822 | 6e:57:ca:90:05:64 | Apple_73:43:26 | 42 | 172.20.10.1 is at 6e:57:ca:90:05:64 |
| 5418 | 48.281053 | Apple_73:43:26 | Broadcast | 42 | Gratuitous ARP for 172.20.10.11 (Request) |
| 5423 | 48.376210 | Apple_73:43:26 | Broadcast | 42 | Who has 172.20.10.1? Tell 172.20.10.11 |
| 5424 | 48.377694 | 6e:57:ca:90:05:64 | Apple_73:43:26 | 42 | 172.20.10.1 is at 6e:57:ca:90:05:64 |
| 5661 | 51.723958 | vmanfredis-MacBook-Pro-2.l… | Broadcast | 42 | Who has 172.20.10.1? Tell 172.20.10.11 |
| 5662 | 52.043516 | vmanfredis-MacBook-Pro-2.l… | Broadcast | 42 | Gratuitous ARP for 172.20.10.11 (Request) |
| 5696 | 52.217609 | 6e:57:ca:90:05:64 | vmanfredis-MacBoo… | 42 | 172.20.10.1 is at 6e:57:ca:90:05:64 |
| 5721 | 52.367215 | vmanfredis-MacBook-Pro-2.l… | Broadcast | 42 | Who has 172.20.10.1? Tell 172.20.10.11 |
| 5802 | 52.483589 | 6e:57:ca:90:05:64 | vmanfredis-MacBoo… | 42 | 172.20.10.1 is at 6e:57:ca:90:05:64 |

```
▶ Frame 5406: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0
▼ Ethernet II, Src: vmanfredis-MacBook-Pro-2.local (78:4f:43:73:43:26), Dst: 6e:57:ca:90:05:64 (6e:57:ca:90:05:64)
  ▶ Destination: 6e:57:ca:90:05:64 (6e:57:ca:90:05:64)
  ▶ Source: vmanfredis-MacBook-Pro-2.local (78:4f:43:73:43:26)
    Type: ARP (0x0806)
▼ Address Resolution Protocol (request)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (1)
    Sender MAC address: vmanfredis-MacBook-Pro-2.local (78:4f:43:73:43:26)
    Sender IP address: vmanfredis-MacBook-Pro-2.local (172.20.10.11)
    Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
    Target IP address: 172.20.10.1 (172.20.10.1)
```

# Forwarding within same LAN

A wants to send pkt to B but B's MAC addr not in A's ARP table

1. A broadcasts ARP query containing B's IP addr
   - destination MAC addr
     - FF-FF-FF-FF-FF-FF
   - all nodes on LAN receive query
2. B receives ARP query
   - replies to A with its MAC addr
   - frame sent to A's MAC addr
3. A caches IP,MAC addr pair
   - until TTL expires



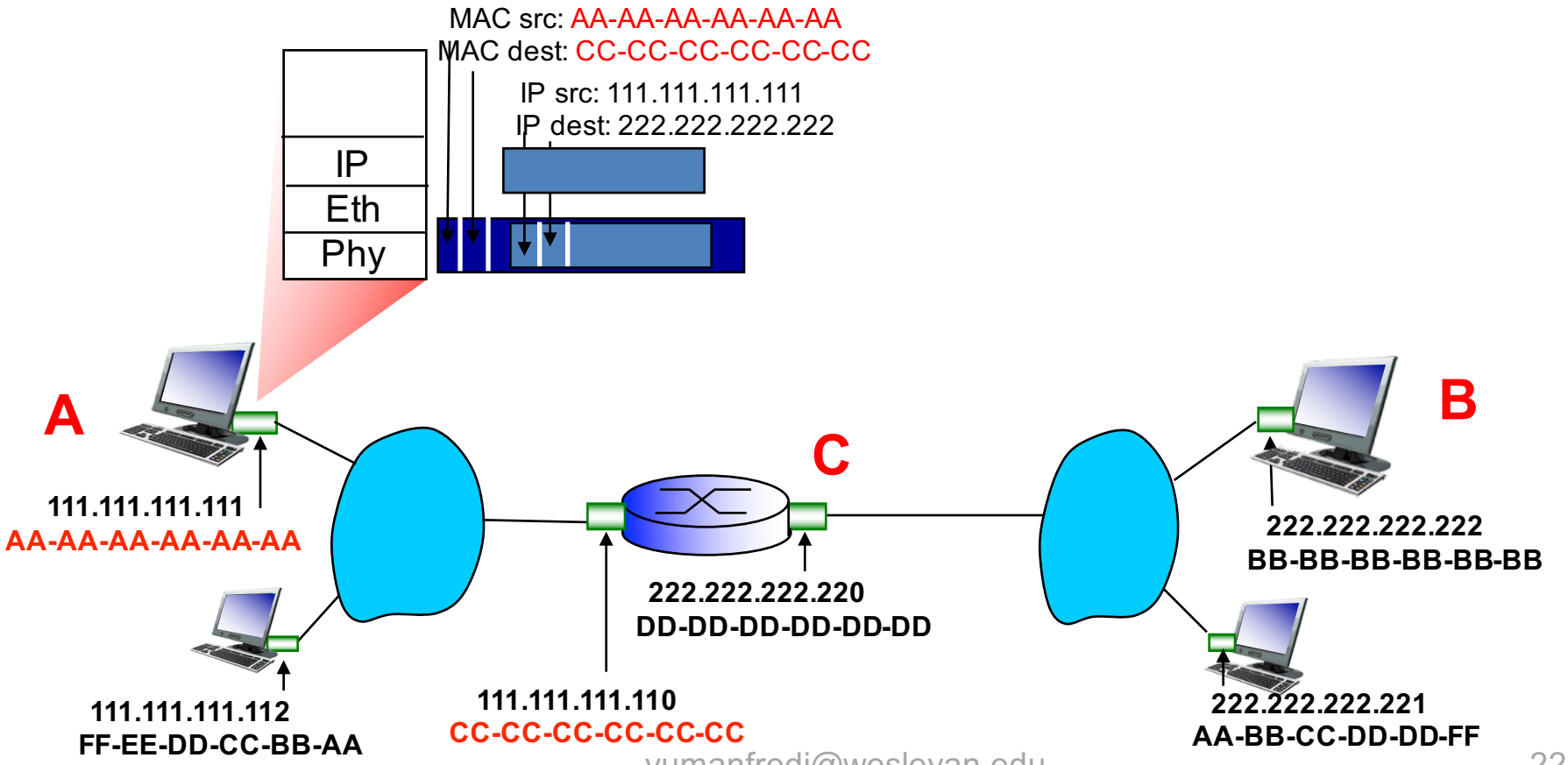
# Routing to another LAN

Work through example on board

# Routing to another LAN

## Send pkt from A to B via gateway router C

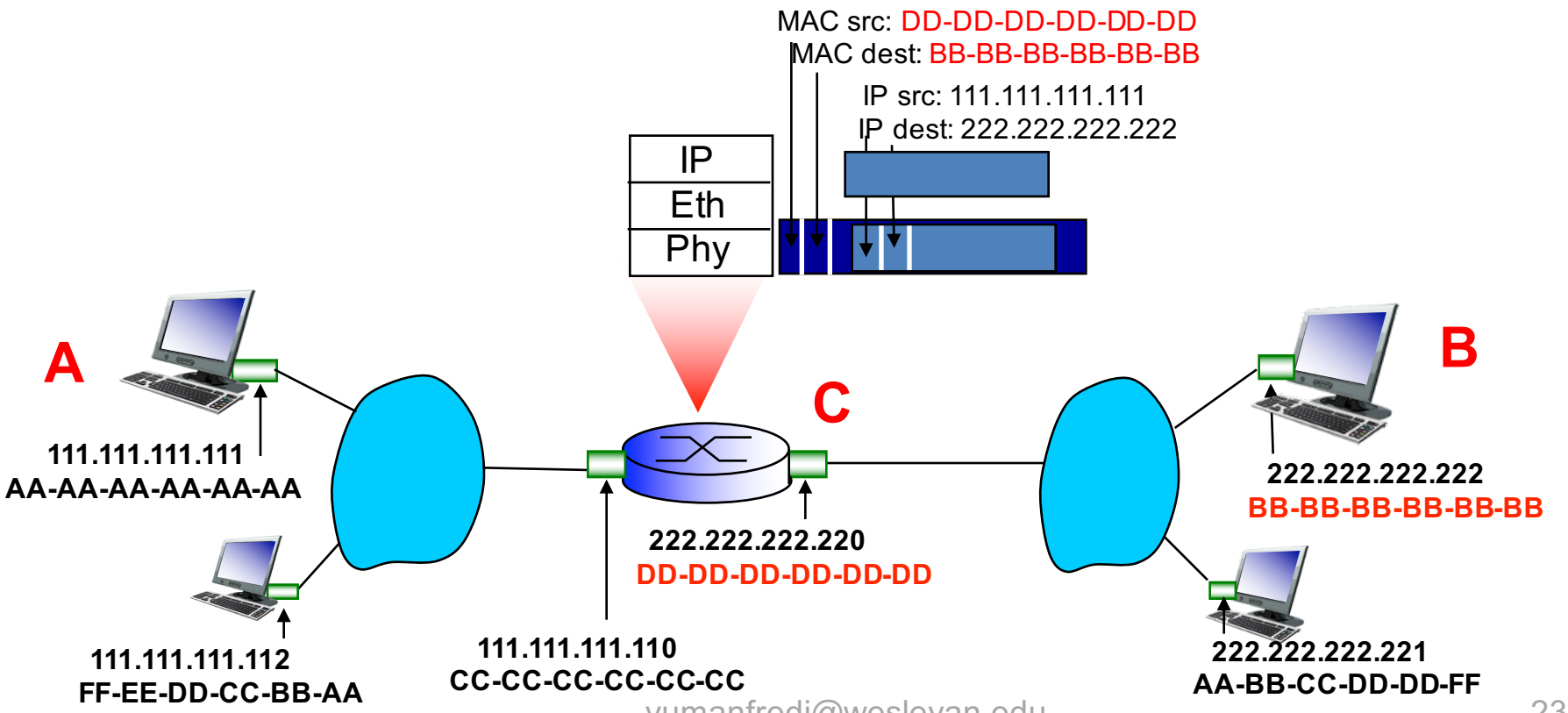
– assume A knows B's IP addr, C's IP addr, C's MAC addr



# Routing to another LAN

## Send pkt from A to B via gateway router C

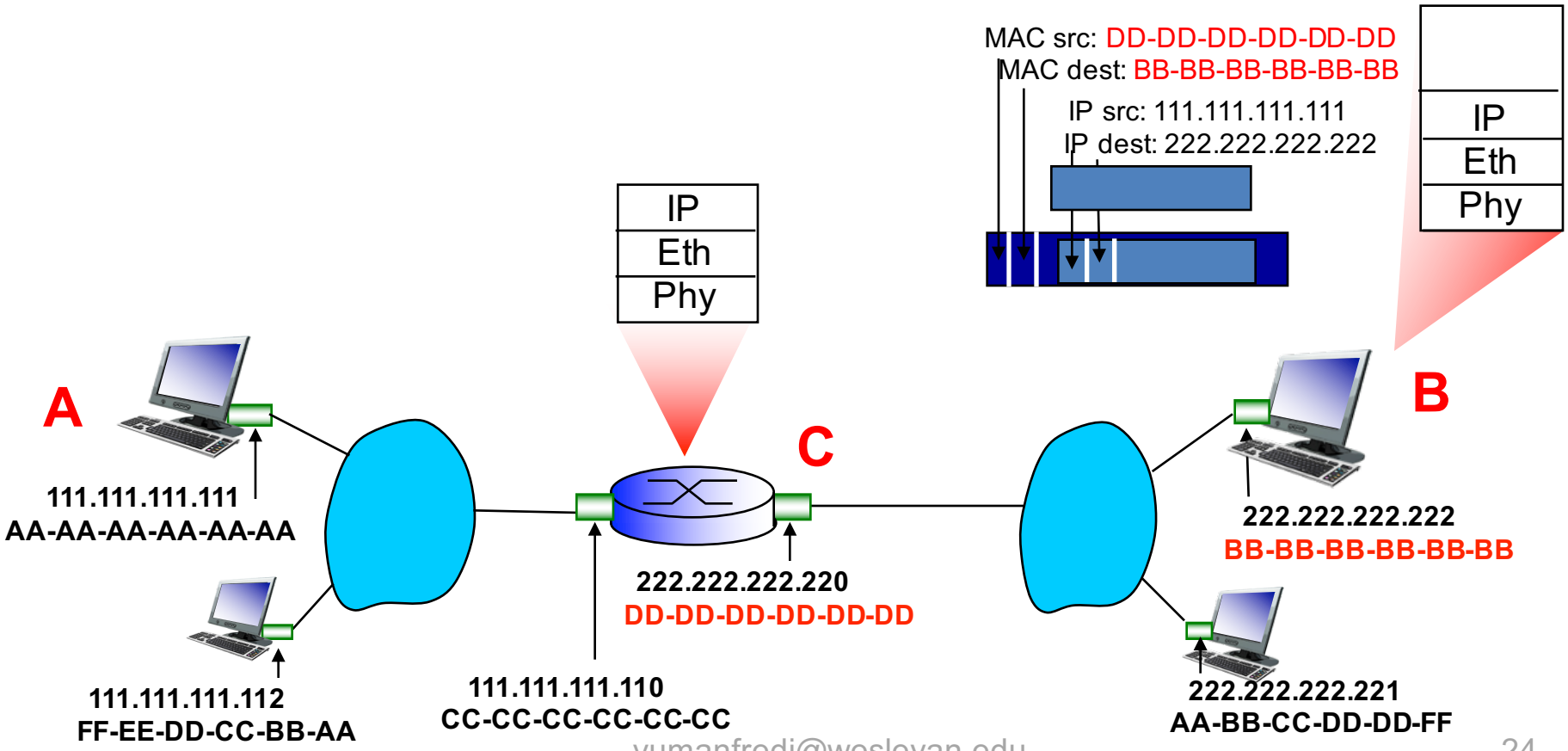
– assume A knows B's IP addr, C's IP addr, C's MAC addr R



# Routing to another LAN

## Send pkt from A to B via gateway router C

– assume A knows B's IP addr, C's IP addr, C's MAC addr R



# Link Layer

# SWITCHES

# Ethernet switch

Ethernet: dominant wired link layer protocol

Link-layer device store and forward Ethernet frames

- examine incoming frame's MAC address
- selectively forward frame to one-or-more outgoing links

Transparent

- hosts are unaware of presence of switches

Plug-and-play, self-learning

- switches do not need to be configured

Aside

- you'll see Ethernet listed as link layer protocol when you use wireless is because of a quirk of wireshark, which we won't get into

# Switches vs. routers

## Both are store-and-forward

- routers
  - data plane only examines network-layer headers
  - control plane (BGP) may look at app layer
- switches
  - examine link-layer headers

## Both have forwarding tables

- routers
  - compute tables using routing algorithms, IP addr
- switches
  - learn forwarding table using flooding, MAC addr