CSC361

Computer Networking

Mantis Cheng

Dept of Computer Science

Unit 4

Network & Link Layers

Important Concepts

- Network Interfaces
- IP Addressing
- Subnetting and CIDR Addressing
- Routers and Switches
- Network Address Translation (NAT)
- NAT and Its Implications
- DHCP

What We Learned So Far

- Internet uses IP addresses for delivering/routing packets.
- IP layer is the **glue** that binds all heterogenous networka into the Internet.
- Internet applications use TCP/UDP for end-to-end communication.
- DNS provides an application level service that maps hostnames to IP addresses.
- IP uses **protocol IDs**, and TCP/UDP uses **port numbers** to demultiplex to upper layers.

Physical Layer Principles (23:02)

Summary

- Ethernet is by-far the most common networking technology used today.
- Applicattion data is embedded as TCP **segments**; TCP segments are embedded as IP **datagrams**; and IP datagrams are embedded as ethernet **frames**. (This is known as **encapsulation**.)
- In the 4-layer Internet model, **Link** layer is the lowest physical layer.
- Ethernet is designed for sharing a **local area network** (LAN) medium.

Summary

- Medium Access Control (MAC) addresses identify nodes on a shared LAN.
- CSMA/CD (**Carrier-Sensing** and **Collision Detection**) is implemented in first Ethernet standard.

802.3 Ethernet packet and frame structure							
Preamble	**Start of frame delimiter**	**MAC destination**	**MAC source**	**802.1Q tag (optional)**	**Ethertype (Ethernet II) or length (IEEE 802.3)**	**Payload**	**Frame check sequence (32-bit CRC)**
7 octets	1 octet	6 octets	6 octets	(4 octets)	2 octets	46-1500 octets	4 octets

```
▾ Ethernet II, Src: Apple_cf:c5:57 (14:10:9f:cf:c5:57), Dst: Apple_d2:38:e3 (6c:70:9f:d2:38:e3)
  ▸ Destination: Apple_d2:38:e3 (6c:70:9f:d2:38:e3)
  ▸ Source: Apple_cf:c5:57 (14:10:9f:cf:c5:57)
    Type: IPv4 (0x0800)

0000  6c 70 9f d2 38 e3 14 10  9f cf c5 57 08 00 45 00   lp··8···  ···W··E·
0010  00 40 86 f1 40 00 40 06  58 72 0a 00 01 07 8e 68   ·@··@·@·  Xr·····h
0020  c1 e5 de 07 00 50 ab cb  ce 29 00 00 00 00 b0 02   ·····P··  ·)······
0030  ff ff 6b d5 00 00 02 04  05 b4 01 03 03 05 01 01   ··k·····  ········
0040  08 0a 70 57 a7 2e 00 00  00 00 04 02 00 00         ··pW·.··  ······
```

Ethernet (19:40)

(Ethernet Switch)

Summary

- A Ethernet (IEEE 802.3) **frame** consists of: `Preamble`, `SOF`, `Destination Address`, `Source Address`, `Data`, `CRC`.
- An Ethernet address is **48-bit** (6 bytes).
- CRC is an error-detection-correction check.
- A **hub** is a shared **dumb** electrical repeater.
- Transmission delay = l/r; propagation delay = d/s. Thus, $l/r > 2 * d/s$ means transmission delay must be **twice** as large as propagation delay. Why?

Summary (continued)

- Ethernet max. distance is 100 meters, max. payload is 1500 bytes.
- Transmission delay: $1500 * 8/100 * 10^6$ sec = 120 μsec.
- Propagation delay: $2 * 100/2 * 10^8$ sec = 1 μsec.
- Ethernet switch is a **smart** hub, which partitions **collision** domain.
- A switch/bridge uses memory buffers to allow ethernet frames to be sent on separate links concurrently. It is **not** a router!

Summary (continued)

- A Ethernet switch learns MAC addresses **incrementally** over time from each of its link.
- When a frame is sent to a **known** MAC address, a switch will forward to to the correct link.
- If a MAC address is **unknown**, a switch will **broadcast** to all links to locate the correct node.
- Ethernet switches are **intelligent** hubs that learn and forward frames to the appropriate MAC address incrementally.

IP Header Format

<table>
<caption>IPv4 Header Format</caption>
<tr><th>Offsets</th><th>Octet</th><th colspan="8">0</th><th colspan="8">1</th><th colspan="8">2</th><th colspan="8">3</th></tr>
<tr><th>Octet</th><th>Bit</th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th><th>16</th><th>17</th><th>18</th><th>19</th><th>20</th><th>21</th><th>22</th><th>23</th><th>24</th><th>25</th><th>26</th><th>27</th><th>28</th><th>29</th><th>30</th><th>31</th></tr>
<tr><th>0</th><th>0</th><td colspan="4">Version</td><td colspan="4">IHL</td><td colspan="6">DSCP</td><td colspan="2">ECN</td><td colspan="16">Total Length</td></tr>
<tr><th>4</th><th>32</th><td colspan="16">Identification</td><td colspan="3">Flags</td><td colspan="13">Fragment Offset</td></tr>
<tr><th>8</th><th>64</th><td colspan="8">Time To Live</td><td colspan="8">Protocol</td><td colspan="16">Header Checksum</td></tr>
<tr><th>12</th><th>96</th><td colspan="32">Source IP Address</td></tr>
<tr><th>16</th><th>128</th><td colspan="32">Destination IP Address</td></tr>
<tr><th>20</th><th>160</th><td colspan="32" rowspan="5">Options (if IHL > 5)</td></tr>
<tr><th>24</th><th>192</th></tr>
<tr><th>28</th><th>224</th></tr>
<tr><th>32</th><th>256</th></tr>
</table>

(Note: TCP/IP uses Big Endian. MSB is bit 0; LSB is bit 31.)

Wireshark IP Header

```
▾ Internet Protocol Version 4, Src: 10.0.1.7, Dst: 142.104.193.229
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▸ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 64
    Identification: 0x86f1 (34545)
  ▸ Flags: 0x4000, Don't fragment
    Time to live: 64
    Protocol: TCP (6)
    Header checksum: 0x5872 [validation disabled]
    [Header checksum status: Unverified]
    Source: 10.0.1.7
    Destination: 142.104.193.229
0000  6c 70 9f d2 38 e3 14 10  9f cf c5 57 08 00 45 00   lp··8···  ···W··E·
0010  00 40 86 f1 40 00 40 06  58 72 0a 00 01 07 8e 68   ·@··@·@·  Xr·····h
0020  c1 e5 de 07 00 50 ab cb  ce 29 00 00 00 00 b0 02   ·····P··  ·)······
0030  ff ff 6b d5 00 00 02 04  05 b4 01 03 03 05 01 01   ··k·····  ········
0040  08 0a 70 57 a7 2e 00 00  00 00 04 02 00 00         ··pW·.··  ······
```

Demo "web.uvic.ca.pcap".

IP Addresses & Network ID

- An IP address consists of two parts: (n,h), where $n + h = 32$ bits, and n is `network id`, h is `host id`.
- Within a network n, there are $2^h - 2$ usable `host id`s, where all `0`s and all `1` are reserved.
- A router typically has at least two network interfaces, one **private** and the other **external**.
- `network id` is used by a router to look up a **forwarding table** to decide which external interface to deliver a packet.

The IPv4 Addresses (3:12)

Summary

- An IP packet contains a **source** and a **destination** IP addresses.
- IPv4 uses 32-bit IP addresses, written as `171.64.64.72`, dotted quad decimal notation.
- A **netmark** (up to 32 bits) applies to an IP address to select its **network id** portion; the remaining bits are known as **host id**.
- For example, a netmask `255.255.255.0` selects the first 24 bits of an IP address as the network id.

Summary (continued)

- Two IP addresses `P` and `Q` are on the **same** network if `P & netmask` == `Q & netmask`.
- A packet not on the same network must be routed to another network through a router.
- Traditionally, IP addresses are classified into classes: `A`, `B`, `C`, `D` and `E`.

Class-based Addressing

Class	Byte0	Byte1	Byte2	Byte3
A	0xxxxxxx	hostid	hostid	hostid
B	10xxxxxx	xxxxxxxx	hostid	hostid
C	110xxxxx	xxxxxxxx	xxxxxxxx	hostid
D	1110xxxx	**multicast**		
E	11110xxx	**reserved**		

(Note: xxxxxxx is the network id.)

[IP Addressing and Subnetting 24:00]

(first 24 min. Introduction to Subnetting)

Network ID	Host ID	Remarks
x	all `0`s	network id
x	all `1`s	direct broadcast
all `1`s	all `1`s	limited broadcast
all `0`s	all `0`s	**this** host
all `0`s	h	h is a local host
127.0.0	h	loopback
10.0.0	h	private
172.[16..32]	h	private
192.168.[0..255]	h	private

[IP Addressing and Subnetting 24:00-46:00]

(Hierarchical Subnetting and CIDR at 39:00)

Network IDs and Subnet Masks (20:31)

(CIDR Addressing Examples)

Classless Inter-Domain Routing (CIDR) Addressing

- `u.x.y.z/n` where `n` is the prefix of `1`s in a **netmask**.
- What is netid for `172.10.85.60/22`? `127.10.84.0`.
- What is max. number of hostid excluding all `0`s and all `1`s? `1024`-`2`=`1022` (`172.10.84.1` to `172.10.87.254`)
- What is the broadcast address? `127.10.87.255`.

Why CIDR Addressing?

- The **old** class-based IP addresses and **netmask** were replaced by CIDR (RFC4632) in 1993, to simplify router's forwarding tables management, and to slow down the exhaustion of IPv4 addresses.
- The network id **prefix** defined by `/n` is used by routers for *pattern matching* in the forwarding tables; conventionally, the **longest prefix match** is chosen for an outgoing link.

Subnetting

- An organization, given an n bit `network id`, may have too many `host id`s, e.g. `142.14.192.2/16` has $2^{16} - 2$ hosts.
- So, it may subdivide its network into several subnets, e.g. `142.14.192.2/24` where `192` is a `subnet id` and `2` is a `host id` within the subnet.
- How many subnets?
- Subnetting is totally **invisible** to routers outside the organization. e.g., `142.14` is still its `network id`.

Packet Forwarding Route Prefix Length (5:53)

(summarizes what longest prefix match means)

Routers vs Switches (17:15)

(how a packet switch works and how longest prefix match works)

Summary

- An ethernet switch is a Link Layer switch that uses MAC addresses to forward frames.
- An Internet router is a Network Layer packet switch that uses IP address to forward packets.
- **Longest prefix match** means matching the **most specific** network ids in order to locate the forwarding link.

NAT (12:19)

RFC3022

Summary

- What is Network Address Translation? Why?
- A NAT hides a **private** (internal) network behinds a **public** (external) IP address.
- All WiFi routers are NATs; a NAT WiFi router allows many internal PCs shared an IP address provided by an ISP.
- An internal private network can use the same IP address range, e.g., `10.0.1.0/8`.
- A NAT extends a typical Network Layer (IP) router with **port forwarding**.

Summary (continued)

- IP (Network) Layer doesn't use port numbers!
- With NAT routers, port numbers are dynamically created to map internal sockets to a new port number, e.g., `10.0.1.5:10000` to `x` where `x` is a new port number only the NAT router knows.
- When an external packet refers to port number `x`, it is then forwarded to `10.0.1.5:10000`.
- A NAT router maintains many **dynamically** created ports, one for each internal (private) socket.

Wireshark Demo (7:13)

(NAT inside `CSC361-VM`)

NATs Implications (14:47)

Summary

- NAT affects the behavoirs of many Internet applications.
- NAT violates the **End-to-End Principle** of Internet design philosophy; it reads the Transport Level port numbers and changes it when necessary.
- NAT allows internal machines to connect to external machines, but not the other way around; it prevents outside applications to connect to internal machine without explicit configurations.

Summary (continued)

- When one machine `A` is behind NAT, then the other machine `B` can use **reserved connection** using an open server for coordination.
- When both machines `A` and `B` are behind NATs, then we need a **relay** server to pass communication back-and-forth between `A` and `B`.
- NATs are here to stay because they add security and reuse IP addresses.
- With NATs, we **can't** add new Transport protocols because they already assume the existence of **port numbers** in TCP/UDP.

DHCP (12:24)

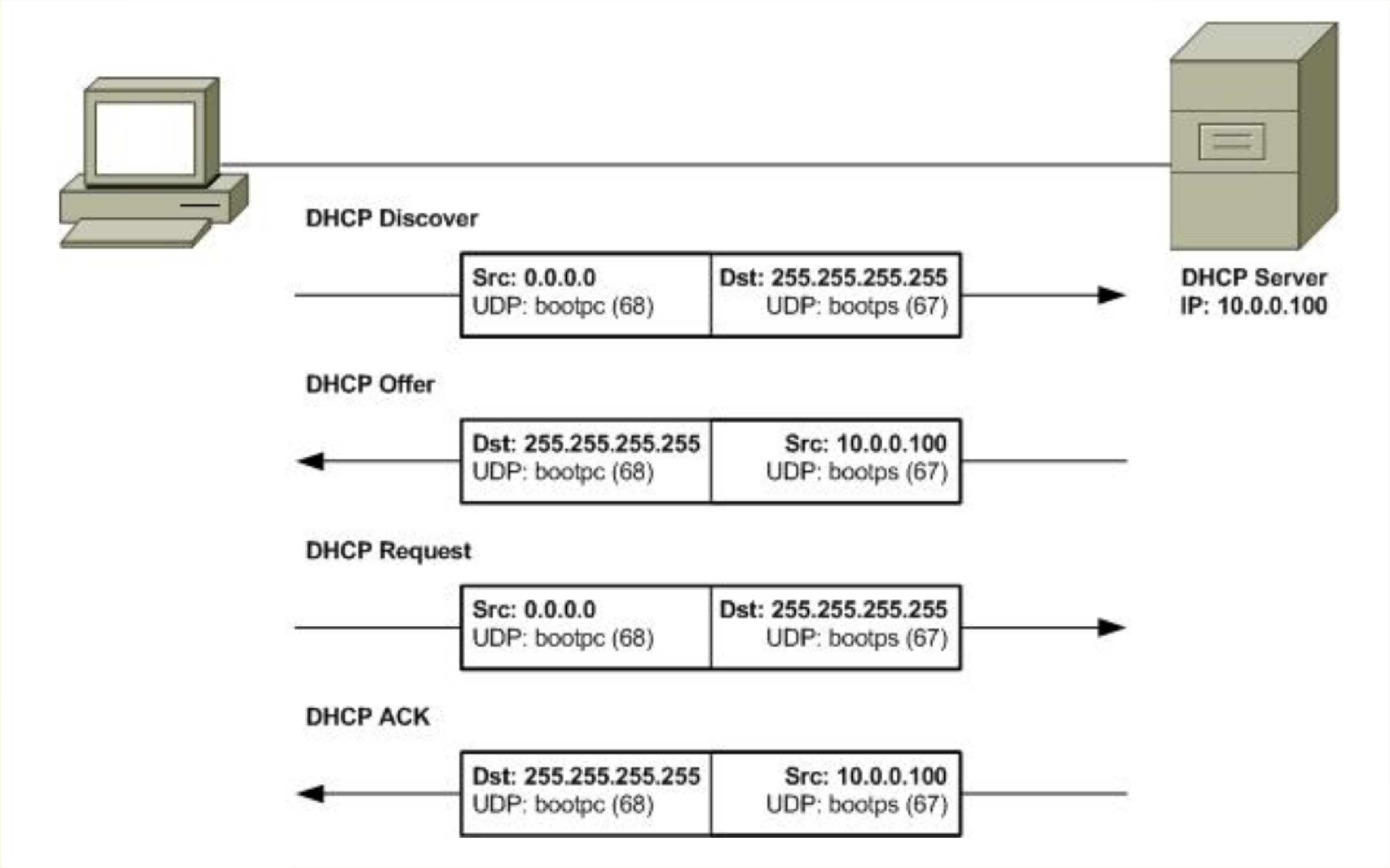
RFC2131

Summary

- Dynamic Host Configuration Protocol (DHCP) allows a machine joining a network and then acquiring an IP address **dynamically**.
- A newly joined machine needs to know its **IP address**, **subnet mask**, **gateway/router**, and also **DNS server**.
- DHCP is particularly necessary in a WiFi environment.

Summary (continued)

- A new client (`0.0.0.0:68`) **broadcasts** (`255.255.255.255:67`) `I need an IP address` using UDP.
- A DHCP server **offers** `Here is an IP address for you` via client's MAC address.
- A client **requests to take** the offered `new IP address` as its assigned IP address.
- A server **acknowledges** that it has now assigned the client's `new IP address`, together with all subnet masks, router, DNS servers, etc.


DHCP Discover
Src: 0.0.0.0
UDP: bootpc (68)
Dst: 255.255.255.255
UDP: bootps (67)
DHCP Server
IP: 10.0.0.100
DHCP Offer
Dst: 255.255.255.255
UDP: bootpc (68)
Src: 10.0.0.100
UDP: bootps (67)
DHCP Request
Src: 0.0.0.0
UDP: bootpc (68)
Dst: 255.255.255.255
UDP: bootps (67)
DHCP ACK
Dst: 255.255.255.255
UDP: bootpc (68)
Src: 10.0.0.100
UDP: bootps (67)

The End