Chapter 3: Review of Important Networking Concepts

Magda El Zarki
Dept. of CS
UC Irvine
elzarki@uci.edu
http://www.ics.uci.edu/~magda
Networking Concepts

- Protocol Architecture
- Protocol Layers
- Encapsulation
- Network Abstractions
Sending a packet from Argon to Neon

argon.tcpip-lab.edu  
"Argon"  
128.143.137.144

router137.tcpip-lab.edu  
"Router137"  
128.143.137.1

router71.tcpip-lab.edu  
"Router71"  
128.143.71.1

neon.tcpip-lab.edu  
"Neon"  
128.143.71.21

Ethernet Network

Router

Ethernet Network
DNS: The IP address of neon.tcpip-lab.edu is 128.143.71.21.

ARP: What is the MAC address of 128.143.137.1?

128.143.71.21 is not on my local network. Therefore, I need to send the packet to my default gateway with address 128.143.137.1.

128.143.71.21 is on my local network. Therefore, I can send the packet directly.
Communications Architecture

• The complexity of the communication task is reduced by using multiple protocol layers:
  • Each protocol is implemented independently
  • Each protocol is responsible for a specific subtask
  • Protocols are grouped in a hierarchy

• A structured set of protocols is called a communications architecture or protocol suite
The TCP/IP protocol suite is the protocol architecture of the Internet.

The TCP/IP suite has four layers: Application, Transport, Network, and Data Link Layer.

End systems (hosts) implement all four layers. Gateways (Routers) only have the bottom two layers.
Functions of the Layers

- **Data Link Layer:**
  - **Service:**
    - Reliable transfer of frames over a link
    - Media Access Control on a LAN
  - **Functions:**
    - Framing, media access control, error checking

- **Network Layer:**
  - **Service:** Move packets from source host to destination
  - **Functions:** Routing, addressing

- **Transport Layer:**
  - **Service:**
    - Delivery of data between hosts
  - **Functions:**
    - Connection establishment/termination, error control, flow control

- **Application Layer:**
  - **Service:** Application specific (delivery of email, retrieval of HTML documents, reliable transfer of file)
  - **Functions:** Application specific
The TCP/IP protocol stack does not define the lower layers of a complete protocol stack.
Assignment of Protocols to Layers

Network Layer
- Network Interface
  - ARP
  - Ethernet
  - DHCP
  - ICMP
  - IGMP

Transport Layer
- TCP
- UDP
  - OSI Protocol Stack
  - Application Layer
    - ping
    - HTTP
    - Telnet
    - FTP
    - DNS
    - SNMP
    - Telnet

Routing Protocols
- RIP
- PIM
- OSPF
Layered Communications

• An entity of a particular layer can only communicate with:
  1. a peer layer entity using a common protocol (Peer Protocol)
  2. adjacent layers to provide services and to receive services
Service Primitives

Communication services are invoked via function calls. The functions are called **service primitives**.
Service Primitives

Recall: A layer N+1 entity sees the lower layers only as a service provider.

N+1 Layer Entity --- N+1 Layer Peer Protocol --- N+1 Layer Entity

Request Delivery

Indicate Delivery

Service Provider
Service Access Points

- A service user accesses services of the service provider at Service Access Points (SAPs)

- A SAP has an address that uniquely identifies where the service can be accessed
Exchange of Data

- Assume a layer-N entity at A wants to send data to a layer-N peer entity to B
- The unit of data sent between peer entities is called a Protocol Data Unit (PDU)
- For now, let us think of a PDU as a single packet
  - What actually happens: Layer N passes the PDU to one of A’s SAPs at layer N-1
  - The layer N-1 entity (at A) then constructs its own PDU which it sends to the layer N-1 entity at B
  - Note: PDU at layer N-1 = Header + PDU at layer N
Exchange of Data

When passed to the SAP, the PDU is called a Service Data Unit (SDU) (Layer-N PDU = Layer-N-1 SDU)

SAPs
Layers in the Example

HTTP

TCP

IP

Ethernet

argon.tcpip-lab.edu
128.143.137.144

router71.tcpip-lab.edu
128.143.137.1
00:e0:f9:23:a8:20

router137.tcpip-lab.edu
128.143.71.1

neon.tcpip-lab.edu
128.143.71.21
Layers in the Example

Establish a connection to 128.143.71.21 at port 80
Open TCP connection to 128.143.71.21 port 80
Send IP datagram to 128.143.71.21
Send the datagram to 128.143.7.21
Frame is an IP datagram
Frame is an IP datagram
Send Ethernet frame to 00:e0:f9:23:a8:20
Send Ethernet frame to 00:20:af:03:98:28
Send Ethernet frame to 00:e0:f9:23:a8:20
Layers and Services

- **Service provided by TCP to HTTP:**
  - reliable transmission of data over a logical connection

- **Service provided by IP to TCP:**
  - unreliable transmission of IP datagrams across an IP network

- **Service provided by Ethernet to IP:**
  - transmission of a frame across an Ethernet segment

- **Other services:**
  - **DNS:** translation between domain names and IP addresses
  - **ARP:** Translation between IP addresses and MAC addresses
Encapsulation and Demultiplexing

- As data is moving down the protocol stack, each protocol is adding layer-specific control information.
Encapsulation and Demultiplexing in our Example

- Let us look in detail at the Ethernet frame between Argon and the Router, which contains the TCP connection request to Neon.

- This is the frame in hexadecimal notation.

```
00e0 f923 a820 00a0 2471 e444 0800 4500
002c 9d08 4000 8006 8bff 808f 8990 808f
4715 065b 0050 0009 465b 0000 0000 6002
2000 598e 0000 0204 05b4
```
Encapsulation and Demultiplexing

- Ethernet Header
- IP Header
- TCP Header
- Application data
- Ethernet Trailer

- Ethernet frame

- destination address
- source address
- type

6 bytes

4 bytes

CRC
Encapsulation and Demultiplexing: Ethernet Header

- **Ethernet Header**
- **IP Header**
- **TCP Header**
- **Application data**
- **Ethernet Trailer**

- **CRC**

- **00:e0:f9:23:a8:20**
- **0:a0:24:71:e4:44**
- **0x0800**

- **6 bytes**

- **4 bytes**

**Ethernet frame**
Encapsulation and Demultiplexing: IP Header

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>version (4 bits)</td>
<td>version of IP protocol</td>
</tr>
<tr>
<td>header length (16 bits)</td>
<td>length of IP header, excluding checksum</td>
</tr>
<tr>
<td>DS</td>
<td>Protocol type identifier</td>
</tr>
<tr>
<td>ECN</td>
<td>ECN (Experimental Control Field)</td>
</tr>
<tr>
<td>Total Length (in bytes) (16 bits)</td>
<td>length of the IP datagram, including header</td>
</tr>
<tr>
<td>Identification (16 bits)</td>
<td>ID used to correlate fragment packets</td>
</tr>
<tr>
<td>flags (3 bits)</td>
<td>flags for fragmentation, etc.</td>
</tr>
<tr>
<td>Fragment Offset (13 bits)</td>
<td>offset within the datagram where the fragment begins</td>
</tr>
<tr>
<td>TTL Time-to-Live (8 bits)</td>
<td>Time-to-Live (number of hops remaining)</td>
</tr>
<tr>
<td>Protocol (8 bits)</td>
<td>protocol number</td>
</tr>
<tr>
<td>Source IP address (32 bits)</td>
<td>source address of the datagram</td>
</tr>
<tr>
<td>Destination IP address (32 bits)</td>
<td>destination address of the datagram</td>
</tr>
<tr>
<td>Header Checksum (16 bits)</td>
<td>checksum of IP header fields</td>
</tr>
</tbody>
</table>
### Encapsulation and Demultiplexing: IP Header

#### Ethernet Frame

<table>
<thead>
<tr>
<th>Layer</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ethernet Header</td>
<td>Contains Ethernet frame structure</td>
</tr>
<tr>
<td>IP Header</td>
<td>Contains IP header information</td>
</tr>
<tr>
<td>TCP Header</td>
<td>Contains TCP header information</td>
</tr>
<tr>
<td>Application data</td>
<td>Contains application data</td>
</tr>
<tr>
<td>Ethernet Trailer</td>
<td>Contains Ethernet trailer information</td>
</tr>
</tbody>
</table>

#### IP Header

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protocol</td>
<td>4 (TCP)</td>
</tr>
<tr>
<td>Source IP Address</td>
<td>128.143.137.144</td>
</tr>
<tr>
<td>Destination IP Address</td>
<td>128.143.71.21</td>
</tr>
<tr>
<td>TTL</td>
<td>64</td>
</tr>
<tr>
<td>Source Port</td>
<td>32</td>
</tr>
<tr>
<td>Destination Port</td>
<td>80</td>
</tr>
<tr>
<td>Sequence Number</td>
<td>12345678</td>
</tr>
<tr>
<td>Acknowledgment Number</td>
<td>98765432</td>
</tr>
<tr>
<td>Window Size</td>
<td>1024</td>
</tr>
<tr>
<td>Checksum</td>
<td>8bff</td>
</tr>
<tr>
<td>Urgent Pointer</td>
<td>0</td>
</tr>
</tbody>
</table>

#### Ethernet Header

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Destination MAC Address</td>
<td>00-00-00-00-00-00</td>
</tr>
<tr>
<td>Source MAC Address</td>
<td>00-00-00-00-00-00</td>
</tr>
</tbody>
</table>

#### Format of the IP Header

```
+----------+----------+----------+----------+----------+
| 0x4      | 0x5      | 0x0      | 0x0      | 4410     |
| 9d08     | 0102     | 00000000000002 |
| 12810    | 0x06     | 8bff     |
+----------+----------+----------+----------+----------+
```

#### Ethernet Frame

```
128.143.137.144
128.143.71.21
```
Encapsulation and Demultiplexing: TCP Header

<table>
<thead>
<tr>
<th>Source Port Number</th>
<th>Destination Port Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequence number (32 bits)</td>
<td></td>
</tr>
<tr>
<td>Acknowledgement number (32 bits)</td>
<td></td>
</tr>
<tr>
<td>Flags</td>
<td>window size</td>
</tr>
<tr>
<td>TCP checksum</td>
<td>urgent pointer</td>
</tr>
<tr>
<td>option type</td>
<td>length</td>
</tr>
</tbody>
</table>

Option: maximum segment size
Encapsulation and Demultiplexing: TCP Header

```
1627_{10}  80_{10}
607835_{10}
0_{10}
6_{10}  000000_{2}  000010_{2}  8192_{10}
0x598e  0000_{2}
2_{10}  4_{10}  1460_{10}
```
Encapsulation and Demultiplexing: Application data

No Application Data in this frame
Different Views of Networking

- Different Layers of the protocol stack have a different view of the network. This is HTTP’s and TCP’s view of the network.

![Diagram showing different views of networking with Argon 128.143.137.144 and Neon 128.143.71.21]
Network View of IP Protocol

Network 128.143.137.0/24

128.143.137.144

128.143.137.1

Router

Network 128.143.71.0/24

128.143.71.21

128.143.71.1
Network View of Ethernet

- Ethernet’s view of the network
IP Addresses

- Structure of an IP address
- Subnetting
- CIDR
IP Addresses

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>version (4 bits)</td>
<td>header length</td>
<td>Type of Service/TOS (8 bits)</td>
<td>Total Length (in bytes) (16 bits)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Identification (16 bits)</td>
<td>flags (3 bits)</td>
<td>Fragment Offset (13 bits)</td>
<td></td>
</tr>
<tr>
<td>TTL Time-to-Live (8 bits)</td>
<td>Protocol (8 bits)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Source IP address (32 bits)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Destination IP address (32 bits)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# IP Addresses

<table>
<thead>
<tr>
<th>0x4</th>
<th>0x5</th>
<th>0x00</th>
<th>44&lt;sub&gt;10&lt;/sub&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>9d08</td>
<td>01&lt;sub&gt;2&lt;/sub&gt;</td>
<td>0000000000000&lt;sub&gt;2&lt;/sub&gt;</td>
<td></td>
</tr>
<tr>
<td>128&lt;sub&gt;10&lt;/sub&gt;</td>
<td>0x06</td>
<td>8bff</td>
<td>128.143.137.144</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>128.143.71.21</td>
</tr>
</tbody>
</table>

The diagram illustrates the structure of an Ethernet frame with IP addresses and TCP headers. The IP addresses are represented in both decimal and hexadecimal formats. The diagram also shows the breakdown of the fields within the IP header and TCP header, with the application data and Ethernet trailer being part of the overall frame.
What is an IP Address?

• An IP address is a unique global address for a network interface

• An IP address:
  • is a 32 bit long identifier
  • encodes a network number (network prefix) and a host number
Dotted Decimal Notation

- IP addresses are written in a so-called *dotted decimal notation*

- Each byte is identified by a decimal number in the range $[0..255]$: 

$$
\begin{array}{cccc}
10000000 & 10001111 & 10001001 & 10010000 \\
1^{st} \text{ Byte} & 2^{nd} \text{ Byte} & 3^{rd} \text{ Byte} & 4^{th} \text{ Byte} \\
= 128 & = 143 & = 137 & = 144 \\
\end{array}
$$

$$128.143.137.144$$
Network prefix and Host number

- The network prefix identifies a network and the host number identifies a specific host (actually, interface on the network).

  - How do we know how long the network prefix is?
  - The network prefix is implicitly defined (class-based addressing)
  - The network prefix is indicated by a netmask.
Example

- **Example**: ellington.cs.virginia.edu

- Network id is: **128.143.0.0**
- Host number is: **137.144**
- Network mask is: **255.255.0.0** or **ffff0000**
- Prefix notation: **128.143.137.144/16**
- Network prefix is 16 bits long
The old way: Classful IP Addresses

- When Internet addresses were standardized (early 1980s), the Internet address space was divided up into classes:
  - **Class A**: Network prefix is 8 bits long
  - **Class B**: Network prefix is 16 bits long
  - **Class C**: Network prefix is 24 bits long

- Each IP address contained a key which identifies the class:
  - **Class A**: IP address starts with “0”
  - **Class B**: IP address starts with “10”
  - **Class C**: IP address starts with “110”
The old way: Internet Address Classes

Class A
- Network Prefix: 8 bits
- Host Number: 24 bits

Class B
- Network Prefix: 16 bits
- Host Number: 16 bits

Class C
- Network Prefix: 24 bits
- Host Number: 8 bits
### The old way: Internet Address Classes

<table>
<thead>
<tr>
<th>Class D</th>
<th>Bit #</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0 1 2 3 4</td>
<td>1110</td>
<td>multicast group id</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class E</th>
<th>Bit #</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0 1 2 3 4 5</td>
<td>11110</td>
<td>(reserved for future use)</td>
</tr>
</tbody>
</table>
Problems with Classful IP Addresses

• The original classful address scheme had a number of problems

**Problem 1. Too few network addresses for large networks**
  • Class A and Class B addresses are gone

**Problem 2. Two-layer hierarchy is not appropriate for large networks with Class A and Class B addresses.**
  • **Fix #1:** Subnetting
Problems with Classful IP Addresses

Problem 3. Inflexible. Assume a company requires 2,000 addresses

- Class A and B addresses are overkill
- Class C address is insufficient (requires 8 Class C addresses)

Fix #2: Classless Interdomain Routing (CIDR)
Problems with Classful IP Addresses

**Problem 4: Exploding Routing Tables:** Routing on the backbone Internet needs to have an entry for each network address. In 1993, the size of the routing tables started to outgrow the capacity of routers.

- **Fix #2:** Classless Interdomain Routing (CIDR)
Problems with Classful IP Addresses

Problem 5. The Internet is going to outgrow the 32-bit addresses

- Fix #3: IP Version 6
Subnetting

Problem: Organizations have multiple networks which are independently managed

Solution 1: Allocate one or more addresses for each network
- Difficult to manage
- -> From the outside of the organization - each network must be addressable.

Solution 2: Add another level of hierarchy to the IP addressing structure
Basic Idea of Subnetting

- Split the host number portion of an IP address into a subnet number and a (smaller) host number.

- Result is a 3-layer hierarchy

- Then:
  - Subnets can be freely assigned within the organization
  - Internally, subnets are treated as separate networks
  - Subnet structure is not visible outside the organization
Routers and hosts use an extended network prefix (subnet mask) to identify the start of the host numbers.

With subnetting, Class B networks have the following structure:

<table>
<thead>
<tr>
<th>10</th>
<th>network</th>
<th>host</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Network Prefix (16 bits)

Extended Network Prefix (24 bits)

Subnet mask (255.255.255.0)

There are different ways of subnetting. Commonly used netmasks for university networks with /16 prefix (Class B) are 255.255.255.0 and 255.255.0.0.
Typical Addressing Plan for an Organization that uses subnetting

- Each layer-2 network (Ethernet segment, FDDI segment) is allocated a subnet address when connected to a router.

```
128.143.0.0 / 24
128.143.14.0 / 24
128.143.7.0 / 24
128.143.8.0 / 24
128.143.16.0 / 24
128.143.17.0 / 24
128.143.18.0 / 24
128.143.22.0 / 24
128.143.136.0 / 24
```
Advantages of Subnetting

- With subnetting, IP addresses use a 3-layer hierarchy:
  - Network
  - Subnet
  - Host

- Improves efficiency of IP addresses by not consuming an entire address space for each physical network.

- Reduces router complexity. Since external routers do not know about subnetting, the complexity of routing tables at external routers is reduced.

- Note: Length of the subnet mask need not be identical at all subnetworks.
CIDR - Classless Interdomain Routing

- IP backbone routers have one routing table entry for each network address:
  - With subnetting, a backbone router only needs to know one entry for each network
  - This is acceptable for Class A and Class B networks
    - $2^7 = 128$ Class A networks
    - $2^{14} = 16,384$ Class B networks
  - But this is not acceptable for Class C networks
    - $2^{21} = 2,097,152$ Class C networks
- In 1993, the size of the routing tables started to outgrow the capacity of routers
- Consequence: The Class-based assignment of IP addresses had to be abandoned
CIDR - Classless Interdomain Routing

- **Goals:**
  - Restructure IP address assignments to increase efficiency
  - Hierarchical routing aggregation to minimize route table entries

**Key Concept:** The length of the network id (prefix) in the IP addresses is kept arbitrary

- **Consequence:** Routers advertise the IP address and the length of the prefix
CIDR Example

- CIDR notation of a network address:
  
  `192.0.2.0/18`

  - "18" says that the first 18 bits are the network part of the address (and 14 bits are available for specific host addresses)

- The network part is called the prefix

- Assume that a site requires a network address with 1000 addresses

- With CIDR, the network is assigned a continuous block of 1024 addresses with a 22-bit long prefix
## CIDR: Prefix Size vs. Network Size

<table>
<thead>
<tr>
<th>CIDR Block Prefix</th>
<th># of Host Addresses</th>
</tr>
</thead>
<tbody>
<tr>
<td>/27</td>
<td>32 hosts</td>
</tr>
<tr>
<td>/26</td>
<td>64 hosts</td>
</tr>
<tr>
<td>/25</td>
<td>128 hosts</td>
</tr>
<tr>
<td>/24</td>
<td>256 hosts</td>
</tr>
<tr>
<td>/23</td>
<td>512 hosts</td>
</tr>
<tr>
<td>/22</td>
<td>1,024 hosts</td>
</tr>
<tr>
<td>/21</td>
<td>2,048 hosts</td>
</tr>
<tr>
<td>/20</td>
<td>4,096 hosts</td>
</tr>
<tr>
<td>/19</td>
<td>8,192 hosts</td>
</tr>
<tr>
<td>/18</td>
<td>16,384 hosts</td>
</tr>
<tr>
<td>/17</td>
<td>32,768 hosts</td>
</tr>
<tr>
<td>/16</td>
<td>65,536 hosts</td>
</tr>
<tr>
<td>/15</td>
<td>131,072 hosts</td>
</tr>
<tr>
<td>/14</td>
<td>262,144 hosts</td>
</tr>
<tr>
<td>/13</td>
<td>524,288 hosts</td>
</tr>
</tbody>
</table>
CIDR and Address assignments

- Backbone ISPs obtain large block of IP addresses space and then reallocate portions of their address blocks to their customers.

**Example:**

- Assume that an ISP owns the address block **206.0.64.0/18**, which represents **16,384** ($2^{32-18}=2^{14}$) IP addresses.
- Suppose a client requires 800 host addresses.
- With CIDR: Assign a **/22 block** ($512=2^9<800<1024=2^{10}$ -> $32-10=22$), i.e., **206.0.68.0/22** gives a block of **1,024** ($2^{10}$) IP addresses.
CIDR and Routing Information

- **Internet Backbone**
- **ISP X** owns:
  - 206.0.64.0/18
  - 204.188.0.0/15
  - 209.88.232.0/21

- **Company X**:
  - 206.0.68.0/22

- **ISP Y**:
  - 209.88.237.0/24

- **Organization z1**:
  - 209.88.237.192/26

- **Organization z2**:
  - 209.88.237.0/26
CIDR and Routing Information

ISP X sends everything which matches the prefix:
- 206.0.68.0/22 to Company X
- 209.88.237.0/24 to ISP y

ISP y sends everything which matches the prefix:
- 209.88.237.192/26 to Organizations z1
- 209.88.237.0/26 to Organizations z2

Backbone sends everything which matches the prefixes:
- 206.0.64.0/18, 204.188.0.0/15, 209.88.232.0/21 to ISP X.

Backbone routers do not know anything about Company X, ISP Y, or Organizations z1, z2.

ISP X does not know about Organizations z1, z2.
Example

- The IP Address: **207.2.88.170**
  - Belongs to: **Cable & Wireless USA**  
    
  - Belongs to: **City of Charlottesville, VA**: **207.2.88.0 - 207.2.92.255**
  - Belongs to: **Cable & Wireless USA**  

<table>
<thead>
<tr>
<th>Address</th>
<th>Subnet Mask</th>
<th>Network ID</th>
<th>Broadcast ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>207.2.88.170</td>
<td>255.255.0.0</td>
<td>207.2.88.0</td>
<td>207.2.92.255</td>
</tr>
<tr>
<td>207.2.88.0</td>
<td>255.255.0.0</td>
<td>207.2.88.0</td>
<td>207.2.92.255</td>
</tr>
<tr>
<td>207.0.0.0</td>
<td>255.255.255.255</td>
<td>207.0.0.0</td>
<td>207.3.255.255</td>
</tr>
</tbody>
</table>
CIDR and Routing

- **Aggregation** of routing table entries:
  - 128.143.0.0/16 and 128.142.0.0/16 are represented as 128.142.0.0/15

- **Longest prefix match**: Routing table lookup finds the routing entry that matches the longest prefix

What is the outgoing interface for 128.143.137.0?

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>128.0.0.0/4</td>
<td>interface #5</td>
</tr>
<tr>
<td>128.128.0.0/9</td>
<td>interface #2</td>
</tr>
<tr>
<td>128.143.128.0/17</td>
<td>interface #1</td>
</tr>
</tbody>
</table>

Routing table