

# Contents

<b>1 Sem Outline</b>	<b>6</b>
<b>2 Exam Notes</b>	<b>7</b>
2.1 Chapter 1 . . . . .	7
2.2 Chapter 2: Application Layer . . . . .	7
2.3 Chapter 3: Transport Layer . . . . .	7
2.4 Chapter 4: Network Layer – Data Plane . . . . .	7
2.5 Chapter 5: Network Layer – Control Plane . . . . .	7
2.6 Chapter 6: Link Layer . . . . .	7
2.7 Chapter 7: Wireless . . . . .	7
2.8 Chapter 8: Security . . . . .	7
2.9 Chapter 9: Multimedia . . . . .	8
2.10 Packet Formats . . . . .	8
<b>3 Chapter 1</b>	<b>9</b>
3.1 Network Structure . . . . .	9
3.2 Access Network . . . . .	9
3.2.1 Digital Subscriber Line (DSL) . . . . .	9
3.2.2 Cable Network . . . . .	9
3.3 Sending . . . . .	9
3.4 Physical Media . . . . .	9
3.4.1 Coax . . . . .	9
3.4.2 Fiber Optic Cable . . . . .	10
3.4.3 Radio . . . . .	10
3.5 Packet-switching . . . . .	10
3.5.1 Store-and-forward . . . . .	10
3.5.2 Packet switching versus circuit switching . . . . .	10
3.6 Packet Loss . . . . .	10
3.6.1 Nodal Processing . . . . .	10
3.6.2 Queuing Delay . . . . .	10
3.6.3 Transmission Delay . . . . .	10
3.6.4 Propagation Delay . . . . .	10
3.7 Throughput . . . . .	10
3.8 Layering . . . . .	11
3.8.1 Why Layering? . . . . .	11
3.8.2 Internet Protocol Stack . . . . .	11
3.8.3 ISO/OSI Reference Model . . . . .	11
3.9 Security . . . . .	11
3.9.1 DoS: Denial of Service . . . . .	11
3.9.2 Sniffing . . . . .	11
3.9.3 IP Spoofing . . . . .	11
<b>4 Chapter 2</b>	<b>11</b>
4.1 Application Architectures . . . . .	11
4.1.1 Client-Server . . . . .	11
4.1.2 Peer-to-Peer (P2P) . . . . .	11
4.2 Transport Service is needed . . . . .	12
4.3 Transport Protocol Services . . . . .	12
4.3.1 TCP . . . . .	12
4.3.2 UDP . . . . .	12
4.3.3 Securing TCP . . . . .	12
4.4 HTTP: Hypertext Transfer Protocol . . . . .	12

4.4.1	Method Types . . . . .	12
4.4.2	Response Codes . . . . .	13
4.5	Cookies . . . . .	13
4.6	Web Caches (proxy server) . . . . .	13
4.6.1	Conditional GET . . . . .	13
4.7	Electronic Mail: SMTP . . . . .	13
4.7.1	Mail Access Protocols . . . . .	13
4.8	DNS: Domain Name System . . . . .	13
4.8.1	DNS Services . . . . .	13
4.8.2	TLD, authoritative servers . . . . .	14
4.8.3	Local DNS name server . . . . .	14
4.8.4	DNS Name Resolution . . . . .	14
4.8.5	Caching . . . . .	14
4.8.6	DNS Records . . . . .	14
4.8.7	Protocol . . . . .	14
4.8.8	Attacking DNS . . . . .	14
4.9	File Distribution Time . . . . .	14
4.9.1	Client-server . . . . .	14
4.9.2	P2P . . . . .	15
4.9.3	BitTorrent . . . . .	15
4.10	Multimedia . . . . .	15
4.10.1	Video . . . . .	15
4.10.2	DASH . . . . .	15
4.10.3	Content Distribution Networks (CDNs) . . . . .	15
<b>5</b>	<b>Chapter 3</b>	<b>15</b>
5.1	Transport vs. Network Layer . . . . .	15
5.2	Multiplexing/demultiplexing . . . . .	15
5.2.1	How demultiplexing works . . . . .	15
5.3	UDP . . . . .	16
5.3.1	UDP Checksum . . . . .	16
5.4	Pipelined Protocols . . . . .	16
5.4.1	Go-Back-N . . . . .	16
5.4.2	Selective Repeat . . . . .	16
5.5	TCP Segment Structure . . . . .	16
5.5.1	TCP Round Trip Time, Timeout . . . . .	16
5.5.2	TCP Flow Control . . . . .	17
5.5.3	Closing . . . . .	17
5.6	TCP Congestion Control . . . . .	17
5.7	Fairness . . . . .	17
5.8	Explicit Congestion Notification . . . . .	17
<b>6</b>	<b>Chapter 4</b>	<b>17</b>
6.1	Network Layer . . . . .	17
6.1.1	Network Layer Functions . . . . .	17
6.1.2	Data Plane, Control Plane . . . . .	17
6.2	Router Forwarding . . . . .	17
6.2.1	Destination-based forwarding . . . . .	18
6.2.2	Switching Fabrics . . . . .	18
6.2.3	Input port queuing . . . . .	18
6.2.4	Output ports . . . . .	18
6.2.5	Scheduling Mechanisms . . . . .	18
6.3	IP . . . . .	19

6.3.1	IP Datagram Format . . . . .	19
6.3.2	IP Fragmentation, Reassembly . . . . .	19
6.3.3	IP Addressing . . . . .	19
6.3.4	Subnets . . . . .	19
6.3.5	DHCP: Dynamic Host Configuration Protocol . . . . .	19
6.3.6	ICANN . . . . .	19
6.3.7	NAT . . . . .	19
6.3.8	IPv6 . . . . .	20
6.3.9	IPv6 Datagram Format . . . . .	20
6.3.10	Other changes from IPv4 . . . . .	20
6.3.11	Transition from IPv4 to IPv6 . . . . .	20
6.4	Generalized Forwarding and SDN . . . . .	20
6.4.1	OpenFlow data plane abstraction . . . . .	20
6.4.2	OpenFlow Abstraction . . . . .	21
<b>7</b>	<b>Chapter 5</b>	<b>21</b>
7.1	Control Plane . . . . .	21
7.1.1	Per-router control plane . . . . .	21
7.1.2	Logically centralized control plane . . . . .	21
7.2	Routing Protocols . . . . .	21
7.2.1	Global or Decentralized information . . . . .	21
7.2.2	Static or Dynamic . . . . .	21
7.2.3	Link-State Routing Algorithm . . . . .	21
7.2.4	Distance Vector Algorithm . . . . .	22
7.2.5	Comparison of LS and DV algorithms . . . . .	22
7.3	Making Routing Scalable . . . . .	22
7.3.1	Interconnected ASes . . . . .	22
7.3.2	Intra-AS Routing . . . . .	22
7.3.3	Hierarchical OSPF . . . . .	23
7.3.4	Internet inter-AS routing . . . . .	23
7.4	Software Defined Networking (SDN) . . . . .	23
7.4.1	SDN perspective: Data Plane Switches . . . . .	23
7.4.2	SDN perspective: SDN controller . . . . .	24
7.4.3	SDN perspective: Control Applications . . . . .	24
7.4.4	OpenFlow: controller-to-switch messages . . . . .	24
7.5	OpenDaylight (ODL) controller . . . . .	24
7.6	ONOS controller . . . . .	24
7.7	ICMP: Internet Control Message Protocol . . . . .	24
7.8	Network Management and SNMP . . . . .	25
7.8.1	SNMP Protocol: Message Types . . . . .	25
<b>8</b>	<b>Chapter 6</b>	<b>26</b>
8.1	Link Layer . . . . .	26
8.1.1	Link layer services . . . . .	26
8.2	Error Detection . . . . .	26
8.2.1	Parity Checking . . . . .	26
8.2.2	Cyclic Redundancy Check . . . . .	26
8.3	Multiple access links, protocols . . . . .	26
8.4	MAC Protocols . . . . .	27
8.4.1	TDMA: time division multiple access . . . . .	27
8.4.2	FDMA: frequency division multiple access . . . . .	27
8.5	Random Access Protocols . . . . .	27
8.5.1	Slotted ALOHA . . . . .	27

8.5.2	Pure (unslotted) ALOHA	27
8.5.3	CSMA (carrier sense multiple access)	27
8.5.4	CSMA/CD (collision detection)	28
8.6	“Taking turns” MAC protocols	28
8.6.1	Polling	28
8.6.2	Token Passing	28
8.7	Cable Access Network	28
8.7.1	DOCSIS: Data Over Cable Service Interface Spec	28
8.8	MAC Addresses and ARP	28
8.8.1	LAN Address	28
8.8.2	ARP: Address Resolution Protocol	29
8.9	Ethernet	29
8.9.1	Physical Topology	29
8.9.2	Ethernet frame structure	29
8.9.3	Ethernet: unreliable, connectionless	29
8.9.4	Ethernet switch	29
8.10	Switches vs routers	29
8.11	VLANs	29
8.11.1	Port-based VLAN	29
<b>9</b>	<b>Chapter 7</b>	<b>30</b>
9.1	Wireless	30
9.1.1	Infrastructure mode	30
9.1.2	Ad Hoc mode	30
9.1.3	Wireless Link Characteristics	30
9.1.4	Code Division Multiple Access (CDMA)	31
9.2	IEEE 802.11 Wireless LAN	31
9.2.1	802.11 LAN architecture	31
9.2.2	Channels, association	31
9.2.3	IEEE 802.11 MAC Protocol: CSMA/CA	31
9.2.4	802.11 frame: addressing	32
9.3	802.11 Power Management	32
9.4	802.15: Personal Area Network	32
<b>10</b>	<b>Chapter 8</b>	<b>32</b>
10.1	What is network security	32
10.1.1	Breaking an encryption scheme	33
10.2	Symmetric Key Cryptography	33
10.2.1	Substitution Cipher	33
10.2.2	DES: Data Encryption Standard	33
10.2.3	AES: Advanced Encryption Standard	33
10.3	Public Key Cryptography	33
10.4	Authentication	33
10.5	Digital Signatures	33
10.6	Hashing	33
10.6.1	Hash Algorithms	34
10.7	Certification Authorities	34
10.8	SSL: Secure Sockets Layer	34
10.8.1	SSL cipher suite	34
10.9	Network-layer Security	34
10.9.1	What is network-layer confidentiality	34
10.9.2	Virtual Private Networks (VPNs)	34
10.9.3	IPsec services	35

10.10	Firewalls . . . . .	35
10.10.1	Limitations of firewalls, gateways . . . . .	35
10.10.2	Intrusion detection systems . . . . .	35
<b>11</b>	<b>Chapter 9</b>	<b>35</b>
11.1	Multimedia . . . . .	35
11.1.1	Audio . . . . .	35
11.1.2	Video . . . . .	35
11.1.3	3 Application Types . . . . .	35
11.2	Voice-over-IP (VoIP) . . . . .	36
11.2.1	Adaptive playout delay . . . . .	36
11.2.2	Recovery from packet loss . . . . .	36
11.3	Real-Time Protocol (RTP) . . . . .	36
11.3.1	RTP runs on top of UDP . . . . .	36
11.3.2	RTP and QoS . . . . .	36
11.4	Real-Time Control Protocol (RTCP) . . . . .	36
11.4.1	RTCP: packet types . . . . .	36
11.5	SIP: Session Initiation Protocol . . . . .	37
11.5.1	SIP services . . . . .	37
<b>12</b>	<b>Packet Format</b>	<b>38</b>
12.1	Application Layer . . . . .	38
12.1.1	Request Message . . . . .	38
12.1.2	Response Message . . . . .	38
12.2	Presentation . . . . .	38
12.3	Session . . . . .	39
12.4	Transport Layer . . . . .	39
12.4.1	UDP . . . . .	39
12.4.2	TCP . . . . .	39
12.5	Network Layer . . . . .	40
12.5.1	IPv4 . . . . .	40
12.6	Link Layer . . . . .	41
12.7	Physical . . . . .	41
<b>13</b>	<b>Protocols</b>	<b>42</b>
13.1	Application Layer . . . . .	42
13.1.1	HTTP: Hypertext Transfer Protocol . . . . .	42
13.1.2	SMTP: Simple Mail Transfer Protocol . . . . .	42
13.1.3	DNS: Domain Name Service . . . . .	42
13.1.4	DASH: Dynamic, Adaptive Streaming over HTTP . . . . .	42
13.2	Transport Layer . . . . .	42
13.2.1	UDP: User Datagram Protocol . . . . .	42
13.2.2	TCP: Transmission Control Protocol . . . . .	42
13.2.3	RTP: Real-Time Protocol . . . . .	42
13.2.4	RTCP: Real-Time Control Protocol . . . . .	42
13.2.5	SIP: Session Initiation Protocol . . . . .	42
13.3	Network Layer . . . . .	42
13.3.1	IP: Internet Protocol . . . . .	42
13.3.2	DHCP: Dynamic Host Configuration Protocol . . . . .	42
13.3.3	NAT: Network Address Translation . . . . .	42
13.3.4	IPv6 . . . . .	42
13.3.5	ICMP: Internet Control Message Protocol . . . . .	42
13.3.6	OSPF: Open Shortest Path First . . . . .	42
13.3.7	BGP: Border Gateway Protocol . . . . .	42

13.3.8 IS-IS: Intermediate System to Intermediate System . . . . .	42
13.3.9 RIP: Routing Information Protocol . . . . .	42
13.3.10 IGRP: Interior Gateway Routing Protocol . . . . .	42
13.4 Link Layer . . . . .	42
13.4.1 TDMA: Time Division Multiple Access . . . . .	42
13.4.2 FDMA: Frequency Division Multiple Access . . . . .	42
13.4.3 Slotted ALOHA . . . . .	42
13.4.4 Pure (unslotted) ALOHA . . . . .	43
13.4.5 CSMA (Carrier Sense Multiple Access) . . . . .	43
13.4.6 CSMA/CD: Collision Detection . . . . .	43
13.4.7 Polling . . . . .	43
13.4.8 Token Passing . . . . .	43
13.4.9 DOCSIS: Data Over Cable Service Interface Spec . . . . .	43
13.4.10 ARP: Address Resolution Protocol . . . . .	43

## 14 Acronyms 44

### Contributors:

- Daniel Fitz (Sanchez)

## 1 Sem Outline

Week (dates)	Lecture
1	Computer Networks and the Internet
2	Principles of Nw Apps: HTTP, SMTP, DNS
3	Application Layer: P2P, CDN, Sockets
4	Networking at UQ
5	Transport Layer: UDP
6	Transport Layer: TCP
7	Network Layer: Data Plane
8	Network Layer: Control Plane
9	Link Layer
11	Wireless and Mobile
12	Security
13	Multimedia

Table 1: Week Outline

## 2 Exam Notes

The exam will consist of:

- A number of analytical questions, similar to the tutorial questions. You won't be asked any complex analytic problems which are completely different to those in tutorials
- A number of short answer questions of the type: compare XXX to YYY and explain the differences, or advantages/disadvantages of these protocols/algorithms/applications/techniques
- Questions about different protocols, their functions and where they fit in the network protocol stack. You won't be asked about protocols you have not seen in lectures
- Questions about packet exchanges in some common protocols (e.g. DHCP, DNS, ARP, TCP, HTTP)

*No multiple choice questions this year 😊*

### 2.1 Chapter 1

- What is the Internet
- Network Edge
- Network Core
- Delay, Loss Throughput
- Protocol Layers and their service models

*Not Examinable:* Networks under attack, history of networking

### 2.2 Chapter 2: Application Layer

- Principles of Networked Applications
- Web and HTTP (including options covered in lectures/labs)
- Electronic Mail
- DNS (but no detailed message/packet format)
- Peer-to-peer
- Internet Video

*Not Examinable:* Detailed message formats for DNS and for email, case studies, socket programming

### 2.3 Chapter 3: Transport Layer

All Material

### 2.4 Chapter 4: Network Layer – Data Plane

All Material

### 2.5 Chapter 5: Network Layer – Control Plane

Most of the material covered, except as below, including a general overview of what SNMP does. You should understand link-state and distance vector routing. You won't be asked any numerical questions with distance-vector. For routing protocols, you should know about BGP, OSPF, IS-IS, RIP (which isn't in lectures, but is an example of an intra-AS distance-vector algorithm). All you really need to know about these algorithms are whether they are inter-AS or intra-AS, link-state or distance-vector.

*Not Examinable:* Details of SNMP architecture and packet formats. Details of BGP (5.4.2, 5.4.3, 5.4.5 are not examinable)

### 2.6 Chapter 6: Link Layer

- General Principles
- Error Detection and Correction – services provided, differences between correction and detection
- Multiple Access Links and Protocols, but NOT DOCSIS
- Switched Local Area Networks
- “Day in the Life of a Web Page Request” – details of each stage are covered in the earlier sections

*Not Examinable:* Exactly how to calculate parity, checksum, CRC, DOCSIS, MPLS, Data Center Networking

### 2.7 Chapter 7: Wireless

- General Principles
- Wireless characteristics
- WiFi (IEEE 802.11) except as below

*Not Examinable:* Mobility in WiFi, advanced features in WiFi (Ch 7.3.5). Personal area Networks. Cellular Internet Access. Mobility Management, Mobile IP, Mobility effects on higher layers

### 2.8 Chapter 8: Security

- What is network security – confidentiality, integrity, authentication
- Cryptographic principles – symmetric and public key algorithms (you won't be asked to calculate any ciphers)
- Names, types and uses of common cyphers, at least: Diffie-Hellman, RSA, DES, 3DES, AES, MD5, SHA-1

- Message integrity and signatures
- SSL and TLS
- IP Sec and VPN
- Firewalls and Intrusion Detection Systems – general principles

*Not Examinable:* Details of cipher algorithms, key lengths. Securing Email. Wireless security

## 2.9 Chapter 9: Multimedia

- Properties of multimedia
- UDP and HTTP streaming
- Voice over IP
- Protocols – RTP, SIP

*Not Examinable:* Case Studies (e.g. Skype). Network Support for multimedia, such as token-bucket, diffserv, QoS

## 2.10 Packet Formats

Must understand and decode the packet contents if given a byte stream for:

**Link Layer:** Ethernet (but not VLAN packets)

**Network Layer:** IPv4 (not IPv6), you won't be asked to decode option fields, but they may be present. These IPv4 packets may contain protocols like DNS or ICMP, but you won't be asked to decode the contents of those packets

**Transport Layer:** TCP, UDP.. You won't be asked to decode option fields, by they may be present

**Application Layer:** Simple HTTP request and reply. If you are required to decode text messages you will be given a table of ASCII codes



## 3 Chapter 1

- billions of connected computing devices
- transmission rate: **bandwidth**
- **Packet Switches:** Forward packets
  - **routers** and **switches**
- **Internet:** “network of networks” (Interconnected ISPs)
- **Protocols** control sending, receiving (e.g. TCP, IP, HTTP, Skype, 802.11)
- **Internet standards**
  - RFC:** Request for comments
  - IETF:** Internet Engineering Task Force

### 3.1 Network Structure

- **Network Edge**
  - hosts: clients and servers
  - servers often in data centers
- **Access networks, physical media:** wired, wireless communication links
- **network core:**
  - interconnected routers
  - network of networks

### 3.2 Access Network

#### 3.2.1 Digital Subscriber Line (DSL)

- use **existing** telephone line to central office DSLAM
  - data over DSL phone line goes to Internet
  - voice over DSL phone line goes to telephone net
- < 2.5 Mbps upstream transmission rate (typically < 1 Mbps)
- < 24 Mbps downstream transmission rate (typically < 10 Mbps)

#### 3.2.2 Cable Network

**frequency division multiplexing:** different channels transmitted in different frequency bands

- **HFC: hybrid fiber coax**
  - asymmetric: up to 30Mbps downstream transmission rate, 2 Mbps upstream transmission rate
- **network** of cable, fiber attaches homes to ISP router
  - homes **share access network** to cable head-end

- unlike DSL, which has dedicated access to central office

#### wireless LANS:

- within building (30 meters)
- 802.11b/g/n (WiFi): 11,54,450 Mbps transmission rate

#### wide-area wireless access:

- provided by telco (cellular) operator, 10's km
- between 1 and 10 Mbps
- 3G, 4G, LTE

### 3.3 Sending

- takes application message
- breaks into smaller chunks, known as **packets**, of length  $L$  bits
- transmits packet into access network at **transmission rate**  $R$ 
  - link transmission rate, aka link **capacity**, aka link **bandwidth**

#### Note 1: Packet Transmission Delay

$$\text{packet transmission delay} = \frac{\text{time needed to transmit } L\text{-bit packet into link}}{R} = \frac{L \text{ (bits)}}{R \text{ (bits/sec)}}$$

### 3.4 Physical Media

- **bit:** propagates between transmitter/receiver pairs
- **physical link:** what lies between transmitter and receiver
- **guided media:** signals propagate in solid media (copper, fiber, coax)
- **unguided media:** signals propagate freely, e.g. radio
- **twisted pair (TP):** two insulated copper wires
  - Category 5: 100 Mbps, 1 Gbps Ethernet
  - Category 6: 10 Gbps

#### 3.4.1 Coax

- two concentric copper conductors
- bidirectional
- broadband: multiple channels on cable, HFC

### 3.4.2 Fiber Optic Cable

- glass fiber carrying light pulses, each pulse a bit
- high-speed operation: high-speed point-to-point transmission (e.g. 10's - 100's Gbps transmission rate)
- low error rate
  - repeaters spaced far apart
  - immune to electromagnetic noise

### 3.4.3 Radio

- signal carried in electromagnetic spectrum
- no physical "wire"
- bidirectional
- propagation environment effects:
  - reflection
  - obstruction by objects
  - interference

#### Radio Link Types:

- **terrestrial microwave:** up to 45 Mbps channels
- **LAN** (e.g. WiFi) 54 Mbps
- **wide-area** (e.g. cellular) 4G cellular: 10 Mbps
- **satellite**
  - Kbps to 45 Mbps channel (or multiple smaller channels)
  - 270 msec end-end delay
  - geosynchronous versus low altitude

## 3.5 Packet-switching

### 3.5.1 Store-and-forward

$L$  bits per packet

Source to destination:  $R$  bps

- takes  $\frac{L}{R}$  seconds to transmit (push out)  $L$ -bit packet into link at  $R$  bps
- **store and forward:** entire packet must arrive at router before it can be transmitted on next link

#### Note 2: End-End delay

$$\text{delay} = 2 \frac{L}{R}$$

(assuming zero propagation delay)

### 3.5.2 Packet switching versus circuit switching

Is packet switching a "slam dunk winner?"

- great for bursty data (resource sharing, simpler, no call setup)
- excessive congestion possible: packet delay and loss (protocols needed for reliable data transfer, congestion control)

## 3.6 Packet Loss

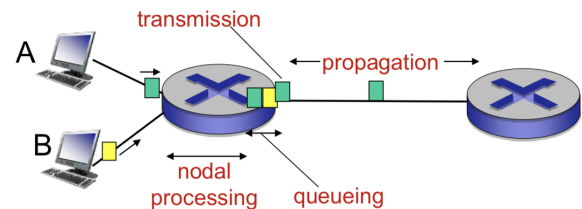


Figure 1: Packet Delay Algorithm Explanation

#### Note 3: Packet Delay Algorithm

$$d_{\text{nodal}} = d_{\text{proc}} + d_{\text{queue}} + d_{\text{trans}} + d_{\text{prop}}$$

### 3.6.1 Nodal Processing

$$d_{\text{proc}}$$

- check bit errors
- determine output link
- typically < msec

### 3.6.2 Queuing Delay

$$d_{\text{queue}}$$

- time waiting at output link for transmission
- depends on congestion level of router

### 3.6.3 Transmission Delay

$$d_{\text{trans}}$$

- $L$ : packet length (bits)
- $R$ : link bandwidth(bps)
- $d_{\text{trans}} = \frac{L}{R}$

### 3.6.4 Propagation Delay

$$d_{\text{prop}}$$

- $d$ : length of physical link
- $s$ : propagation speed ( $\approx 2 \times 10^8$  m/sec)
- $d_{\text{prop}} = \frac{d}{s}$

## 3.7 Throughput

Rate (bits/time unit) at which bits transferred between sender/receiver

**Instantaneous:** rate at given point in time

**Average:** rate over longer period of time

#### Note 4: Bottleneck Link

Link on end-end path that constrains end-end throughput

## 3.8 Layering

### 3.8.1 Why Layering?

Dealing with complex systems:

- Explicit structure allows identification, relationship of complex system's pieces (layered **reference model** for discussion)
- Modularization eases maintenance, updating system
  - change of implementation of layer's service transparent to rest of system
  - e.g. change in gate procedure doesn't affect rest of system
- layering considered harmful?

### 3.8.2 Internet Protocol Stack

**Application:** supporting network applications (FTP, SMTP, HTTP)

**Transport:** process-process data transfer (TCP, UDP)

**Network:** routing of datagrams from source to destination (IP, routing protocols)

**Link:** data transfer between neighboring network elements (Ethernet, 802.111 (WiFi), PPP)

**Physical:** bits "on the wire"

### 3.8.3 ISO/OSI Reference Model

Internet stack "missing" these layers. These services, if needed, must be implemented in application.

**Application:**

**Presentation:** allow applications to interpret meaning of data, e.g. encryption, compression, machine-specific conventions

**Session:** synchronization, check-pointing, recovery of data exchange

**Transport:**

**Network:**

**Link:**

**Physical:**

## 3.9 Security

- Malware can get in host from:

**Virus:** self-replicating infection by receiving/executing object (e.g. e-mail attachment)

**Worm:** self-replicating infection by passively receiving object that gets itself executed

- **Spyware malware** can record keystrokes, web sites visited, upload info to collection site
- Infected host can be enrolled in **botnet**, used for spam. DDoS attacks

### 3.9.1 DoS: Denial of Service

**Denial of Service (DoS):** attackers make resources (server, bandwidth) unavailable to legitimate traffic by overwhelming resource with bogus traffic

1. select target
2. break into hosts around the network (botnet)
3. send packets to target from compromised hosts

### 3.9.2 Sniffing

- broadcast media (shared Ethernet, wireless)
- promiscuous network interface reads/records all packets (e.g. including passwords) passing by

### 3.9.3 IP Spoofing

Send packet with false source address

## 4 Chapter 2

### 4.1 Application Architectures

#### 4.1.1 Client-Server

**Server:** Always-on host, Permanent IP address

**Clients:** Do not communicate directly with each other, May have dynamic IP addresses

#### 4.1.2 Peer-to-Peer (P2P)

- No always-on server
- Peers request service from other peers, provide service in return to other peers
- **Self Scalability** – new peers bring new service capacity, as well as new service demands

- Pers are intermittently connected and change IP addresses

#### Note 5: App-layer protocol defines

- **type of messages exchanged** – e.g. request, response
- **message syntax** – what fields in messages and how fields are delineated
- **message semantics** – meaning of information in fields
- **rules** for when and how processes send and respond to messages
- **open protocols** – defined in RFCs, allows for interoperability (e.g. HTTP, SMTP)
- **proprietary protocols** – e.g. Skype

### 4.3.3 Securing TCP

#### TCP and UDP

- no encryption
- cleartext passwords sent into socket traverse Internet in cleartext

#### SSL

- provides encrypted TCP connection
- data integrity
- end-point authentication

#### SSL is at app layer

- app use SSL libraries, that “talk” to TCP

#### SSL socket API

- cleartext passwords sent into socket traverse Internet encrypted

## 4.2 Transport Service is needed

**Data Integrity:** Some programs need 100% reliable data transfer (e.g. file transfer, web transactions), others can tolerate loss (e.g. audio)

**Timing:** Some programs require low delay to be “effective” (e.g. online games)

**Throughput:** Some programs require minimum amount of throughput to be “effective” (e.g. multimedia), some use whatever they have available (“elastic apps”)

**Security:** Encryption, Data Integrity

## 4.3 Transport Protocol Services

### 4.3.1 TCP

**Reliable Transport** between sending and receiving process

**Flow Control:** sender won’t overwhelm receiver

**Congestion Control:** throttle sender when network overloaded

**Connection-Oriented:** setup required between client and server processes

**Does Not Provide:** timing, minimum throughput guarantee, security

### 4.3.2 UDP

**Unreliable Data Transfer** between sending and receiving process

**Does Not Provide:** reliability, flow control, congestion control, timing, throughput guarantee, security, or connection setup

## 4.4 HTTP: Hypertext Transfer Protocol

- Web’s application layer protocol
- client/server model. Client request website and server serves HTTP object in response
- Uses TCP
- HTTP is stateless. Server maintains no information about past client requests
- **non-persistent HTTP:** one object sent over one TCP connection, downloading multiple object required multiple connections
- **persistent HTTP:** multiple object sent over single TCP connection

Non-persistent HTTP issues:

- requires 2 RTTs per object
- OS overhead for each TCP connection
- browsers often open parallel TCP connections to fetch referenced objects

Persistent HTTP:

- server leaves connection open after sending response
- subsequent HTTP messages between same client/server sent over open connection
- client sends requests as soon as it encounters a referenced object
- as little as one RTT for all the referenced objects

### 4.4.1 Method Types

**HTTP/1.0:** GET, POST, HEAD (asks server to leave requested object out of response)

**HTTP/1.1:** GET, POST, HEAD, PUT (uploads file in entity body to path specified in URL field), DELETE (deletes file specified in the URL field)

#### 4.4.2 Response Codes

**200 OK:** request succeeded, requested object later in this msg

**301 Moved Permanently:** requested object moved, new location specified later in this msg

**400 Bad Request:** request msg not understood by server

**404 Not Found:** requested document not found on this server

**505 HTTP Version Not Supported**

#### 4.5 Cookies

Uses: authorization, shopping carts, recommendations, user session state (Web, email)

#### 4.6 Web Caches (proxy server)

**Goal:** satisfy client request without involving origin server

- Browsers requests object from cache, if in cache the object is sent back otherwise cache requests object from origin
- Cache acts as both client and server
- Reduce response time for client request
- Reduce traffic

##### 4.6.1 Conditional GET

**Goal:** don't send object if cache has up-to-date cached version (lower link usage)

- **Cache:** specify date of cached copy in HTTP request `If-modified-since: <date>`
- **Server:** response contains no object if cached copy is up-to-date: `HTTP/1.0 Not Modified`

#### 4.7 Electronic Mail: SMTP

*RFC 2821*

- uses TCP to reliably transfer email message from client to server, port 25
- direct transfer: sending server to receiving server
- three phases of transfer: handshaking, transfer of messages, closure
- command/response interaction
- messages must be in 7-bit ASCII
- uses persistent connections

- requires message to be in 7-bit ASCII
- uses CRLF.CRLF to determine end of message

Difference to HTTP being, HTTP is server sending data, SMTP is client connection sending data

**SMTP:** protocol for exchanging email messages

**RFC 822:** standard for text message format (To, From, Subject, Body)

##### 4.7.1 Mail Access Protocols

**SMTP:** delivery/storage to receiver's server

**POP:** Post Office Protocol (*RFC 1939*): authorization, download

- POP3 is stateless across sessions
- Two main modes; download and delete, download and keep (allows multiple clients to read the same email)

**IMAP:** Internet Mail Access Protocol (*RFC 1730*): more features, including manipulation of stored message on server

- All messages stored on server
- Supports folders
- Keeps user state across sessions: names of folders and mappings between message IDs and folder name

**HTTP:** gmail, Hotmail, Yahoo, etc

#### 4.8 DNS: Domain Name System

- Lookup between names (e.g. google.com) and IP addresses
- **Distributed Database** implemented in hierarchy of many **name servers**
- **Application-layer protocol:** hosts, name servers communicate to **resolve** names (address/name translation)

Why not centralize DNS? Single point of failure, traffic volume, doesn't scale

##### 4.8.1 DNS Services

- hostname to IP address translation
- host aliasing (canonical, alias names)
- mail server aliasing
- load distribution (many IP addresses correspond to one name)

## 4.8.2 TLD, authoritative servers

### top-level domain (TLD) servers:

- responsible for com, org, net, edu, aero, jos, io
- and top-level country domains au, uk, ca
- Network Solutions maintains servers for .com TLD
- Educause for .edu TLD

### Authoritative DNS servers:

- organization's own DNS server(s), providing authoritative hostname to IP mappings for organization's named hosts
- can be maintained by organization or service provider

## 4.8.3 Local DNS name server

- does not strictly belong to hierarchy
- each ISP (residential ISP, company, university) has one (also called "default name server")
- when host makes DNS query, query is sent to its local DNS server
  - has local cache of recent name-to-address translation pairs (but may be out of date!)
  - acts as proxy, forwards query into hierarchy

## 4.8.4 DNS Name Resolution

**Iterated query:** contacted server replies with name of server to contact. So root dns sends the ip of the next dns server to contact

**Recursive query:** puts burden of name resolution on contacted name server. So root dns server contacts the next levels down which contacts next level down.

## 4.8.5 Caching

Once (any) name server learns mapping, it **caches** mapping. Cache entries timeout (disappear) after some time (TTL). If name host changes IP address, the name servers might not update until TTLs expire.

update/notify mechanisms proposed IETF standard RFC 2136

## 4.8.6 DNS Records

### Note 6: RR Format

(name, value, type, ttl)

**type=A** name is hostname, value is IP address

**type=NS** name is domain (e.g. google.com), value is hostname of authoritative name server for this domain

**type=CNAME** name is alias name for some "canonical" (the real) name (www.ibm.com is really servereast.backup2.ibm.com), value is canonical name

**type=MX** value is name of mailserver associated with name

## 4.8.7 Protocol

Query and reply messages both follow same format

Table 2: Protocol Layout	
2 bytes	2 bytes
identification	flags
# questions	# answer RRs
# authority RRs	# additional RRs
questions (variable # of questions)	answers (variable # of RRs)
authority (variable # of RRs)	additional info (variable # of RRs)

## 4.8.8 Attacking DNS

### DDoS attacks

- bombard root servers with traffic. Not successful to date, traffic filtering, local DNS servers cache protecting root DNS
- bombard TLD server. Potentially more dangerous

### Redirect Attacks

- man-in-middle (Intercept queries)
- DNS Poisoning (Send bogus replies to DNS server, which caches)

### Exploit DNS for DDoS

- send queries with spoofed source address: target IP
- requires amplification

## 4.9 File Distribution Time

### 4.9.1 Client-server

**Server Transmission:** must sequentially send (upload)  $N$  file copies. Time to send one copy:



$\frac{F}{u_s}$ . Time to send  $N$  copies:  $\frac{NF}{u_s}$

**Client:** each client must download file copy.  $d_{\min}$  = min client download rate. min client download time  $\frac{F}{d_{\min}}$

#### Note 7: Client-server File Distribution

time to distribute  $F$  to  $N$  clients using client-server approach

$$D_{c-s} \geq \max\left\{\frac{NF}{u_s}, \frac{F}{d_{\min}}\right\}$$

### 4.9.2 P2P

**Server Transmission:** must upload at least one copy. Time to send one copy:  $\frac{F}{u_s}$

**Client:** each client must download file copy. Min client download time:  $\frac{F}{d_{\min}}$

**Clients:** as aggregate must download  $NF$  bits. Max upload rate (limiting max download rate) is  $u_s + \sum u_i$

#### Note 8: P2P File Distribution

time to distribute  $F$  to  $N$  clients using P2P approach

$$D_{P2P} \geq \max\left\{\frac{F}{u_s}, \frac{F}{d_{\min}}, \frac{NF}{u_s + \sum u_i}\right\}$$

### 4.9.3 BitTorrent

File divided into 256Kb chunks

**Tracker:** tracks peers participating in torrent

**Torrent:** group of peers exchanging chunks of a file

## 4.10 Multimedia

### 4.10.1 Video

Coding: used redundancy **within** and **between** images to decrease # bits used to encode image

**Spatial:** within image

**Temporal:** from one image to next

**CBR** (constant bit rate): video encoding rate fixed

**VBR** (variable bit rate): video encoding rate changes as amount of spatial, temporal coding changes

### 4.10.2 DASH

DASH: Dynamic, Adaptive Streaming over HTTP

**Server:** Divides video file into multiple chunks.

Each chunk stored, encoded at different rates.

**Manifest file:** provides URLs for different chunks

**Client:** Periodically measures server-to-client bandwidth. Consulting manifest, requests one chunk at a time. Chooses maximum coding rate sustainable given current bandwidth. Can choose different coding rates at different points in time (depending on available bandwidth at time)

“intelligence” at client: client determines

- **when** to request chunk (so that buffer starvation, or overflow does not occur)
- **what encoding rate** to request (higher quality when more bandwidth available)
- **where** to request chunk (can request from URL server that is “close” to client or has high available bandwidth)

### 4.10.3 Content Distribution Networks (CDNs)

CDN stores copies of content at CDN nodes. Subscriber requests content from CDN, directed to nearby copy, retrieves content, may choose different copy if network path congested.

## 5 Chapter 3

### 5.1 Transport vs. Network Layer

**Network Layer:** logical communication between hosts

**Transport Layer:** logical communication between processes; relies on, enhances, network layer services

### 5.2 Multiplexing/demultiplexing

#### 5.2.1 How demultiplexing works

- host receives IP datagrams
  - each datagram as source IP address, destination IP address
  - each datagram carries one transport-layer segment
  - each segment has source, destination port number
- host uses **IP addresses and port numbers** to direct segment to appropriate socket

## Connectionless Demultiplexing

A UDP socket needs to have a local port number assigned to it (both client and server)

### Connection-oriented demux

TCP socket identified by 4-tuple: (**source IP address, source port number, dest IP address, dest port number**)

## 5.3 UDP

Table 3: UDP Segment Header

32 bits	
source port #	dest port #
length	checksum
application data (payload)	

### 5.3.1 UDP Checksum

#### Sender:

- treat segment contents, including header fields, as sequence of 16-bit integers
- checksum: addition (one's complement sum) of segment contents
- sender puts checksum value into UDP checksum field

#### Receiver:

- compute checksum of received segment
- check if computed checksum equals checksum field value

## 5.4 Pipelined Protocols

**Pipelining:** sender allows multiple, "in-flight", yet-to-be-acknowledged packets. Range of sequence numbers must be increased, buffering at sender and/or receiver.

### 5.4.1 Go-Back-N

- sender can have up to  $N$  unacked packets in pipeline
- receiver only sends **cumulative ack**. Doesn't ack packet if there's a gap
- sender has timer for oldest unacked packet. When timer expires, retransmit all unacked packets

### 5.4.2 Selective Repeat

- sender can have up to  $N$  unacked packets in pipeline
- receiver sends **individual ack** for each packet
- sender maintains timer for each unacked packet. When timer expires, retransmit only that unacked packet

## 5.5 TCP Segment Structure

TCP contains a handshake to make sure both ends are willing to open a connection

Table 4: TCP Segment Structure

32 bits	
source port #	dest port #
sequence number	
acknowledgment number	
(head len, not used, UAPRSF)	receive window
checksum	urg data pointer
options (variable length)	
application data (variable length)	

**sequence number, acknowledgment number:** counting by bytes of data (not segments)  
**U:** urgent data (generally not used)  
**A:** ACK # valid  
**P:** push data now (generally not used)  
**RSF:** RST, SYN, FIN; connection established (setup, teardown commands)  
**checksum:** Internet checksum (as in UDP)  
**receive window:** # bytes receiver willing to accept

**Sequence Numbers:** byte stream "number" of first byte in segment's data

**Acknowledgements:** sequence # of next byte expected from other side, cumulative ACK

### 5.5.1 TCP Round Trip Time, Timeout

$$E = (1 - \alpha) \times E + \alpha \times \text{SampleRTT}$$

Where  $E$  is EstimatedRTT. Influence of past sample decreases exponentially fast. Typical value:  $\alpha = 0.125$

$$\text{TimeoutInterval} = E + 4 \times \text{DevRTT}$$

Where  $\text{DevRTT}$  is the safety margin ( $\text{DevRTT} = (1 - \beta) \times \text{DevRTT} + \beta \times |\text{SampleRTT} - E|$  (typically,  $\beta = 0.25$ ))



### 5.5.2 TCP Flow Control

- receiver “advertises” free buffer space by including `rwnd` value in TCP header of receiver-to-sender segments
  - `RcvBuffer` size set via socket options (typical default is 4096 bytes)
  - many operating systems autoadjust `RcvBuffer`
- sender limits amount of unacked (“in-flight”) data to receiver’s `rwnd` value
- guarantees receive buffer will not overflow

### 5.5.3 Closing

- client, server each close their side of connection (send TCP segment with FIN bit 1)
- respond to received FIN with ACK (on receiving FIN, ACK can be combined with own FIN)
- simultaneous FIN exchanges can be handled

## 5.6 TCP Congestion Control

**Approach:** sender increases transmission rate (window size), probing for usable bandwidth, until loss occurs

**Additive Increase:** increase `cwnd` by 1 MSS every RTT until loss detected

**Multiplicative Decrease:** cut `cwnd` in half after loss

## 5.7 Fairness

TCP is fair because:

- additive increase gives slope of 1, as throughput increases
- multiplicative decrease decreases throughput proportionally

UDP is not fair:

- do not want rate throttled by congestion control
- send audio/video at constant rate, tolerate packet loss

## 5.8 Explicit Congestion Notification

Network-assisted Congestion Control:

- two bits in IP header (ToS field) marked by **network router** to indicate congestion
- congestion indication carried to receiving host
- receiver (seeing congestion indication in IP datagram) sets ECE bit on receiver-to-sender ACK segment to notify sender of congestion

# 6 Chapter 4

## 6.1 Network Layer

- transport segment from sending to receiving host
- on sending side encapsulates segments into datagrams
- on receiving side, delivers segments to transport layer
- network layer protocols in **every** host, router
- router examines header fields in all IP datagrams passing through it

### 6.1.1 Network Layer Functions

**Forwarding:** move packets from router’s input to appropriate router output

**Routing:** determine route taken by packets from source to destination (*routing algorithms*)

### 6.1.2 Data Plane, Control Plane

#### Data Plane

- local, per-router function
- determines how datagram arriving on router input port is forwarded to router output port
- forwarding function

#### Control plane

- network-wide logic
- determines how datagram is routed among routers along end-end path from source host to destination host

- two control-plane approaches:

**Traditional Routing Algorithms:** implemented in routers

**Software-defined networking (SDN):** implemented in (remote) servers

## 6.2 Router Forwarding

**Destination-based forwarding:** forward based only on destination IP address (traditional)

**Generalized forwarding:** forward based on any set of header field values

### 6.2.1 Destination-based forwarding

A link interface is assigned to a range of destination address ranges

#### Note 9: Longest Prefix Matching

When looking for forwarding table entry for given destination address, use **longest** address prefix that matches destination address. Longest prefix matching: often performed using ternary content addressable memories (TCAMs). Cisco Catalyst can hold up  $\approx 1\text{M}$  routing table entries in TCAM.

**Content Addressable:** present address to TCAM; retrieve address in one clock cycle, regardless of table size

### 6.2.2 Switching Fabrics

- transfer packet from input buffer to appropriate output buffer
- switching rate: rate at which packets can be transferred from inputs to outputs (often measured as multiple of input/output line rate,  $N$  inputs: switching rate  $N$  times line rate desirable)
- three types of switching fabrics

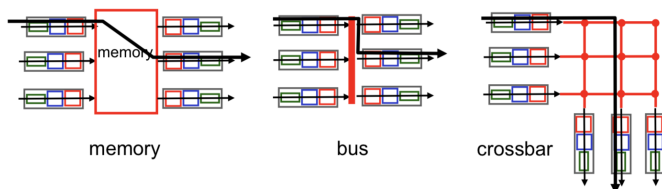


Figure 2: Different Types of Switching Fabrics

#### Switching via Memory

- traditional computers with switching under direct control of CPU
- packet copied to system's memory
- speed limited memory bandwidth (2 bus crossing per datagram)

#### Switching via a Bus

- datagram from input port memory to output port memory via a shared bus
- **bus contention:** switching speed limited by bus bandwidth
- 32 Gbps bus, Cisco 5600: sufficient speed for access and enterprise routers

#### Switching via Interconnection Network

- overcome bus bandwidth limitations
- banyan networks, crossbar, other interconnection nets initially developed to connect processors in multiprocessor

- advanced design: fragmenting datagram into fixed length cells, switch cells through the fabric
- Cisco 12000: switches 60 Gbps through the interconnection network

### 6.2.3 Input port queuing

- fabric slower than input ports combined  $\rightarrow$  queuing may occur at input queues (queuing delay and loss due to input buffer overflow)
- **Head-of-the-Line (HOL) blocking:** queued datagram at front of queue prevents others in queue from moving forward

### 6.2.4 Output ports

- **Buffering** required from fabric faster rate (Datagram (packets) can be lost due to congestion, lack of buffers)
- **Scheduling** datagrams (Priority scheduling – who gets best performance, network neutrality)

#### Note 10: How much buffering?

RFC 3439 rule of thumb: average buffering equal to “typical” RTT (say 250 msec) times link capacity  $C$  (e.g.  $C = 10$  Gbps link, 2.5 Gbit buffer). Recent recommendation with  $N$  flows, buffering equal to

$$\frac{RTT \times C}{\sqrt{N}}$$

### 6.2.5 Scheduling Mechanisms

**Scheduling:** choose next packet to send on link

**FIFO scheduling:** send in order of arrival to queue

**discard policy:** if packet arrives to full queue, who to discard

**tail drop:** drop arriving packet

**priority:** drop/remove on priority basis

**random:** drop/remove randomly

**priority scheduling:** send highest priority queued packet. Multiple *classes*, with different priorities (class may depend on marking or other header info, e.g. IP source/dest, port number, etc)

**RR scheduling:** multiple classes. Cyclically scan class queues, sending one complete packet from each class (if available)

**WFQ scheduling:** generalized Round Robin.  
Each class gets weighted amount of service in each cycle

- address format: a.b.c.d/x, where x is # bits in subnet portion of address

## 6.3 IP

### 6.3.1 IP Datagram Format

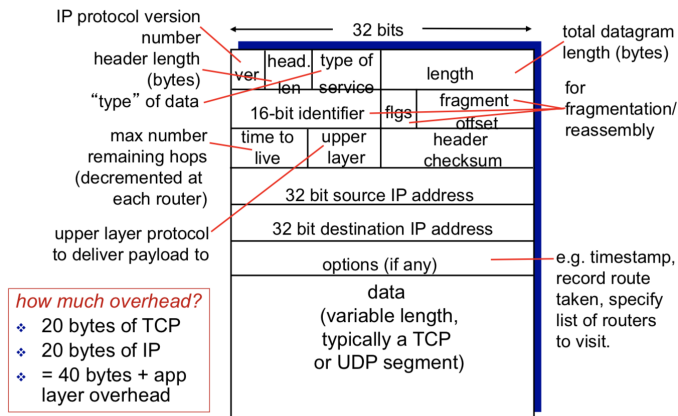


Figure 3: IP Datagram Format

### 6.3.2 IP Fragmentation, Reassembly

Large IP datagram divided ("fragmented") within net

- one datagram becomes several datagrams
- "reassembled" only at final datagrams
- IP header bits used to identify, order related fragments

### 6.3.3 IP Addressing

**IP Address:** 32-bit identifier for host, router interface

**interface:** connection between host/router and physical link. Router's typically have multiple interfaces

### 6.3.4 Subnets

**Subnet part** – high order bits. **Host part** – low order bits

- device interfaces with same subnet part of IP address
- can physically reach each other **without intervening router**
- to determine the subnets, detach each interface from its host or router, creating islands of isolated networks
- each isolated network is called a **subnet**

CIDR: Classless InterDomain Routing

- subnet portion of address of arbitrary length

### 6.3.5 DHCP: Dynamic Host Configuration Protocol

**Goal:** allow host to *dynamically* obtain its IP address from network server when it joins network

- can renew its lease on address in use
- allows reuse of addresses (only hold address while connected)
- support for mobile users who want to join network (more shortly)

DHCP overview:

- host broadcasts "DHCP discover" msg [*optional*]
- DHCP server responds with "DHCP offer" msg [*optional*]
- host requests IP address: "DHCP request" msg
- DHCP server sends address: "DHCP ack" msg

DHCP can return more than just allocated IP address on subnet:

- address of first-hop router for client
- name and IP address of DNS server
- network mask (indicating network versus host portion of address)

### 6.3.6 ICANN

- allocates addresses
- manages DNS
- assigns domain names, resolves disputes

### 6.3.7 NAT

All datagrams **leaving** local network have **same** single source NAT IP address

**Motivation:** local network uses just one IP address as far as outside world is concerned

- range of addresses not needed from ISP: just one IP address for all devices
- can change addresses of devices in local network without notifying outside world
- can change ISP without changing addresses of devices in local network
- devices inside local net not explicitly addressable, visible by outside world (a security plus)

**Implementation:** NAT router must

**Outgoing datagrams:** replace (source IP address, port #) of every outgoing datagram to (NAT IP address, new port #) ... remote clients/servers will respond using (NAT IP address, new port #) as destination address

**Remember (in NAT translation table)** every (source IP address, port #) to (NAT IP address, new port #) translation pair

**Incoming datagrams:** replace (NAT IP address, new port #) in destination fields of every incoming datagram with corresponding (source IP address, port #) stored in NAT table

- 16-bit port-number field: 60,000 simultaneous connections with a single LAN-side address
- NAT is controversial:
  - routers should only process up to layer 3
  - address shortage should be solved by IPv6
  - violates end-to-end argument (NAT possibility must be taken into account by app designers, e.g. P2P applications)
  - NAT traversal: what if client wants to connect to server behind NAT?

### 6.3.8 IPv6

32-bit address space soon to be completely allocated

Additionally:

- header format helps speed processing/forwarding
- header changes to facilitate QoS

**IPv6 datagram format:**

- fixed-length 40 byte header
- no fragmentation allowed

### 6.3.9 IPv6 Datagram Format

**Priority:** identify priority among datagrams in flow

**Flow Label:** identify datagrams in same “flow” (concept of “flow” not well defined)

**Next Header:** identify upper layer protocol for data

Table 5: IPv6 Format

32 bits					
version	pri	flow label			
payload len	next hdr	hop limit			
source address (128 bits)					
destination address (128 bits)					
data					

### 6.3.10 Other changes from IPv4

**checksum:** removed entirely to reduce processing time at each hop

**options:** allowed, but outside of header, indicated by “Next Header” field

**ICMPv6:** new version of ICMP (additional message types e.g. “Packet Too Big”, multicast group management functions)

### 6.3.11 Transition from IPv4 to IPv6

- not all routers can be upgraded simultaneously (no “flag days”, how will network operate with mixed IPv4 and IPv6 routers)
- **tunneling:** IPv6 datagram carried as *payload* in IPv4 datagram among IPv4 routers

## 6.4 Generalized Forwarding and SDN

Each router contains a **flow table** that is computed and distributed by a logically centralized routing controller

### 6.4.1 OpenFlow data plane abstraction

**Flow:** defined by header fields

**Generalized Forwarding:** simple packet-handling rules

**Pattern:** match values in packet header fields

**Actions:** for matched packet: drop, forward, modify, matched packet and send matched packet to controller

**Priority:** disambiguate overlapping patterns

**Counters:** # bytes and # packets

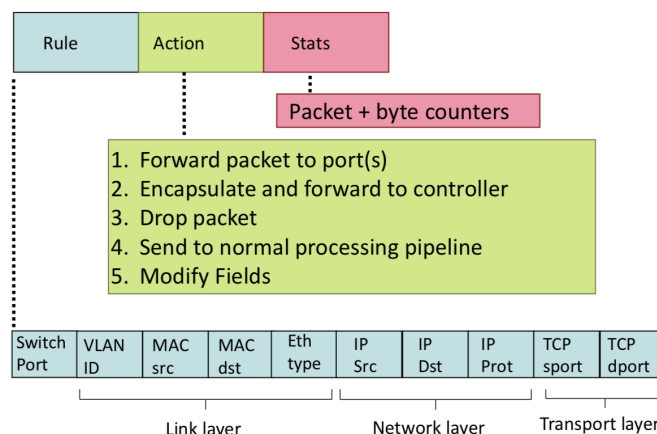


Figure 4: Flow Table Entries

## 6.4.2 OpenFlow Abstraction

- **Match+Action:** unifies different kinds of devices
- Router
  - match:** longest destination IP prefix
  - action:** forward out a link
- Switch
  - match:** destination MAC address
  - action:** forward or flood
- Firewall
  - match:** IP addresses and TCP/UDP port numbers
  - action:** permit or deny
- NAT
  - match:** IP address and port
  - action:** rewrite address and port

- router knows physically-connected neighbors, link costs to neighbors
- iterative process of computation, exchange of info with neighbors
- **“distance vector” algorithms**

## 7.2.2 Static or Dynamic

### Static:

- routes change slowly over time

### Dynamic:

- routes change more quickly (periodic update, in response to link cost changes)

# 7 Chapter 5

## 7.1 Control Plane

Two approaches to structuring network control plane:

### 7.1.1 Per-router control plane

Individual routing algorithm components in **each and every router** interact with each other in control plane to compute forwarding tables

### 7.1.2 Logically centralized control plane

A distinct (typically remote) controller interacts with local control agents (CAs) in routers to compute forwarding tables

## 7.2 Routing Protocols

**Routing protocol goal:** determine “good” paths (equivalently, routes), from sending hosts to receiving host, through network of routers  
**path:** sequence of routers packets will traverse in going from given initial source host to given final destination host  
**“good”:** least “cost”, “fastest”, “least congested”

### 7.2.1 Global or Decentralized information

#### Global:

- all routers have complete topology, link cost info
- **“link state” algorithms**

#### Decentralized:

### 7.2.3 Link-State Routing Algorithm

#### Note 11: Dijkstra’s algorithm

- net topology, link costs known to all nodes
  - accomplished via “link state broadcast”
  - all nodes have same info
- computes least cost paths from one node (‘source’) to all other nodes (gives **forwarding table** for that node)
- iterative: after  $k$  iterations, know least cost path to  $k$  destinations

#### Notation:

$c(x, y)$ : link cost from node  $x$  to  $y$ ;  $= \infty$  if not direct neighbors

$D(v)$ : current value of cost path from source to destination  $v$

$p(v)$ : predecessor node along path from source to  $v$

$N'$ : set of nodes whose least cost path definitively known

#### Algorithm Complexity: $n$ nodes

- each iteration: need to check all nodes,  $W$ , not in  $N'$
- $\frac{n(n+1)}{2}$  comparisons:  $O(n^2)$
- more efficient implementations possible:  $O(n \log n)$

## 7.2.4 Distance Vector Algorithm

### Note 12: Bellman-Ford equation

(dynamic programming)

let  $d_x(y) :=$  cost of least-cost path from  $x$  to  $y$   
then

$$d_x(y) = \min\{c(x, v), d_v(y)\}$$

$c(x, v)$ : cost to neighbor  $v$

$d_v(y)$ : cost from neighbor  $v$  to destination  $y$

- $D_x(y)$  = estimate of least cost from  $x$  to  $y$  ( $x$  maintains distance vector  $\mathbf{D}_x = [D_x(y) : y \in N]$ )
- node  $x$ :
  - knows cost to each neighbor  $v : c(x, v)$
  - maintains its neighbor's distance vectors. For each neighbor  $v$ ,  $x$  maintains  $\mathbf{D}_v = [D_v(y) : y \in N]$

#### key idea:

- from time-to-time, each node sends its own distance vector estimate to neighbors
- when  $x$  receives new DV estimate from neighbor, it updates its own DV using B-F equation:

$$D_x(y) \rightarrow \min\{c(x, v) + D_v(y)\} \text{ for each node } y \in N$$

## 7.2.5 Comparison of LS and DV algorithms

### Message Complexity:

**LS:** with  $n$  nodes,  $E$  links,  $O(nE)$  msgs sent

**DV:** exchange between neighbors only (convergence time varies)

### Speed of Convergence:

**LS:**  $O(n^2)$  algorithm requires  $O(nE)$  msgs (may have oscillations)

**DV:** convergence time varies (may be routing loops, count-to-infinity problem)

**Robustness:** What happens if router malfunctions?

**LS:** Node can advertise incorrect *link* cost. Each node computes only its *own* table

**DV:** DV node can advertise incorrect *path* cost. Each node's table used by others, error propagate through network

## 7.3 Making Routing Scalable

At the moment, can't store all destinations in routing tables. Routing table exchange would swamp links. Solution: Aggregate routers into regions known as "autonomous systems" (AS) (a.k.a "domains")

### intra-AS routing:

- routing among hosts, routers in same AS ("network")
- all routers in AS must run **same** intra-domain protocol
- routers in *different* AS can run *different* intra-domain routing protocol
- gateway router: at "edge" of its own AS, has link(s) to router(s) in other AS'es

### inter-AS routing

- routing among AS'es
- gateways perform inter-domain routing (as well as intra-domain routing)

## 7.3.1 Interconnected ASes

Forwarding table configured by both intra-AS and inter-AS routing algorithm

- intra-AS routing determine entries for destinations within AS
- inter-AS and intra-AS determine entries for external destinations

## 7.3.2 Intra-AS Routing

Also known as **Interior Gateway Protocols (IGP)**. Most common intra-AS routing protocols:

**RIP:** Routing Information Protocol

**OSPF:** Open Shortest Path First (IS-IS protocol essentially same as OSPF)

**IGRP:** Interior Gateway Routing Protocol (Cisco proprietary for decades, until 2016)



### Note 13: OSPF (Open Shortest Path First)

- “open”: publicly available
- uses link-state algorithm
  - link state packet dissemination
  - topology map at each node
  - route computation using Dijkstra’s algorithm
- router floods OSPF link-state advertisements to all other routers in **entire** AS
  - carried in OSPF messages directly over IP (rather than TCP or UDP)
  - link state: for each attached link
- **IS-IS routing** protocol: nearly identical to OSPF

#### “Advanced Features”

- **security**: all OSPF messages authenticated (to prevent malicious intrusion)
- **multiple** same-cost **paths** allowed (only one path in RIP)
- for each link, multiple cost metrics for different **TOS** (e.g. satellite link cost set low for best effort ToS; high for real-time ToS)
- integrated uni- and **multi-cast** support (Multicast OSPF (MOSPF) uses same topology data base as OSPF)
- **hierarchical** OSPF in large domains

### 7.3.3 Hierarchical OSPF

**two-level hierarchy**: local area, backbone

- link-state advertisements only in area
- each nodes has detailed area topology; only know direction (shortest path) to nets in other areas

**area border routers**: “summarize” distances to nets in own area, advertise to other Area Border routers

**backbone routers**: run OSPF routing limited to backbone

**boundary routers**: connect to other AS’es

### 7.3.4 Internet inter-AS routing

#### Note 14: BGP (Border Gateway Protocol)

- The de facto inter-domain routing protocol
- BGP provides each AS a means to:
  - **eBGP**: obtain subnet reachability information from neighboring ASes
  - **iBGP**: propagate reachability information to all AS-internal routers
  - determine “good” routes to other networks based on reachability information and **policy**
- allows subnet to advertise its existence to rest of Internet: “I am here”

## 7.4 Software Defined Networking (SDN)

Internet network layer: historically has been implemented via distributed, per-router approach

- **monolithic** router contains switching hardware, runs proprietary implementation of Internet standard protocols (IP, RIP, IS-IS, OSPF, BGP) in proprietary router OS (e.g. Cisco IOS)
- different “middleboxes” for different network layer functions: firewalls, load balancers, NAT boxes, ...

Why a logically centralized control plane?

- easier network management: avoid router misconfiguration, greater flexibility of traffic flows
- table-based forwarding (recall OpenFlow API) allows “programming” routers
  - centralized “programming” easier: compute tables centrally and distribute
  - distributed “programming” more difficult: compute tables as result of distributed algorithm (protocol) implemented in each and every router
- open (non-proprietary) implementation of control plane

### 7.4.1 SDN perspective: Data Plane Switches

- fast, simple, commodity switches implementing generalized data-plane forwarding in hardware
- switch flow table computed, installed by controller

- API for table-based switch control (e.g. OpenFlow) – defines what is controllable and what is not
- protocol for communicating with controller (e.g. OpenFlow)

Fortunately, network operators don't "program" switches by creating/sending OpenFlow messages directly. Instead use higher-level abstraction at controller

#### 7.4.2 SDN perspective: SDN controller

- maintain network state information
- interacts with network control applications "above" via northbound API
- interacts with network switches "below" via southbound API
- implemented as distributed system for performance, scalability, fault-tolerance, robustness

#### 7.4.3 SDN perspective: Control Applications

- "brains" of control: implement control functions using lower-level services, API provided by SDN controller
- unbundled: can be provided by 3<sup>rd</sup> party: distinct from routing vendor, or SDN controller

##### Note 15: OpenFlow Protocol

- operates between controller, switch
- TCP used to exchange messages (optional encryption)
- three classes of OpenFlow messages:
  - controller-to-switch
  - asynchronous (switch to controller)
  - symmetric (misc)

#### 7.4.4 OpenFlow: controller-to-switch messages

##### Key controller-to-switch messages

**features:** controller queries switch features, switch replies

**configure:** controller queries/sets switch configuration parameters

**modify-state:** add, delete, modify flow entries in the OpenFlow tables

**packet-out:** controller can send this packet out of specific switch port

**packet-in:** transfer packet (and its control) to controller. See packet-out message from controller

**flow-removed:** flow table entry deleted at switch

**port status:** inform controller of a change on a port

### 7.5 OpenDaylight (ODL) controller

- ODL Lithium controller
- network apps may be contained within, or be external to SDN controller
- Service Abstraction Layer: interconnects internal, external applications and services

### 7.6 ONOS controller

- control apps separate from controller
- intent framework: high-level specification of service: what rather than how
- considerable emphasis on distributed core: service reliability, replication performance scaling

### 7.7 ICMP: Internet Control Message Protocol

Table 6: ICMP Types and Codes

Type	Code	Description
0	0	echo reply (ping)
3	0	destination network unreachable
3	1	destination host unreachable
3	2	destination protocol unreachable
3	3	destination port unreachable
3	6	destination network unknown
3	7	destination host unknown
4	0	source quench (congestion control – not used)
8	0	echo request (ping)
9	0	route advertisement
10	0	router discovery
11	0	TTL expired
12	0	bad IP header

- used by hosts and routers to communicate network-level information
  - error reporting: unreachable host, network, port, protocol
  - echo request/reply (used by ping)
- network-layer "above" IP: ICMP messages carried in IP datagrams



- **ICMP message:** type, code, plus first 8 bytes of IP datagram causing error

## 7.8 Network Management and SNMP

“Network management includes the deployment, integration and coordination of the hardware, software, and human elements to monitor, test, poll, configure, analyze, evaluate, and control the network and element resources to meet the real-time, operational performance, and Quality of Service requirements at a reasonable cost.”

**Managed devices** contain **managed objects** whose data is gathered into a **Management Information Base (MIB)**

### 7.8.1 SNMP Protocol: Message Types

Table 7: SNMP Message Types

Message type	Function
GetRequest GetNextRequest GetBulkRequest	manager-to-agent: “get me data” (data instance, next data in list, block of data)
InformRequest	manager-to-manager: here’s MIB value
SetRequest	manager-to-agent: set MIB value
Response	agent-to-manager: value, response to request
Trap	agent-to-manager: inform manager of exceptional event

Table 8: SNMP Message Formats

← SNMP PDU →							
PDU Type	← Request ID	Get/set header	→ Error Index	← Name	Variables to get/set	→	
(0 - 3)		Error Status (0 - 5)			Value	Name	Value
PDU Type	← Enterprise	Agent Addr	Trap header	Specific code	→ Timestamp	← Trap info	→
4		(0 - 7)	Trap Type			Name	Value

## 8 Chapter 6

Link layer is implemented in the adapter (aka **network interface card (NIC)**)

### 8.1 Link Layer

- hosts and routers: **nodes**
- communication channels that connect adjacent nodes along communication path: **links**
- layer-2 packet: **frame**, encapsulates datagram

#### Note 16: Link Layer Introduction

**data-link layer** has responsibility of transferring datagram from one node to **physically adjacent** node over a link

### 8.2 Error Detection

#### 8.2.1 Parity Checking

**single bit parity:** detect single bit errors

**two-dimensional bit parity:** detect and correct single bit errors

#### 8.1.1 Link layer services

- **framing, link access**
  - encapsulate datagram into frame, adding header, trailer
  - channel access if shared medium
  - “MAC” addresses used in frame headers to identify source, destination (different from IP address!)
- **reliable delivery between adjacent nodes**
  - seldom used on low bit-error link (fiber, some twisted pair)
  - wireless links: high error rates
- **Flow Control:** pacing between adjacent sending and receiving nodes
- **error detection**
  - errors caused by signal attenuation, noise
  - receiver detects presence of errors: signals sender for retransmission or drops frame
- **error correction:** receiver identifies and **corrects** bit error(s) without resorting to retransmission
- **half-duplex and full-duplex:** with half duplex, nodes at both ends of link can transmit, but not at same time

#### 8.2.2 Cyclic Redundancy Check

- more powerful error-detection coding
- view data bits,  $D$ , as a binary number
- choose  $r + 1$  bit pattern (generator),  $G$
- goal: choose  $r$  CRC bits,  $R$ , such that
  - $\langle D, R \rangle$  exactly divisible by  $G$  (modulo 2)
  - receiver knows  $G$ , divides  $\langle D, R \rangle$  by  $G$ . If non-zero remainder: error detected!
  - can detect all burst errors less than  $r + 1$  bits
- widely used in practice (Ethernet, 802.11 WiFi, ATM)

### 8.3 Multiple access links, protocols

- point-to-point
  - PPP for dial-up access
  - point-to-point link between Ethernet switch, host
- broadcast (shared wire or medium)
  - old-fashioned Ethernet
  - upstream HFC

### Note 17: Multiple Access Protocol

- distributed algorithm that determines how nodes share channel, i.e. determine when node can transmit
- communication about channel sharing must use channel itself! (no out-of-band channel for coordination)

Given a broadcast channel of rate  $R$  bps. An ideal multiple access protocol needs:

- when one node wants to transmit, it can send at rate  $R$
- when  $M$  nodes want to transmit, each can send at average rate  $\frac{R}{M}$
- fully decentralized:
  - no special node to coordinate transmissions
  - no synchronization of clocks, slots
- simple

## 8.4 MAC Protocols

**Channel Partitioning:** divide channel into smaller “pieces” (time slots, frequency, code). Allocate piece to node for exclusive use

**Random Access:** channel not divided, allow collisions, “recover” from collisions

**“Taking Turns”:** nodes take turns, but nodes with more to send can take longer turns

### 8.4.1 TDMA: time division multiple access

- access to channel in “rounds”
- each station gets fixed length slot (length = packet transmission time) in each round
- unused slots go idle

### 8.4.2 FDMA: frequency division multiple access

- channel spectrum divided into frequency bands
- each station assigned fixed frequency band
- unused transmission time in frequency bands go idle

## 8.5 Random Access Protocols

- when node has packet to send
  - transmit at full channel data rate  $R$
  - no examination between nodes prior
- two or more transmitting nodes → “collision”
- **random access MAC protocol** specifies:

- how to detect collisions
- how to recover from collisions (e.g. via delayed retransmissions)
- examples of random access MAC protocols:
  - slotted ALOHA
  - ALOHA
  - CSMA, CSMA/CD, CSMA/CA

### 8.5.1 Slotted ALOHA

#### Assumptions:

- all frames same size
- time divided into equal size slots (time to transmit 1 frame)
- nodes start to transmit only slot beginning
- nodes are synchronized
- if 2 or more nodes transmit in slot, all nodes detect collision

#### Operation:

- when node obtains fresh frame, transmits in next slot
  - if no collision: node can send new frame in next slot
  - if collision: node retransmits frame in each subsequent slot with probability  $p$  until success

#### Pros:

- single active node can continuously transmit at full rate of channel
- highly decentralized: only slots in nodes need to be in sync
- simple

#### Cons:

- collisions, wasting slots
- idle slots
- nodes may be able to detect collision in less than time to transmit packet
- clock synchronization

### 8.5.2 Pure (unslotted) ALOHA

- unslotted Aloha: simpler, no synchronization
- when frame first arrives → transmit immediately
- collision probability increases: frame sent at  $t_0$  collides with other frames sent in  $[t_0 - 1, t_0 + 1]$

### 8.5.3 CSMA (carrier sense multiple access)

**CSMA:** listen before transmit

- **if channel sensed idle:** transmit entire frame
- **if channel sensed busy:** defer transmission

- **collisions can still occur:** propagation delay means two nodes may not hear each other's transmission
- **collision:** entire packet transmission time wasted (distance and propagation delay play role in determining collision probability)

#### 8.5.4 CSMA/CD (collision detection)

**CSMA/CD:** carrier sensing, deferral as in CSMA

- collisions *detected* within short time
  - colliding transmissions aborted, reducing channel wastage
  - collision detection:
    - easy in wired LANs: measure signal strengths, compare transmitted, received signals
    - difficult in wireless LANs: received signal strength overwhelmed by local transmission strength
1. NIC receives datagram from network layer, create frame
  2. If NIC sense channel idle, starts frame transmission. If NIC sense channel busy, waits until channel idle, then transmits
  3. If NIC transmits entire frame without detecting another transmission, NIC is done with frame
  4. If NIC detects another transmission while transmitting, aborts and sends jam signal
  5. After aborting, NIC enter **binary (exponential) backoff**:
    - after  $m$ th collision, NIC chooses  $K$  at random from  $\{0, 1, 2, \dots, 2^m - 1\}$ . NIC waits  $K \cdot 512$  bit times, returns to Step 2
    - longer backoff interval with more collisions

■ Better performance than ALOHA

## 8.6 “Taking turns” MAC protocols

**Channel partitioning MAC protocols:**

- share channel efficiently and fairly at high load
- inefficient at low load: delay in channel access,  $\frac{1}{N}$  bandwidth allocated even if only 1 active node

**random access MAC protocols**

- efficient at low load: single node can fully utilize channel
- high load: collision overhead

**“taking turns” protocols:** look for best of both worlds!

#### 8.6.1 Polling

- master node “invites” slave nodes to transmit in turn
- typically used with “dumb” slave devices
- concerns: polling overhead, latency, single point of failure (master)

#### 8.6.2 Token Passing

- control **token** passed from one node to next sequentially
- token message
- concerns: token overhead, latency, single point of failure (token)

## 8.7 Cable Access Network

#### 8.7.1 DOCSIS: Data Over Cable Service Interface Spec

- FDM over upstream, downstream frequency channels
- TDM upstream: some slots assigned, some have contention
  - downstream MAP frame: assigns upstream slots
  - request for upstream slots (and data) transmitted random access (binary backoff) in selected slots

## 8.8 MAC Addresses and ARP

32-bit IP address:

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

MAC (or LAN or physical or Ethernet) address:

- function: used ‘locally’ to get frame from one interface to another physically-connected interface (same network, in IP-addressing sense)
- 48 bit MAC address (for most LANs) burned in NIC ROM, also sometimes software settable

#### 8.8.1 LAN Address

- MAC address allocation administered by IEEE
- manufacturer buys portion of MAC address space (to assure uniqueness)
- MAC flat address → portability (can move LAN card from one LAN to another)

- IP hierarchical address not portable (address depends on IP subnet to which node is attached)

## 8.8.2 ARP: Address Resolution Protocol

**ARP table:** each IP node (host, router) on LAN has table

- IP/MAC address mappings for some LAN nodes: <IP address; MAC address; TTL>
- TTL (Time To Live): time after which address mapping will be forgotten (typically 20 mins)

## 8.9 Ethernet

### 8.9.1 Physical Topology

**bus:** popular through mid 90s (all nodes in same collision domain (can collide with each other))

**star:** prevails today (active **switch** in center, each “spoke” runs a (separate) Ethernet protocol (nodes do not collide with each other))

### 8.9.2 Ethernet frame structure

Table 9: Ethernet Frame Structure					
7b	6b	6b			
preamble	dest addr	source addr	type	data (payload)	CRC

**preamble:** 7 bytes with pattern 10101010 followed by one byte with pattern 10101011. Used to synchronize receiver, sender clock rates

**addresses:** 6 byte source, destination MAC addresses

- if adapter receives frame with matching destination address, or with broadcast address (e.g. ARP packet), it passes data in frame with network layer protocol
- otherwise, adapter discards frame

**type:** indicated higher layer protocol (mostly IP but others possible, e.g. Novell IPX, AppleTalk)

**CRC:** cyclic redundancy check at receiver (error detected: frame is dropped)

### 8.9.3 Ethernet: unreliable, connectionless

**connectionless:** no handshaking between sending and receiving NICs

**unreliable:** receiving NIC doesn't send acks or nacks to sending NIC (data in dropped frames recovered only if sender uses higher layer rdt (e.g. TCP), otherwise dropped data lost)

Ethernet's MAC protocol: unslotted **CSMA/CD with binary backoff**

### 8.9.4 Ethernet switch

**link-layer device:** takes an active role

- store, forward Ethernet frames
- examine incoming frame's MAC address, **selectively** forward frame to one-or-more outgoing links when frame is to be forwarded on segment, uses CSMA/CD to access segment

**transparent:** hosts are unaware of presence of switches

**plug-and-play, self-learning:** switches do not need to be configured

## 8.10 Switches vs routers

both are store-and-forward:

**routers:** network-layer devices (examine network-layer headers)

**switches:** link-layer devices (examine link-layer headers)

both have forwarding tables:

**routers:** compute tables using routing algorithms, IP addresses

**switches:** learn forwarding table using flooding, learning, MAC addresses

## 8.11 VLANs

### Note 18: Virtual Local Area Network

switch(es) supporting VLAN capabilities can be configured to defined multiple **virtual** LANs over single physical LAN infrastructure

### 8.11.1 Port-based VLAN

switch ports grouped (by switch management software) so that **single** physical switch operates as **multiple** virtual switches

**traffic isolation:** frames to/from ports 1-8 can *only* reach ports 1-8 (can also define VLAN based on MAC addresses of endpoints, rather than switch port)

**dynamic membership:** ports can be dynamically assigned among VLANs

**forwarding between VLANs:** done via routing (just as with separate switches). In practice vendors sell combined switches plus routers

### VLANs spanning multiple switches

**trunk port:** carries frames between VLANs defined over multiple physical switches

- frames forwarded within VLAN between switches can't be vanilla 802.1 frames (must carry VLAN ID info)
- 802.1q protocol adds/removed additional header fields for frames forwarded between trunk ports

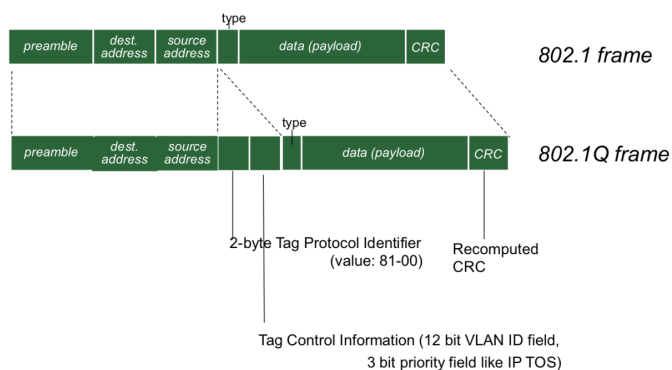


Figure 5: 802.1Q VLAN frame format

### 9.1.2 Ad Hoc mode

- no base stations
- nodes can only transmit to other nodes within link coverage
- nodes organize themselves into a network: route among themselves

Table 10: Wireless Network Taxonomy

	single hop	multiple hops
<b>infrastructure (e.g. APs)</b>	host connects to base station (WiFi, WiMAX, cellular) which connects to larger Internet	host may have to relay through several wireless nodes to connect to larger Internet: mesh net
<b>no infrastructure</b>	no base station, no connection to larger Internet (Bluetooth, ad hoc nets)	no base station, no connection to larger Internet. May have to relay to reach another given wireless node. MANET, VANET

## 9 Chapter 7

### 9.1 Wireless

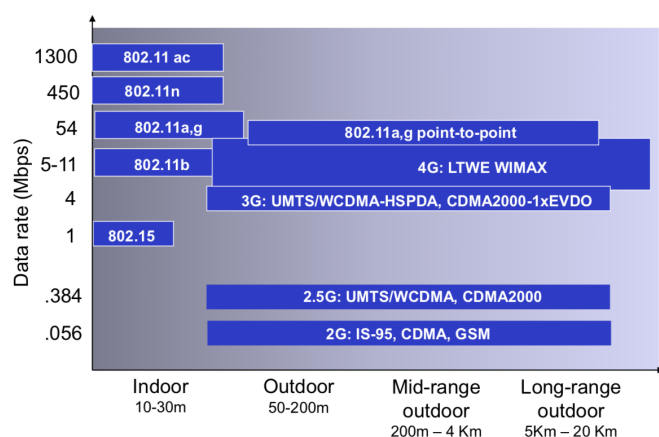


Figure 6: Characteristics of selected wireless links

#### 9.1.1 Infrastructure mode

- base station connects mobiles into wired network
- handoff: mobile changes base station providing connection into wired network

#### 9.1.3 Wireless Link Characteristics

**important** difference from wired link...

**decreased signal strength:** radio signal attenuates as it propagates through matter (path loss)

**interference from other sources:** standardized wireless network frequencies (e.g. 2.4GHz) shared by other devices (e.g. phone); devices (motors) interfere as well

**multipath propagation:** radio signal reflects off objects ground, arriving at destination at slightly different times

...make communication across (even a point to point) wireless link much more "difficult"

**SNR:** signal-to-noise ratio. Larger SNR – easier to extract signal from noise (a "good thing")

#### SNR versus BER tradeoffs

**given physical layer:** increase power → increase SNR → decrease BER

**given SNR:** choose physical layer that meets BER requirement, giving highest throughput (SNR may change with mobility: dynamically adapt physical layer (modulation technique, rate))

### 9.1.4 Code Division Multiple Access (CDMA)

- unique “code” assigned to each user; i.e. code set partitioning
  - all users share same frequency, but each user has own “chipping” sequence (i.e. code) to encode data
  - allows multiple users to “coexist” and transmit simultaneously with minimal interference (if codes are “orthogonal”)
- **encoded signal** = (original data) × (chipping sequence)
- **decoding**: inner-product of encoded signal and chipping sequence

## 9.2 IEEE 802.11 Wireless LAN

### 802.11b

- 2.4 - 5 GHz unlicensed spectrum
- up to 11 Mbps
- direct sequence spread spectrum (DSSS) in physical layer (all hosts use same chipping code)

### 802.11a

- 5 - 6 GHz range
- up to 54 Mbps

### 802.11g

- 2.4 - 5 GHz
- up to 54 Mbps

### 802.11n: multiple antennae

- 2.4 - 5 GHz range
- up to 200 Mbps

All use CSMA/CA for multiple access. All have base-station and ad-hoc network versions

### 9.2.1 802.11 LAN architecture

- wireless host communicates with base station (base station = access point (AP))
- **Basic Service Set (BSS)** (aka “cell”) in infrastructure mode contains:
  - wireless hosts
  - access point (AP): base station
  - ad hoc mode: hosts only

### 9.2.2 Channels, association

- 802.11b: 2.4GHz - 2.485GHz spectrum divided into 11 channels at different frequencies
  - AP admin chooses frequency for AP
  - interference possible: channel can be same as that chosen by neighboring AP
- host: must **associate** with an AP

- scans channels, listening for *beacon frames* containing AP’s name (SSID) and MAC address
- selects AP to associate with
- may perform authentication
- will typically run DHCP to get IP address in AP’s subnet

802.11 has no collision detection

### 9.2.3 IEEE 802.11 MAC Protocol: CSMA/CA

#### 802.11 sender

1. if sense channel idle for **DIFS** then transmit entire frame (no CD)
2. if sense channel busy then:
  - (a) start random backoff time
  - (b) timer counts down while channel idle
  - (c) transmit when timer expires
  - (d) if no ACK, increase random backoff interval, repeat 2

**802.11 receiver** If frame received OK → return ACK after **SIFS** (ACK needed due to hidden terminal problem)

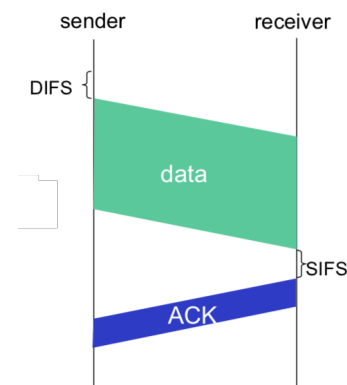


Figure 7: 802.11 CSMA/CA

#### Avoiding Collisions

- sender first transmits *small* request-to-send (RTS) packets to BS using CSMA (RTSs may still collide with each other (but they’re short))
- BS broadcasts clear-to-send (CTS) in response to RTS
- CTS heard by all nodes (sender transmits data frame, other stations defer transmissions)



## 9.2.4 802.11 frame: addressing

Table 11: 802.11 frame								
2	2	6	6	6	2	6	0-2312	4
frame control	duration	address 1	address 2	address 3	sequence control	address 4	payload	CRC

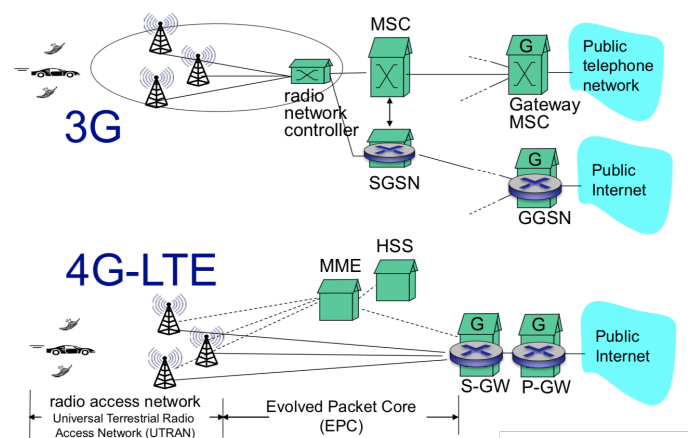
Table 12: Frame Control Break Down										
2	2	4	1	1	1	1	1	1	1	1
Protocol version	Type	Subtype	To AP	From AP	More frag	Retry	Power mgt	More data	WEP	Rsvd

**Address 1:** MAC address of wireless host or AP to receive this frame

**Address 2:** MAC address of wireless host or AP transmitting this frame

**Address 3:** MAC address of router interface to which AP is attached

**Address 4:** used only in ad hoc mode



## 9.3 802.11 Power Management

- node-to-AP: Sleeping till next frame (AP knows not to transmit frames to this node, node wakes up before next beacon frame)
- beacon frame: contains list of mobiles with AP-to-mobile frames waiting to be sent (node will stay awake if AP-to-mobile frames to be sent; otherwise sleep again until next beacon frame)

## 9.4 802.15: Personal Area Network

- less than 10m diameter
- replacement for cables (mouse, keyboard, headphones)
- ad hoc: no infrastructure
- master/slaves: (slaves request permission to send (to master), master grants requests)
- 802.15: evolved from Bluetooth specification (2.4 - 2.5 GHz radio band, up to 721 kbps)

Figure 8: 3G vs 4G LTE

## 10 Chapter 8

### 10.1 What is network security

**confidentiality:** only sender, intended receiver should “understand” message contents (sender encrypts message, receiver decrypts message)

**authentication:** sender, receiver want to conform identity of each other

**message integrity:** sender, receiver want to ensure message not altered (in transit, or afterwards) without detection

**access and availability:** services must be accessible and available to users

What can a “bad guy” do?

- **eavesdrop:** intercept messages
- actively **insert** messages into connection
- **impersonation:** can fake (spoof) source address in packet (or any field in packet)
- **hijacking:** “take over” ongoing connection by removing sender or receiver, inserting himself



in place

- **denial of service:** prevent service from being used by other (e.g. by overloading resources)

### 10.1.1 Breaking an encryption scheme

**cipher-text only attack:** Trudy has ciphertext she can analyze

**two approaches:** brute force: search through all keys. Statistical analysis

**known-plaintext attack:** Trudy has plaintext corresponding to ciphertext (e.g. in monoalphabetic cipher, Trudy determines pairings for a,l,i,c,e,b,o)

**chosen-plaintext attack:** Trudy can get ciphertext for chosen plaintext

## 10.2 Symmetric Key Cryptography

Bob and Alice share same (symmetric) key, e.g. key is knowing substitution pattern in mono alphabetic substitution cipher

### 10.2.1 Substitution Cipher

substituting one thing for another. Monoalphabetic cipher: substitute one letter for another.

Make it more sophisticated by adding cycling pattern. For  $n$  substitution ciphers:  $M_1, M_2, \dots, M_n$ . Cycling pattern:  $n = 4$   $M_1, M_3, M_4, M_2$ . Example: dog, d from  $M_1$ , o from  $M_3$ , g from  $M_4$ . Need to send  $n$  substitution ciphers and cyclic pattern

### 10.2.2 DES: Data Encryption Standard

- US encryption standard [NIST 1993]
- 56-bit symmetric key, 64-bit plaintext input
- block cipher with cipher block chaining
- 56-bit-key-encrypted phrase decrypted (brute force) in less than a day
- no known good analytic attack
- making DES more secure: **3DES**, encrypt 3 times with 3 different keys

### 10.2.3 AES: Advanced Encryption Standard

- symmetric-key NIST standard, replaced DES (Nov 2001)
- processes data in 128 bit blocks
- 128, 192, or 256 bit keys

- brute force decryption (try each key) taking 1 sec on DES, takes 149 trillion years for AES

## 10.3 Public Key Cryptography

- radically different approach (Diffie-Hellman76, RSA78)
- sender, receiver do not share secret key
- public encryption key known to all
- private decryption key known only to receiver

### Note 19: Why is RSA secure?

- suppose you know Bob's public key  $(n, e)$ . How hard is it to determine  $d$ ?
- essentially need to find factors of  $n$  without knowing the two factors  $p$  and  $q$  (factoring a big number is hard)

### RSA in practice

- exponentiation in RSA is computationally intensive
- DES is at least 100 times faster than RSA
- use public key crypto to establish secure connection, then establish second key – symmetric session key – for encrypting data

## 10.4 Authentication

Can use nonce (a once-in-a-lifetime value) to encrypt the password. This is still open to a man-in-the-middle attack, and is undetectable.

## 10.5 Digital Signatures

**Cryptographic technique analogous to handwritten signatures**

- sender digitally signs document, establishing he is document owner/creator
- verifiable, nonforgeable: recipient can prove to someone that sender, and no one else (including receiver), must have signed document
- encrypt the document with private key, the document can only be decrypted with public key

## 10.6 Hashing

- produces a fixed length message size
- many-to-1
- given  $x$  computationally infeasible to find  $m$  such that  $x = H(m)$

*Internet checksum is poor hash algorithm, easy to find message with same hash*

### Note 20: Signed Message Hash

Because Public Key Cryptography is computationally heavy for a large message, hash the message then sign the hash. Then the receiver can hash the message and apply the public key to see if they match

### Note 21: Toy SSL: simple secure channel

1. **handshake:** Alice and Bob use their certificates, private keys to authenticate each other and exchange shared secret
2. **key derivation:** Alice and Bob use shared secret to derive set of keys
3. **data transfer:** data to be transferred is broken up into series of records
4. **connection closure:** special messages to securely close connection

## 10.6.1 Hash Algorithms

- **MD5 hash function widely used (RFC 1321)**
  - computes 128-bit message digest in 4-step process
  - arbitrary 128-bit string  $x$ , appears difficult to construct message  $m$  whose MD5 hash is equal to  $x$
- **SHA-1 is also used**
  - US standard [NIST, FIPS PUB 180-1]

## 10.7 Certification Authorities

- **certification authority (CA):** binds public key to particular entity,  $E$
- $E$  (person, router) registers its public key with CA.
  - $E$  provides “proof of identity” to CA
  - CA creates certificate binding  $E$  to its public key
  - certificate containing  $E$ ’s public key digitally signed by CA – CA says “this is  $E$ ’s public key”

## 10.8 SSL: Secure Sockets Layer

- widely deployed security protocol (supported by almost all browser, web servers), (https, billions of \$/year over SSL)
- variation -TLS, RFC 2246
- provides:
  - confidentiality
  - integrity
  - authentication
- available to all TCP applications (secure socket interface)
- SSL provides application programming interface (API) to applications
- SSL is considered an application layer protocol on top of TCP, but used by programmers like a transport service

## 10.8.1 SSL cipher suite

- cipher suite (public-key algorithm, symmetric encryption algorithm, MAC algorithm)
- SSL supports several cipher suites
- negotiation: client, server agree on cipher suite (client offers choice, server picks one)

### Note 22: Common SSL Algorithms

#### common SSL symmetric ciphers

- DES – Data Encryption Standard: block
- 3DES – Triple strength: block
- RC2 – Rivest Cipher 2: block
- RC4 – Rivest Cipher 4: stream

#### SSL Public key encryption

- RSA

Table 13: SSL record format

1 byte	2 bytes	3 bytes			
content type	SSL version	length	data	MAC	

*data and MAC encrypted (symmetric algorithm)*

## 10.9 Network-layer Security

### 10.9.1 What is network-layer confidentiality

- sending entity encrypts datagram payload, payload could be: TCP or UDP segment, ICMP message, OSPF message...
- all data sent from one entity to other would be hidden: web pages, e-mail, P2P file transfers, TCP SYN packets
- **“blanket coverage”**

### 10.9.2 Virtual Private Networks (VPNs)

VPN: institution’s inter-office traffic is sent over public Internet instead

- encrypted before entering public Internet

- logically separate from other traffic

### 10.9.3 IPsec services

- data integrity
- origin authentication
- replay attack prevention
- confidentiality
- two protocols providing different service models:
  - Authentication Header (AH) protocol → provides source authentication and data integrity but not confidentiality
  - Encapsulation Security Protocol (ESP) → provides source authentication, data integrity, and confidentiality. More widely used than AH
- two modes:
  - host mode: host computers encrypt and decrypt
  - tunneling mode: edge routers IPsec-aware

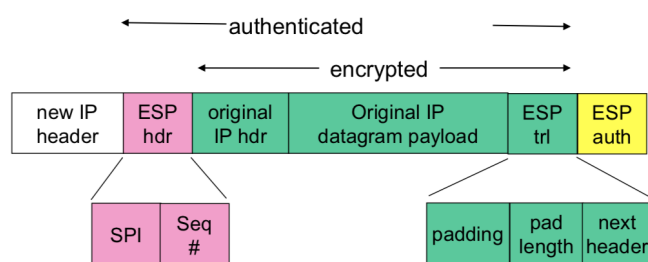


Figure 9: IPsec datagram (tunnel, ESP)

- IKE message exchange for algorithms, secret keys, SPI numbers

## 10.10 Firewalls

**prevent DoS attacks:** SYN flooding, attacker establishes many bogus TCP connections, no resources left for “real” connections

**prevent illegal modification/access of internal data:** e.g. attack replaces CIA’s homepage with something else

**allow only authorized access to inside network:** set of authenticated users/hosts

**three types of firewalls:** stateless packet filters, stateful packet filters, application gateways

### 10.10.1 Limitations of firewalls, gateways

- **IP spoofing:** router can’t know if data “really” comes from claimed source

- if multiple applications need special treatment, each has own application gateway
- client software must know how to contact gateway (e.g. must set IP address of proxy in Web browser)
- filters often use all or nothing policy for UDP
- **tradeoff:** degree of communication with outside world, level of security
- many highly protected sites still suffer from attacks

### 10.10.2 Intrusion detection systems

- packet filtering:
  - operates on TCP/IP headers only
  - no correlation check among sessions
- **IDS:** Intrusion detection system
  - deep packet inspection:** look at packet contents (e.g. check character strings in packet against database of known virus, attack strings)
  - examine correlation** among multiple packets (port scanning, network mapping, DoS attack)

## 11 Chapter 9

### 11.1 Multimedia

#### 11.1.1 Audio

- Analog audio signal sampled at constant rate (telephone: 8000 samples/sec, CD music: 44,100 samples/sec)
- quantized, i.e. rounded (8 bits for 256 values)
- Example rates (CD: 1.411 Mbps, MP3: 96,128,160 kbps, Internet Telephony: 5.3 kbps)

#### 11.1.2 Video

- MPEG 1 (CD-ROM) 1.5 Mbps
- MPEG 2 (DVD) 3-6 Mbps
- MPEG 4 (often used in Internet) <1 Mbps

#### 11.1.3 3 Application Types

- **Streaming, stored** audio, video
  - streaming:** can begin playback before downloading entire file
  - stored (at server):** can transmit faster than audio/video will be rendered (implies storing/buffering at client)

- **conversational** voice/video over IP (interactive nature of human-to-human conversation limits delay tolerance)
- **streaming live** audio, video
- Once playout begins, playback must match original timing (network delays are variable)
  - **Client-side buffer** to match playout requirements
- RTP [RFC 2326] multimedia payload types

## 11.2 Voice-over-IP (VoIP)

- **VoIP end-end-delay requirement:** needed to maintain “conversational” aspect (< 150msec: good, > 400msec bad)
- **session initialization:** how does callee advertise IP address, port number, encoding algorithms?
- **value-added services:** call forwarding, screening, recording
- **emergency services:** 911
- **network loss:** IP datagram lost due to network congestion (router buffer overflow)
- **delay loss:** IP datagram arrives too late for playout at receiver
- **loss tolerance:** depending on voice encoding, loss concealment, packet loss rates between 1% and 10% can be tolerated

### 11.2.1 Adaptive playout delay

- estimate network delay, adjust playout delay at beginning of each talk spurt
- silent periods compressed and elongated

$$d_i = (1 - \alpha)d_{i-1} + \alpha(r_i - t_i)$$

$d_i$ : delay estimate after  $i$ th packet

$\alpha$ : small constant e.g. 0.1

$r_i$ : time received

$t_i$ : time sent (timestamp)

### 11.2.2 Recovery from packet loss

#### simple FEC (Forward Error Correction)

- for every group of  $n$  chunks, create redundant chunk by exclusive OR-ing  $n$  original chunks
- send  $n + 1$  chunks, increasing bandwidth by factor  $\frac{1}{n}$
- can reconstruct original  $n$  chunks if at most one lost chunk from  $n + 1$  chunks, with playout delay

## 11.3 Real-Time Protocol (RTP)

- RTP specifies packet structure for packets carrying audio, video data
- RFC 3550
- RTP packet provides
  - payload type identification
  - packet sequence numbering
  - time stamping
- RTP runs in end systems
- RTP packets encapsulated in UDP segments
- interoperability: if two VoIP applications run RTP, they may be able to work together

### 11.3.1 RTP runs on top of UDP

RTP libraries provide transport-layer interface that extends UDP:

- port numbers, IP addresses
- payload type identification
- packet sequence numbering
- time-stamping

### 11.3.2 RTP and QoS

- RTP does **not** provide any mechanism to ensure timely data delivery or other QoS guarantees
- RTP encapsulation only seen at end systems (not by intermediate routers)
  - routers provide best-effort service, making no special effort to ensure that RTP packets arrive at destination in timely matter

## 11.4 Real-Time Control Protocol (RTCP)

- works in conjunction with RTP
- each participant in RTP session periodically sends RTCP control packets to all other participants
- each RTCP packet contains sender and/or receiver reports
  - report statistics useful to application: # packets sent, # packets lost, interarrival jitter
- feedback used to control performance
  - sender may modify its transmissions based on feedback

### 11.4.1 RTCP: packet types

**receiver report packets:** fraction of packets lost, last sequence number, average interarrival jitter

ter

**sender report packets:** SSRC of RTP stream, current time, number of packets sent, number of bytes sent

**source description packets:** email address of sender, sender's name, SSRC of associated RTP stream. Provide mapping between the SSRC and the user/host name

#### Note 23: Comparison with H.323

- Another signaling protocol for real-time, interactive multimedia
- Complete, vertically integrated suite of protocols for multimedia conferencing; signaling, registration, admission control, transport, codecs
- SIP is a single component. Works with RTP, but does not mandate it. Can be combined with other protocols, services
- SIP uses KISS principle

## 11.5 SIP: Session Initiation Protocol

### long-term vision:

- all telephone calls, video conference calls take place over Internet
- people identified by names or e-mail addresses, rather than by phone numbers
- can reach callee (if callee so desires), no matter where callee roams, no matter what IP device callee is currently using

### 11.5.1 SIP services

- SIP provides mechanisms for call setup:
  - for caller to let callee know she wants to establish a call
  - so caller, callee can agree on media type, encoding
  - to end call
- determine current IP address of callee: maps mnemonic identifier to current IP address
- call management:
  - add new media streams during call
  - change encoding during call
  - invite others
  - transfer, hold calls

*SIP default port 506*

#### **SIP registrar**

Function of SIP server (registrar). When Bob starts SIP client, client sends SIP REGISTER message to Bob's registrar server

#### **SIP proxy**

Function of SIP server (proxy). Alice sends invite message to her proxy, proxy responsible for routing SIP messages to callee possibly through multiple proxies. Bob sends response back through same set of SIP proxies, proxy returns Bob's SIP response message to Alice. SIP proxy analogous to local DNS server plus TCP setup

## 12 Packet Format

### 12.1 Application Layer

#### 12.1.1 Request Message

The diagram shows an HTTP request message with the following text and annotations:

```
GET /index.html HTTP/1.1\r\n
Host: www-net.cs.umass.edu\r\n
User-Agent: Firefox/3.6.10\r\n
Accept: text/html,application/xhtml+xml\r\n
Accept-Language: en-us,en;q=0.5\r\n
Accept-Encoding: gzip,deflate\r\n
Accept-Charset: ISO-8859-1,utf-8;q=0.7\r\n
Keep-Alive: 115\r\n
Connection: keep-alive\r\n
\r\n
```

Annotations:

- request line (GET, POST, HEAD commands)**: Points to the first line of the message.
- header lines**: Points to the lines between the request line and the blank line.
- carriage return, line feed at start of line indicates end of header lines**: Points to the `\r\n` sequence at the end of the header section.
- carriage return character**: Points to the `\r` character in the request line.
- line-feed character**: Points to the `\n` character in the request line.

Figure 10: HTTP Request Message

#### 12.1.2 Response Message

The diagram shows an HTTP response message with the following text and annotations:

```
HTTP/1.1 200 OK\r\n
Date: Sun, 26 Sep 2010 20:09:20 GMT\r\n
Server: Apache/2.0.52 (CentOS)\r\n
Last-Modified: Tue, 30 Oct 2007 17:00:02 GMT\r\n
ETag: "17dc6-a5c-bf716880"\r\n
Accept-Ranges: bytes\r\n
Content-Length: 2652\r\n
Keep-Alive: timeout=10, max=100\r\n
Connection: Keep-Alive\r\n
Content-Type: text/html; charset=ISO-8859-1\r\n
\r\n
data data data data data ...
```

Annotations:

- status line (protocol status code status phrase)**: Points to the first line of the message.
- header lines**: Points to the lines between the status line and the blank line.
- data, e.g., requested HTML file**: Points to the body of the message.

Figure 11: HTTP Response Message

### 12.2 Presentation

**ISO/OSI Reference:** Allow applications to interpret meaning of data, e.g. encryption, compression, machine-specific conventions



## 12.3 Session

**ISO/OSI Reference:** Synchronization, check-pointing, recovery of data exchange

## 12.4 Transport Layer

### 12.4.1 UDP

Table 14: UDP Breakdown

Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Source port																Destination port															
4	32	Length																Checksum															

**Source Port:** Sender's port number

**Destination Port:** Receiver's port number

**Length:** Length of UDP header and data in bytes. Min is 8, max is 65,535 bytes

**Checksum:** Optional for IPv4

### 12.4.2 TCP

Table 15: TCP Breakdown

Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Source port																Destination port															
4	32	Sequence number																															
8	64	Acknowledgment number (if ACK set)																															
12	96	Data offset				Reserved 0 0 0			N S	C W R	E C R	U R C	A P C	P S S	R S S	F Y I	Window Size																
16	128	Checksum																Urgent pointer (if URG set)															
20	160	Options (if <i>data offset</i> > 5. Padded at the end with "0" bytes if necessary.)																															
...	...	...																															

**Source Port:** Sending Port

**Destination Port:** Receiving Port

**Sequence Number:** If the SYN flag is set (1), then this is the initial sequence number. The sequence number of the actual first data byte and the acknowledged number in the corresponding ACK are then this sequence number plus 1

If the SYN flag is clear (0), then this is the accumulated sequence number of the first data byte of this segment for the current session

**Acknowledgment number:** If the ACK flag is set then the value of this field is the next sequence number that the sender of the ACK is expecting. This acknowledges receipt of all prior bytes (if any). The first ACK sent by each end acknowledges the other end's initial sequence number itself, but no data

**Data offset:** Size of TCP header in 32-bit words. Min 5, Max 15 words (max of 60 bytes → 40 byte options)

**Flags:** **NS:** ECN-nonce concealment

**CWR:** Congestion Window Reduction

**ECE:** ECN-Echo

**URG:** Urgent pointer field is significant

**ACK:** Acknowledgment field is significant

**PSH:** Push function  
**RST:** Reset the connection  
**SYN:** Synchronize sequence numbers  
**FIN:** Last packet from sender

**Window Size:** Number of bytes the receiver is currently willing to accept

**Checksum:** Error-checking of header, payload, and Pseudo-Header. Pseudo-Header includes; Source IP Address, Destination IP Address, protocol number (typically 0x0006), and length of TCP-Headers including payload

**Urgent Pointer:** Offset from sequence number indicating last urgent data byte (if URG set (1))

**Options:** *Pray to god this doesn't show up on the exam*

## 12.5 Network Layer

### 12.5.1 IPv4

Table 16: IPv4 Breakdown

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

**Version:** For IPv4, this is always 4

**Internet Header Length(IHL):** Length of the header times 4 (e.g. IHL = 5, header length is  $5 \times 4 = 20$  bytes. Minimum value is 5, max is 15)

**Differentiated Services Code Point (DSCP):** Originally ToS, used for DiffServ technologies requiring real-time streaming (e.g. VoIP)

**Explicit Congestion Notification (ECN):** Notifies each end of congestion without dropping packets

**Total Length:** Entire packet size, including header and data. Min is 20, max is 65,535 bytes

**Identification:** Helps identify groups of IP packets that have been fragmented

**Flags:** bit 0: Reserved, must be zero. bit 1: Don't Fragment (DF). bit 2: More Fragments (MF)

**Fragment Offset:** Measured in 8-byte blocks, provides the offset since the original unfragmented IP datagram

**Time To Live (TTL):** Used to make sure the packet doesn't stay in circulation on the Internet

**Protocol:** See Table 17. If port 53 and protocol is UDP, DNS request most likely

**Header Checksum:** Checksum for the header only, encapsulated protocol must handle incorrect data errors

**Source Address:** IPv4 sender address for the packet

**Destination Address:** IPv4 receiver address for the packet

**Options:** Options field not often used. IHL must include enough width to allocate for options. Options can end with an EOL (End of Options List, 0x00) if it doesn't match up with ending of IHL. Possible options are:

**Copied:** (1 bit) Set to 1 if the options need to be copied into all fragments of a fragmented packet

**Option Class:** (2 bits) A general options category. 0 is for "control" options, and 2 is for "debugging and measurement". 1 and 3 reserved

**Option Number:** (5 bits) Specifies an option



**Option Length:** (8 bits) Indicated the size of the entire option (including this field). This field may not exist for simple options

**Option Data:** (Variable Bits) Option-specific data. This field may not exist for simple options

Table 17: Internet Protocol Numbers from RFC 790

Decimal	Hex	Protocol
1		Internet Control Message Protocol (ICMP)
2		Internet Group Management Protocol (IGMP)
6		Transmission Control Protocol (TCP)
17		User Datagram Protocol (UDP)
41		IPv6 encapsulation (ENCAP)
89		Open Shortest Path First (OSPF)
132		Stream Control Transmission Protocol (SCTP)

## 12.6 Link Layer

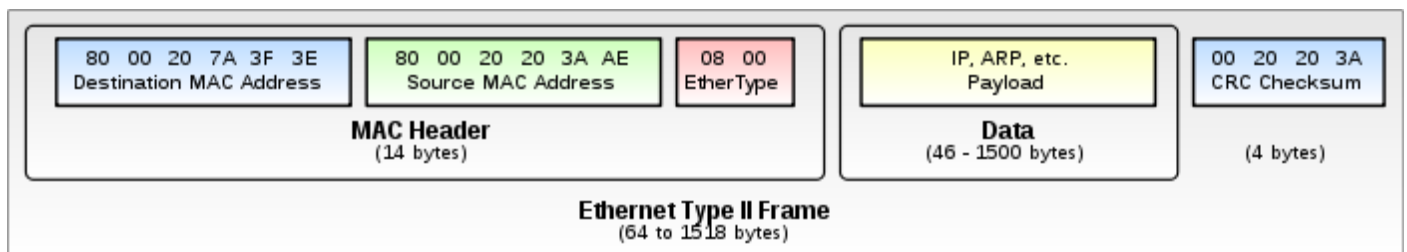


Figure 12: Ethernet Breakdown

**Destination:** (6 bytes) Destination MAC Address

**Source:** (6 bytes) Source MAC Address

**Type:** (2 bytes) The type of internet communication (IPv4: 0x0800)

## 12.7 Physical

## 13 Protocols

### 13.1 Application Layer

#### 13.1.1 HTTP: Hypertext Transfer Protocol

See Section 4.4

#### 13.1.2 SMTP: Simple Mail Transfer Protocol

See Section 4.7

#### 13.1.3 DNS: Domain Name Service

See Section 4.8

#### 13.1.4 DASH: Dynamic, Adaptive Streaming over HTTP

See Section 4.10.2

### 13.2 Transport Layer

#### 13.2.1 UDP: User Datagram Protocol

See Section 5.3

#### 13.2.2 TCP: Transmission Control Protocol

See Section 5.5

#### 13.2.3 RTP: Real-Time Protocol

See Section 11.3

#### 13.2.4 RTCP: Real-Time Control Protocol

See Section 11.4

#### 13.2.5 SIP: Session Initiation Protocol

See Section 11.5

### 13.3 Network Layer

#### 13.3.1 IP: Internet Protocol

See Section 6.3

#### 13.3.2 DHCP: Dynamic Host Configuration Protocol

See Section 6.3.5

#### 13.3.3 NAT: Network Address Translation

See Section 6.3.7 (this is a technique that just modifies this layer)

#### 13.3.4 IPv6

See Section 6.3.8

#### 13.3.5 ICMP: Internet Control Message Protocol

See Section 7.7

#### 13.3.6 OSPF: Open Shortest Path First

See Section 7.3.2  
Intra-AS Routing, Link-State

#### 13.3.7 BGP: Border Gateway Protocol

See Section 7.3.4  
Inter-AS Routing, Distance Vector (Path Vector Protocol)

#### 13.3.8 IS-IS: Intermediate System to Intermediate System

Intra-AS Routing, Link-State

#### 13.3.9 RIP: Routing Information Protocol

Intra-AS Routing, Distance Vector

#### 13.3.10 IGRP: Interior Gateway Routing Protocol

Intra-AS Routing, Distance Vector

### 13.4 Link Layer

#### 13.4.1 TDMA: Time Division Multiple Access

See Section 8.4.1

#### 13.4.2 FDMA: Frequency Division Multiple Access

See Section 8.4.2

#### 13.4.3 Slotted ALOHA

See Section 8.5.1

#### **13.4.4 Pure (unslotted) ALOHA**

See Section 8.5.2

#### **13.4.5 CSMA (Carrier Sense Multiple Access)**

See Section 8.5.3

#### **13.4.6 CSMA/CD: Collision Detection**

See Section 8.5.4

#### **13.4.7 Polling**

See Section 8.6.1

#### **13.4.8 Token Passing**

See Section 8.6.2

#### **13.4.9 DOCSIS: Data Over Cable Service Interface Spec**

See Section 8.7.1

#### **13.4.10 ARP: Address Resolution Protocol**

See Section 8.8.2

## 14 Acronyms

**IP:** Internet Protocol  
**TCP:** Transmission Control Protocol  
**UDP:** User Datagram Protocol  
**HTTP:** Hypertext Transfer Protocol  
**SMTP:** Simple Mail Transfer Protocol  
**RDP:** Remote Desktop Protocol  
**VOIP:** Voice over IP  
**RTT:**  
**POP:** Post Office Protocol  
**IMAP:** Internet Mail Access Protocol  
**DNS:** Domain Name System  
**SSN:**  
**TLD:** Top-level Domain  
**TTL:** Time To Live  
**RR:** Resource Records  
**DDoS:** Distributed Denial of Service  
**DoS:** Denial of Service  
**CBR:** Constant bit rate  
**VBR:** Variable bit rate  
**ABR:**  
**UBR:**  
**DASH:** Dynamic, Adaptive Streaming over HTTP  
**CDN:** Content Distribution Networks  
**RDT:** Reliable Data Transfer  
**MSS:**  
**ECN:** Explicit Congestion Notification  
**ECE:**  
**SDN:** Software-defined networking  
**TCAMs:** Ternary Content Addressable Memories  
**HOL:** Head-of-the-Line  
**FIFO:** First in first out  
**RR:** Round Robin  
**WFQ:** Weighted Fair Queuing  
**CIDR:** Classless InterDomain Routing  
**DHCP:** Dynamic Host Configuration Protocol  
**ICANN:** Internet Corporation for Assigned Names and Numbers  
**NAT:** Network Address Translation  
**QoS:** Quality of Service  
**OSPF:** Open Shortest Path First  
**ODL:**  
**ONOS:**  
**ICMP:** Internet Control Message Protocol  
**SNMP:**  
**CA:** Control Agent  
**DV:** Distance Vector  
**B-F:** Bellman-Ford  
**LS:** Link State  
**AS:** Autonomous Systems  
**IGP:** Interior Gateway Protocols  
**RIP:** Routing Information Protocol

**OSPF:** Open Shortest Path First  
**IGRP:** Interior Gateway Routing Protocol  
**ToS:** Type of Service  
**MOSPF:** Multicast OSPF  
**BGP:** Border Gateway Protocol  
**SDN:** Software Defined Networking  
**SNMP:**  
**ODL:** OpenDaylight  
**SAL:** Service Abstraction Layer  
**OVSDB:**  
**MIB:** Management Information Base  
**MAC:** Medium Access Control  
**NIC:** Network Interface Card  
**CRC:** Cyclic Redundancy Check  
**TDMA:** Time Division Multiple Access  
**FDMA:** Frequency Division Multiple Access  
**ALOHA:**  
**CSMA:** Carrier Sense Multiple Access  
**DOCSIS:** Data Over Cable Service Interface Spec  
**FDM:**  
**TDM:**  
**ARP:**  
**LAN:** Local Area Network  
**WAN:** Wide Area Network  
**ARP:** Address Resolution Protocol  
**VLAN:** Virtual Local Area Network  
**SNR:** Signal-to-noise ratio  
**CDMA:** Code Division Multiple Access  
**AP:** Access Point  
**BSS:** Basic Service Set  
**SSID:**  
**DIFS:**  
**SIFS:**  
**BS:** Base Station  
**RTS:** Request-To-Send  
**CTS:** Clear-To-Send  
**WEP:**  
**MSC:** Mobile Switching Center  
**BTS:** Base Transceiver Station  
**BSC:** Base Station Controller  
**MSC:** Mobile Switching Center  
**GPRS:** General Packet Radio Service  
**SGSN:** Serving GPRS Support Node  
**GGSN:** Gateway GPRS Support Node  
**WCDMA:**  
**HSPA:**  
**UTRAN:** Universal Terrestrial Radio Access Network  
**EPC:** Evolved Packet Core  
**UE:** User Element  
**MME:** Mobility Management Entity  
**HSS:** Home Subscriber Server  
**HLR:**

**VLR:**

**S-GW:** Serving Gateway

**P-GW:** Packet data network Gateway

**DES:** Data Encryption Standard

**AES:** Advanced Encryption Standard

**NIST:**

**FIPS PUB 180-1:**

**CA:** Certification Authority

**SSL:** Secure Sockets Layer

**TLS:** Transport Layer Security

**VPN:** Virtual Private Network

**AH:** Authentication Header

**ESP:** Encapsulation Security Protocol

**IKE:**

**SPI:**

**IDS:** Intrusion Detection System

**FEC:** Forward Error Correction

**RTP:** Real-Time Protocol

**RTCP:** Real-Time Control Protocol

**SSRC:**

**SIP:** Session Initiation Protocol

**SDP:** Session Description Protocol

**SCTP:** Stream Control Transmission Protocol