3N03 Final Cheatsheet

Network Fundamentals

Network: Collection of devices interconnected by a single technology (internet)

Network Uses

Business Applications: Resource/info sharing, communication, client-server model
Home Applications: Peer-to-peer model
Mobile Users: Wireless connectivity

E-Commerce Types

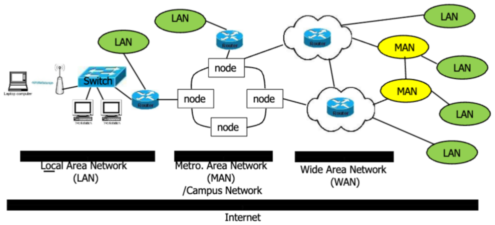
B2C: Business to Consumer
B2B: Business to Business
G2C: Government to Consumer
C2C: Consumer to Consumer
P2P: Peer to Peer

Social Issues

Network Neutrality, Content ownership, Anonymity & Censorship, Privacy, Info Theft

Network Scales

PAN: Personal (Bluetooth)
LAN: Local (Office)
MAN: Metropolitan (City)
WAN: Wide (Country/ISP (Internet Service Provider), a company). Serve as modern internet backbone.
Internet: Network of networks (Planet)



Connection Types

Internetwork: Network of smaller networks
Internet: Set of all connected networks
Gateway: Device transferring data between layers
Low-Level Gateways:

- Operate at the lower layers of the network protocol
- More limited in functionality
- Cannot effectively connect different types of networks
- *Focus on basic data transmission*

High-Level Gateways:

- Operate at higher layers of the protocol stack (application layer)
- Very specific in their function
- Limited to particular applications or services
- Example: An email gateway that only handles email traffic
- *While powerful for specific uses, they're too specialized for general network connectivity*

Best Mid-level Gateways

- "just right"
- Represented by routers, which operate at the network layer
- Provide the optimal balance between functionality and flexibility
- Can effectively route packets between different networks
- *Handle most common networking needs*
- Can connect different types of networks while maintaining good performance

Router: Gateway for network layer information

Transmission Technology

- broadcast links - communication channel shared by all machines in network
- point-to-point - direct connection between two machines
- packet - small unit of data sent over a network.

Layered Network Models

Each layer implements a service. Layering provides encapsulation, where each layer adds its own header to data.

Layered Network Design Issues

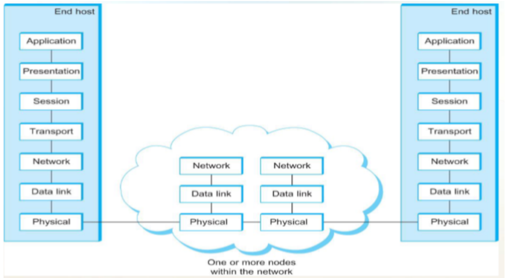
- Reliability/failure handling
- Network growth capability
- Resource allocation
- Security against threats

Layer services

Vertical: services for uni directional communication provided by bottom layer to top layer, like a gateway
Horizontal: protocols for communication between same layers
connection oriented: set up for ongoing use, torn down afterwords **connectionless:** separately and temporary handled messages

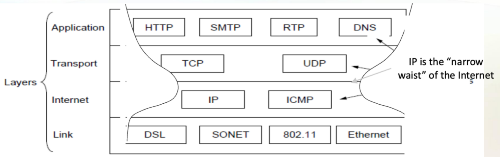
OSI Model

Open Systems Interconnection. Makes essential concepts explicit (services, interfaces, protocols).
7. Application: Network apps, end-user access, protocols (FTP, SMTP, HTTP)
6. Presentation: Data interpretation/formatting
5. Session: provides locality for different transport streams to not confuse individual streams. Estabashing a method of communicatino.
4. Transport: Process-to-process data transfer, segmentation, reliability (TCP, UDP)
3. Network: packet routing between networks (IP)
2. Data Link: raw bit collection, physical addressing, unit of data in frames
1. Physical: Raw bit transmission (cables, signals)



TCP/IP Model

More specific to the internet, heavily relies on protocols, whereas OSI model is more generalized.
4. Application: TELNET, FTP, SMTP, DNS, HTTP, RTP
3. Transport: TCP (reliable, connection-oriented), UDP (unreliable, connectionless)
2. Internet: IP packet delivery, multi-network connection support.
1. Link: implemented by combinatino of hardware (ethernet, fiberoptics, etc).



Data Transmission

Packet Transmission Delay: $\frac{L}{R}$ (Length in bits/Transmission rate in bits/s)
Store & Forward: Full packet received before forwarding, end-to-end delay $= 2\frac{L}{R}$

Network Core Functions

Routing: Determining packet paths
Forwarding: Moving packets between router interfaces

Application Layer

Architectures

Client-Server:

- Clients communicate with server
- Server: permanent IP, always-on
- Clients: dynamic IP, intermittent connectivity

Peer-to-Peer (P2P):

- Minimal server reliance
- Direct end-system communication
- Self-scaling with new peers
- Challenges: security, performance, management

Application Requirements

Data Transfer: Reliability needs
Timing: Delay sensitivity
Throughput: Bandwidth requirements
Security: Encryption, authentication

Transport Protocols

A protocol is a set of rules governing the format and meaning of the packets or messages that are exchanged. Protocols implement the services.

TCP Service:

- Reliable transport
- Flow control
- Congestion control
- Connection-oriented
- No timing/throughput guarantees
- No built-in security

UDP Service:

- Unreliable data transfer
- No guarantees
- Low overhead

Securing TCP: SSL (Secure Socket Layer) at application layer

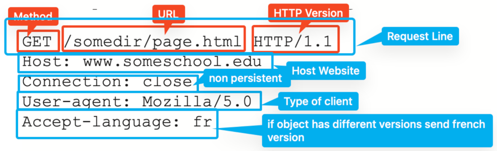
HTTP Protocol

Properties:

- Client/server model
- TCP on port 80
- Stateless
- Non-Persistent: One object per connection
- Persistent: Multiple objects per connection

Request Message:

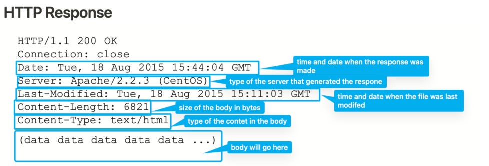
- ASCII format
- Request line, header lines, empty body



```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
Connection: close
User-agent: Mozilla/5.0
Accept-language: fr
```

Response Message:

- ASCII format
- Status line, header lines, data



HTTP Response

```
HTTP/1.1 200 OK
Connection: close
Date: Tue, 18 Aug 2015 15:44:04 GMT
Server: Apache/2.2.3 (CentOS)
Last-Modified: Tue, 18 Aug 2015 15:11:03 GMT
Content-Length: 6821
Content-Type: text/html

(data data data data data ...)
```

time and date when the response was made
type of the server that generated the respone
time and date when the file was last modifed
size of the body in bytes
type of the contet in the body
body will go here

Status Codes:

- 1xx: Informational
- 2xx: Success (200 OK)
- 3xx: Redirection (301 Moved)
- 4xx: Client Error (404 Not Found)
- 5xx: Server Error

Services

- GET: Retrieve data
- HEAD: read a webpage's header
- POST: Create new data
- PUT: update existing data
- DELETE: remove data
- TRACE: echo incoming request
- CONNECT: connect through a proxy
- OPTIONS: query options for a page

Cookies

- Stateful client/server interactions
- saves user data and activity in servers
- sent via clients/browsers
- Cookie header line in http response messag
- Cookie header line in http request message
- cookie file stored locally, managed by user client
- backend database at the website
- Uses
  - authorization
  - shopping carts
  - recommendations
  - user session state

Web Cache/Proxy

Browser connects to proxy not web server; reduces latency, proxy acts as client+server

MIME type

Multipurpose Internet Mail Extension. Encoding rules.

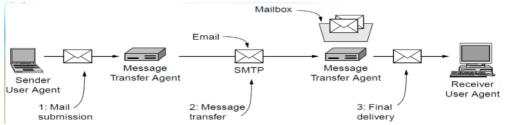
MIME Header Fields

Header	Meaning
MIME-Version	Identifies the MIME version
Content-Description	Human-readable string telling what is in the message
Content-Id	Unique identifier
Content-Transfer-Encoding	How the body is wrapped for transmission
Content-Type	Type and format of the content

MIME Content Types

Type	Example subtypes	Description
text	plain, html, xml, css	Text in various formats
image	gif, jpeg, tiff	Pictures
audio	basic, mpeg, mp4	Sounds
video	mpeg, mp4, quicktime	Movies
model	vrml	3D model
application	octet-stream, pdf, javascript, zip	Data produced by applications
message	http, rfc822	Encapsulated message
multipart	mixed, alternative, parallel, digest	Combination of multiple types

Email Architecture



Components:

- User Agents: For reading and sending email. Mail clients (Outlook, Apple Mail)
- Mail Servers: Mailbox and message queue

SMTP (Simple Mail Transfer Protocol):

- Client-server between mail servers
- Persistent TCP (port 25)
- Phases: Handshake, Transfer, Closure
- "Push" protocol (sending)

Mail Access Protocols:

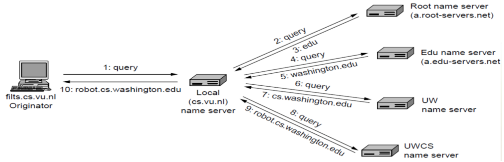
- POP3: Download from server (port 110)
  - Authorization, Transaction, Update
  - Download-and-Delete or Download-and-Keep
  - Stateless across sessions
  - No remote folders
- IMAP: Internet message Access Protocol is used for final delivery. Listens to port 143. More secure and more features than POP3.

DNS (Domain Name System)

Hierarchical domain based naming scheme with a database to implement it. Maps hostname to IP addresses, called **resolution**. **Name server** is in charge of a select group of domains. Runs UDP and uses port 53.

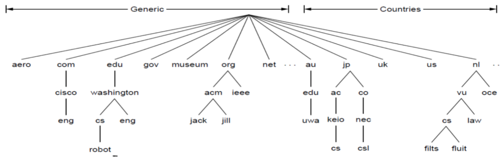
How it works

1. application program has hostname, passes it to resolver
2. resolver passes *hostname* to DNS server
3. DNS server returns IP address, sent as UDP packets



Hierarchy:

- ICANN (Internet Corporation for Assigned Names and Numbers)
- 250 top level domains, generics and one per country
- top level domains divided into subdomains



Vulnerabilities:

- DDoS attacks
- Redirect attacks (MITM, poisoning)
- Using DNS for DDoS amplification

Transport Layer

Functions:

- Communication between applications where from sender process the reciever process is on the same device.
- Implemented in end systems (not routers)
- Segments application messages and pass to network layer
- Routers read network layer datagrams not transport layer segments

Multiplexing/Demultiplexing

Process:

- ensures communication between processes/host on sender&reciever in transport layer/network layer
- On client side ts layer assigns src/dest ports to segments when multiplexing
- On reciever side, ts layer uses dest port to demultiplex segments into correct sockets

Protocols:

- TCP sockets: (source IP, source port, dest IP, dest port) tuple so based on source different sockets are made
- UDP sockets: (dest IP, dest port) tuple so the socket doesn't differentiate between different clients and sends all data to the same process

Reliable Data Transfer

Error types:

- Corruption (packet received incorrectly)
- Loss (packet never arrives)

UDP Characteristics

Features:

- Connectionless
- No handshaking
- Each segment handled independently
- No congestion control
- Has checksum for error detection

UDP Checksum Calculation

Process:

- Covers UDP header + data + pseudo-header
- Sender: Sum all 16-bit words, wrap when overflow, negate the result
- Receiver: Sum all 16-bit words including checksum, result should be all 1's

Example:

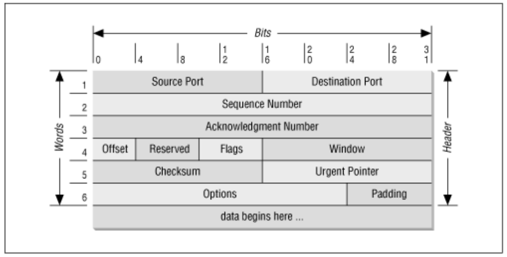
- Data: 0x1234, 0x5678, 0xABCD
- Sum: 0x1234 + 0x5678 = 0x68AC
- Sum: 0x68AC + 0xABCD = 0x11479 (overflow)
- Wrap around: 0x1479 + 0x0001 = 0x147A
- 1's complement: 0xFFFF - 0x147A = 0xEB85
- Checksum field: 0xEB85
- Receiver adds: 0x1234 + 0x5678 + 0xABCD + 0xEB85 = 0x1FFFE
- Wrap around: 0x1FFFE + 0x0001 = 0xFFFF (all 1's)

TCP Characteristics

Features:

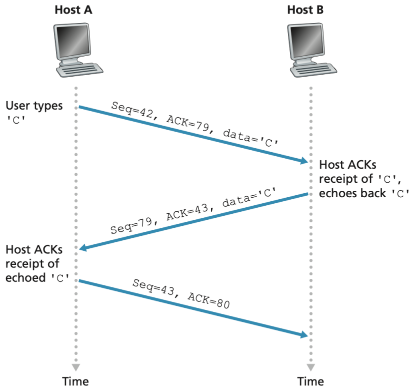
- Full-duplex service
- Point-to-point (single sender, single receiver)
- Connection-oriented with handshaking
- Reliable, ordered byte stream
- Flow control
- Congestion control

Segment Structure:



- Data field (limited by MSS)
- 32-bit sequence number
- 32-bit acknowledgment number
- 16-bit receive window
- 4-bit header length
- 6-bit flag field
- Options field

TCP interaction example



3-Way Handshake

SYN-ACK Process:

- **Step 1 (SYN):** Client sends SYN packet with initial sequence number x
  - SYN flag = 1
  - Sequence number = x (random)
- **Step 2 (SYN-ACK):** Server responds with SYN-ACK packet
  - SYN flag = 1
  - ACK flag = 1
  - Sequence number = y (random)
  - Acknowledgment number = x+1
- **Step 3 (ACK):** Client completes handshake with ACK
  - ACK flag = 1
  - Sequence number = x+1
  - Acknowledgment number = y+1

Socket Programming

Socket: Interface between application and transport protocol

UDP Socket Programming:

- No connection required
- Client attaches destination IP/port

TCP Socket Programming:

- Connection required
- Server needs welcome socket
- Creates new socket per client

```
int sock = socket(AF_INET, SOCK_STREAM, 0);

// Setup server address struct
struct sockaddr_in serv = {0};
serv.sin_family = AF_INET;
serv.sin_port = htons(PORT); // Convert port to network byte orde
inet_pton(AF_INET, "127.0.0.1", &serv.sin_addr); // IP string to

// Connect to server
connect(sock, (struct sockaddr*)&serv, sizeof(serv));

// Send message to server
char *msg = "Hello Server";
send(sock, msg, strlen(msg), 0);

// Receive server reply
char buffer[1024] = {0};
recv(sock, buffer, sizeof(buffer), 0);
printf("Reply: %s\n", buffer);

// Close socket
close(sock);
return 0;
}
```

}

Socket Programming

```
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>

#define PORT 8080
#define BACKLOG 5  // Max queued connections

int main() {
    // Create TCP socket (IPv4, Stream)
    int server_fd = socket(AF_INET, SOCK_STREAM, 0);

    // Setup server address struct
    struct sockaddr_in addr = {0};
    addr.sin_family = AF_INET;
    addr.sin_addr.s_addr = INADDR_ANY;     // Accept from any IP
    addr.sin_port = htons(PORT);           // Host to network byte or

    // Bind socket to IP and port
    bind(server_fd, (struct sockaddr*)&addr, sizeof(addr));

    // Start listening for client connections
    listen(server_fd, BACKLOG);

    // Accept first client (blocking call)
    int client_fd = accept(server_fd, NULL, NULL);

    // Receive data from client
    char buffer[1024] = {0};
    recv(client_fd, buffer, sizeof(buffer), 0);
    printf("Received: %s\n", buffer);

    // Send response to client
    char *msg = "Server Ack";
    send(client_fd, msg, strlen(msg), 0);

    // Close sockets
    close(client_fd); close(server_fd);
    return 0;
}
```

Socket Programming

```
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>

#define PORT 8080
#define BACKLOG 5  // Max queued connections

int main() {
    // Create TCP socket (IPv4, Stream)
    int server_fd = socket(AF_INET, SOCK_STREAM, 0);

    // Setup server address struct
    struct sockaddr_in addr = {0};
    addr.sin_family = AF_INET;
    addr.sin_addr.s_addr = INADDR_ANY;     // Accept from any IP
    addr.sin_port = htons(PORT);           // Host to network byte or

    // Bind socket to IP and port
    bind(server_fd, (struct sockaddr*)&addr, sizeof(addr));

    // Start listening for client connections
    listen(server_fd, BACKLOG);

    // Accept first client (blocking call)
    int client_fd = accept(server_fd, NULL, NULL);

    // Receive data from client
    char buffer[1024] = {0};
    recv(client_fd, buffer, sizeof(buffer), 0);
    printf("Received: %s\n", buffer);

    // Send response to client
    char *msg = "Server Ack";
    send(client_fd, msg, strlen(msg), 0);

    // Close sockets
    close(client_fd); close(server_fd);
    return 0;
}
```

Network Layer

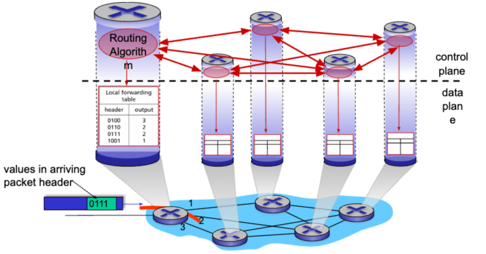
In charge of datagram routing between networks.

Routing

Network-wide path determination (seconds). Determine a route to move datagrams from source to destinatino. **Control plane:** network wide logic. the plane that plans over all route an ip datagram takes.

Forwarding

Move packets from router input link to router output link (microseconds). When looking for forwarding table entry for given destination address, use longest prefix that matches destination address. **Data plane:** local per router, plane that decides data transmission.



Service Models

Potential Properties:

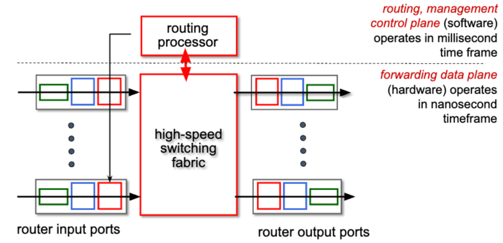
- Guaranteed delivery - guarantees sent packet is delivered
- Bounded delay - gurantees delivery within a specified delay bound
- In-order delivery - guarantees ordering of sent packets is consistent with ordering of recieved packets
- Minimum bandwidth - Guarantees delivery if packets are sent below a specified bit rate.
- Security - encryption at source, decruption at destination.
- best effort service model. **Disadvantages** - no guarantees for successful ip diagram delivery, delivery timing or order, or available bandwidth. **Advantages** - simple for wide accessibility and implementation. Good enough bandwidth. Supplemented with application layer services like datacenters and CDNs to allow services everywhere.

Router

Allows multiple devices to communicate with each other on a network. Multiple devices can share one ip address **Input Ports:** Link-layer functions (hardware)
Switching Fabric: Connects ports (hardware)
Output Ports: Transmits packets (hardware)

Routing Processor: Control plane (software)

}


routing processor
high-speed switching fabric
router input ports
router output ports
routing, management control plane (software)
operates in millisecond time frame
forwarding data plane (hardware) operates in nanosecond timeframe

IP Datagram

Wraps around transport layer segments. 1-1 mapping.
Characteristics:

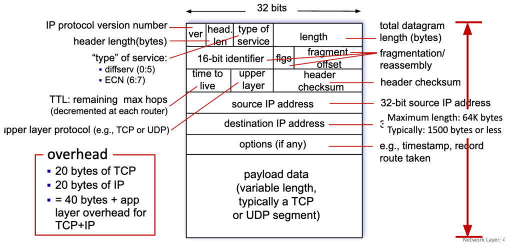
- IPv4: 32-bit identifier
- IPv6: 128-bit identifier
- Assigned by ICANN
- Hierarchical: network ID + host ID

Interface: Connection between host/router and link
Network ID: IP with host ID all zeros
Prefix: Lowest IP in block + size (bits in network portion)

IP Support Protocols

ARP: Finds MAC for local IP
DHCP: Dynamic IP assignment

IPv4

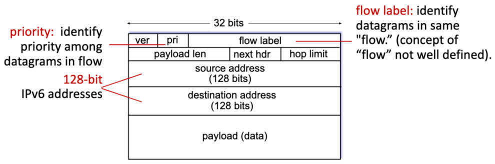

32 bits
IP protocol version number
header length(bytes)
"type" of service:
diffserv (0:5)
ECN (6:7)
TTL: remaining max hops (decremented at each router)
upper layer protocol (e.g., TCP or UDP)
ver
head. len
type of service
length
16-bit identifier
flgs
fragment offset
time to live
upper layer
header checksum
source IP address
destination IP address
options (if any)
payload data (variable length, typically a TCP or UDP segment)
total datagram length (bytes)
fragmentation/ reassembly
header checksum
32-bit source IP address
Maximum length: 64K bytes
Typically: 1500 bytes or less
e.g., timestamp, record route taken
Network Layer: 4

overhead

- 20 bytes of TCP
- 20 bytes of IP
- = 40 bytes + app layer overhead for TCP+IP

IPv6

Format: 3fff:0000:0000:0000:0123:4567:89AB:CDEF
Shortened: 3fff::123:4567:89AB:CDEF


32 bits
ver
pri
flow label
payload len
next hdr
hop limit
source address (128 bits)
destination address (128 bits)
payload (data)
priority: identify priority among datagrams in flow
128-bit IPv6 addresses
flow label: identify datagrams in same "flow." (concept of "flow" not well defined).

What's missing (compared with IPv4):

- no checksum (to speed processing at routers)
- no fragmentation/reassembly
- no options (available as upper-layer, next-header protocol at router)

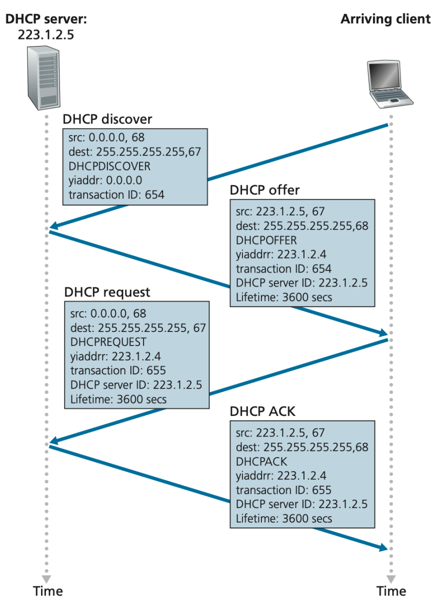
Address Types:

- Unicast: Single interface
- Anycast: Set of interfaces (closest)
- Multicast: Group of interfaces (all)

DHCP (Dynamic Host Configuration Protocol)

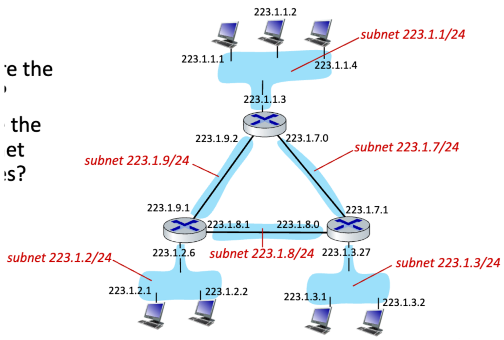
Dynamically get IP address upon joining network. Can renew adresses, reuse addresses, and have mobile support. A **DHCP request** is encapsulated in UDP (transport layer), then IP datagram (network layer), then ethernet (link layer).

DHCP Handshake



Subnet

- connection with direct device interface communication. (no intervening router)
- the blue shit in the diagram
- high order bits in ip addresses - common in the same subnet
- low order bits - unique



ARP - address resolution protocol Determines the mac address from the IP address.

ARP Protocol

Translates IP addresses to MAC addresses. Uses **ARP table** to store this mapping.

Algorithm

- Sender A broadcasts B's ip address to every host
- Hosts compare, B identifies it's theirs and sends its mac address towards A

Link Layer

Functions

- Encapsulates network datagrams in frames
- Error detection/correction
- Link access control
- Reliable delivery (optional)

Implementation: Hardware (NIC) + software

MAC Addresses

Media Access Control address. Used locally to get a frame to travel across a subnet.

- 48-bit (6 bytes, 12 HEX digits)
- IEEE managed
- Typically permanent (can be spoofed)

Ethernet

Topologies:

- Bus: Shared collision domain (old)
- Switched: Star topology with switch (current)

Frame Structure:

- Addresses: 6B source, 6B destination
- Type field
- CRC error checking
- Preamble (7B synchronization)

Properties: Connectionless, unreliable

Switch

- Stores/forwards frames
- Transparent to hosts
- Self-learning (no configuration)
- Maintains switch table (MAC to interface)

Physical Layer

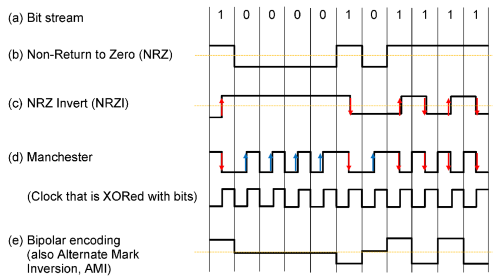
Signal Modulation

Digital Modulation: Converting bits to signals

Transmission Types:

- Baseband: Signal occupies frequencies from zero up to a maximum (wires)
- Passband: Schemes that regular amplitude, phase, or frequency of carrier signal to convey bits. The signal occupies a band of frequencies around the frequency of the carrier signal. (wireless/optical)

Encoding Methods



NRZ: Use a positive voltage to represent 1, negative for 0. Can use more levels of voltages, then the symbol carries more bits. Symbol rate = baud rate.

Manchester: Mixes clock signal with data signal by XORing them together. When the clock is XORed with 0 level, it makes a low-to-high transition (logical 0). When XORed with the 1 level, it is inverted and makes a high-to-low transition (logical 1).

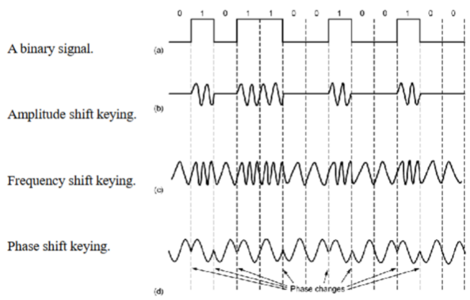
NRZI: Same as NRZ but code one as a transition and zero as no transition (or other way around).

4B/5B: Introduced to limit the number of consecutive 0s or 1s. Every 4 bits mapped to a 5-bit pattern with a fixed translation table. **4B/5B Line Encoding**

Data (4B)	Codeword (5B)	Data (4B)	Codeword (5B)
0000	11110	1000	10010
0001	01001	1001	10011
0010	10100	1010	10110
0011	10101	1011	10111
0100	01010	1100	11010
0101	01011	1101	11011
0110	01110	1110	11100
0111	01111	1111	11101

Passband Modulation

ASK: Amplitude shift keying (different amplitudes)
FSK: Frequency shift keying (different frequencies)
PSK: Phase shift keying (different phases)



Multiplexing

TDM: Time division multiplexing (users take turns)
FTTH: Deployment of fiber optic cables to provide high data rates to customers. One wavelength can be shared among many houses, up to 100Mbps. **FDM:** Different channels transmitted in different frequency bands. **Cable Internet:**Internet over cable reuses the cable television plant. Data sent on a shared cable tree from head-end, not on a dedicated line per subscriber.

Transmission Media

Guided Media:

- Twisted Pair:
  - Cat 5: 100Mbps (2 pairs)
  - Cat 5e: 1Gbps (4 pairs)
  - Cat 6: 10Gbps (up to 100m)
  - Cat 7: Shielded twisted pair
- Coaxial Cable: Better shielding, high bandwidth
- Power Lines: Convenient but noisy
- Fiber Optic: Light pulses, low error, high data rate
  - Single-mode: Narrow, laser, long distance
  - Multi-mode: Wider, LED, shorter distance

Transmission Modes:

- Full-Duplex Link: Transmission in both directions at the same time
- Half-Duplex Link: Both direction transmission, but not simultaneously
- Simplex Link: Only one fixed direction at all times, not common

Unguided Media:

- Terrestrial wireless
- Satellite
- Laser through air

Network Topologies

Bus: Single line, simple, one sender at a time
Star: Central switch, more cabling, higher reliability, single point of failure, mutliple devies can communicate simultaneously
Ring: Closed loop, token passing (one device at a time), difficult to expand, one computer down whole netowrk down

Network Hardware

NIC: Network adapter with MAC address
Hub: All nodes receive transmissions, slow, insecure
Switch: Only intended recipients receive data, fast, secure, plug and play
Router: Connects LANs via IP addresses, can connect across ineternet, needs configuration
Gateway: Connects dissimilar networks, connect coax to twisted pair.

Wave Properties

Frequency (f): Oscillations per second (Hz)
Period (T): Time between maxima (sec), T = 1/f
Wavelength (λ): Distance between maxima (m)
Relationship: $\lambda = c/f, c \approx 3 \times 10m/s$

Wireless Networks

Types

Wireless LANs: 100ft range, WiFi (54/300/1000 Mbps)
Wide-area Wireless: Cellular, 10's km, 1-100 Mbps

Characteristics

Advantages: Easy deployment, mobility support, broadcast capability
Challenges: Interference, variable signal strength/data rates

Network Security

Security Properties

Confidentiality: Only sender/receiver understand content
Message Integrity: Content unaltered in transit
Authentication: Verify sender/receiver identity
Operational Security: prevent malcious attacks from public network onto private network through firewall

Security Concepts

Firewall: Controls access between networks
Eavesdropping: Intercepting messages
Encryption: Disguising data from intruders

Key Types:

- Private/Symmetric: Same key for encrypt/decrypt
- Public/Asymmetric: Separate public/private keys

Cryptographic Hash

Fixed-size output that's computationally infeasible to reverse or find collisions

Message Authentication:

- Calculate H(m+s) where s is shared secret
- MAC = H(m+s)
- Send (m, MAC)
- Recipient verifies MAC

Digital Signatures

proof of ownership of some asset they should be verifiable and the signature should not be forgable.

Security Layers

Network layer security provides blanket coverage but not user-level security

Network Security Fundamentals

Core Terminology

- **Resource:** Something valuable to the organization that must be protected.
- **Vulnerability:** A weakness that a threat can exploit to gain unauthorized access to a resource.
- **Threat:** A potential danger or circumstance that could harm a resource.
- **Attack:** The act of exploiting a vulnerability to compromise or steal a resource.
- **Risk:** The likelihood that a resource is lost, modified, or removed (Risk = Resource + Threat + Vulnerability).
- **Counter-measure:** A safeguard that mitigates a threat or reduces risk.

Threat-Actor Taxonomy

- **White-hat:** Ethical testing, permission-based security audits
- **Black-hat:** Malicious financial or political gain
- **Gray-hat:** Mix of ethical and malicious activity
- **Blue-hat:** External penetration tester prior to release
- **Script kiddie:** Uses pre-written exploits with minimal skill
- **Hacktivist:** Social or political agenda
- **Phreaker:** Telephony exploits for free calls or network access
- **Carder:** Steals and trades credit-card data

Security Domains

- **Physical Security**: Cameras, locks, controlled server-room access.
- **Logical / Technical Security**: Password policy, antivirus, firewalls, VPN.
- **Administrative Security**: Training, phishing simulations, data-leak prevention.

Threat Landscape

Network Threats

Malware Types:

- Virus – self-replicating; needs user activation
- Worm – self-replicating; auto-spreads without user action
- Spyware – covertly monitors users
- Adware – injects unwanted advertisements
- Scareware – fake security warnings to provoke action
- Trojan – legitimate-looking program with hidden payload
- Ransomware – encrypts data until ransom is paid

Attack Types

Reconnaissance (Passive):

- Ping Sweep – identify live hosts
- Port Scanning – discover open services
- Packet Sniffing – capture and inspect traffic

Access Attacks:

- Phishing – deceptive e-mails / sites for credentials
- Pharming – DNS / hosts-file redirection
- MITM – intercept traffic
  - Spoofing – falsify source identity
  - Hijacking – take over authenticated session

Denial-of-Service (DoS):

- Saturation Flood – overwhelm with requests
- Vulnerability Exploitation – crash service via bug

Distributed DoS (DDoS) Examples:

- SYN Flood – half-open TCP handshakes
- ICMP Flood – excessive echo/response traffic

Security Best Practices

- Segmentation / security zones
- Defense-in-depth (layered controls)
- Least-privilege access
- Adequate protection at every OSI layer
- Information-access restriction
- Separation of duties & job rotation

Security Measures by Goal

- **Preventive** – firewalls, locks, policies
- **Detective** – logs, IDS/IPS, CCTV
- **Corrective** – patching, configuration fixes
- **Recovery** – backups, system restore
- **Deterrent** – legal notices, sanctions

End–to–End Packet Journeys

DHCP Address Assignment (Bootstrapping)

1. **Link-up, no IP yet** : When the host joins a wired or Wi-Fi LAN it has no IP address, so it must obtain one via DHCP.
2. **DHCP DISCOVER broadcast** : The client crafts a DHCP message and encapsulates it as
   - UDP src=68, dst=67
   - IPv4 src=0.0.0.0, dst=255.255.255.255
   - Ethernet src= client-MAC, dst=FF:FF:FF:FF:FF:FF

   The frame is flooded by any switches until a DHCP server hears it.
3. **Server processing** : The server demultiplexes the frame, extracts the DHCP request, and allocates network parameters (IP, subnet mask, default gateway, DNS server, lease time).
4. **DHCP OFFER / ACK unicast** : The server replies (UDP 67 → 68) with the chosen configuration. Now the reply can be unicast because the client's MAC address is known; the IP header still uses the offered yiaddr field, but Ethernet dst=client-MAC.
5. **Client configuration** : The host installs the assigned IP address and other options; L3/L4 are now ready for normal traffic.

DNS Name Resolution

1. **Need for a destination IP** : To reach `google.com` the host must map the domain to an IPv4/IPv6 address.
2. **DNS query construction** : A DNS query is built and sent to the resolver address learned from DHCP:
   - UDP src=random_port, dst=53
   - IP src=client-IP, dst=DNS-server-IP
3. **ARP first hop** : If the router's MAC is unknown, the host broadcasts an ARP REQUEST; after the ARP REPLY, the frame can be forwarded to the default gateway.
4. **Resolver/recursive lookup** : The ISP resolver consults its cache or walks the DNS hierarchy (root → TLD → authoritative) and formulates a DNS RESPONSE containing the A/AAAA record(s).
5. **Delivery and caching** : The UDP response traverses the reverse path to the host, which caches the mapping and can now open connections to the server IP.

Fetching a Web Page (HTTP over TCP)

1. **TCP three-way handshake** : The client opens a socket to the web-server IP (default port 80 or 443). SYN → SYN+ACK → ACK completes connection establishment.
2. **HTTP request/response** : The browser sends an `HTTP GET /` (or `HTTPS` inside TLS). The server replies with the HTML object (and subsequent resources).
3. **Rendering** : The application layer (browser) parses the HTML, issues additional object requests, and renders the page.
4. **End-to-end path** : Every packet follows the full route: host → access switch → edge router → ISP core → Google edge → Google data-center, and back, traversing the protocol stack at each hop.

Protocol Summary by Layer

Application: DNS (53), HTTP (80), HTTPS (443), SMTP (25), POP3 (110), IMAP (143), FTP (20/21), TELNET (23), SSH (22), DHCP (67/68), RTP, VoIP, SSL/TLS, MPEG-4, H.264, HTML5, CSS

Transport: TCP, UDP

Network: IP, ICMP, ARP, IPX, AppleTalk, OSPF, BGP, RIP, MPLS-VPN, EVPN

Link: Ethernet (802.3), Wi-Fi (802.11), Token Ring (802.5), Bluetooth, Zigbee, Frame Relay, CSMA/CD, Token passing

Physical: ADSL (G.992), Coaxial cable, Twisted pair copper, Fiber optic, Wireless transmission media