# Course Summary
## Networks and Communication IK1203

## Chapter 1 – Roadmap

### Internet transport protocols services:

Protocols define format and order of messsges sent and received among network entities, and actions taken on message transmission and reception.

### TCP

- **Reliable transport** between sending and recievig processes
- **Flow control** sender won't overhelm reciever
- **Congestion control** throttle sender when network overloaded
- **Connection-oriented** setup required between client and server processes

- **Does not provide** timing, minimum throughput guarantee, security

**UDP**

- **Unreliable data transfer** between sending and receiving processes

- **Does not provide** reliability, flow control, congestion control, timing, througput guarantee, security or connection setup.

---

### Internet protocol stack:

**Application layer:** supporting network applications

- FTP
- SMTP
- HTTP

**Transport layer:** process – process data transfer

- TCP
- UDP

**Network layer:** routing of datagrams from source to destination.

- IP
- Routing protocols

**Link layer:** data transfer between neighgboring network elements

- Ethernet
- 802.11 (WiFi)
- PPP

**Physical layer:** bits “on the wire”

- Fiber
- Copper
- Radio

**Transmission and propagation delay**

$$Packet\ transmission\ delay = \frac{L(bits)}{R(bits/sec)}$$

Time from A starts sending until B has received entire message:

$$T = T_T + T_P = \frac{L}{r} + \frac{d}{s}$$

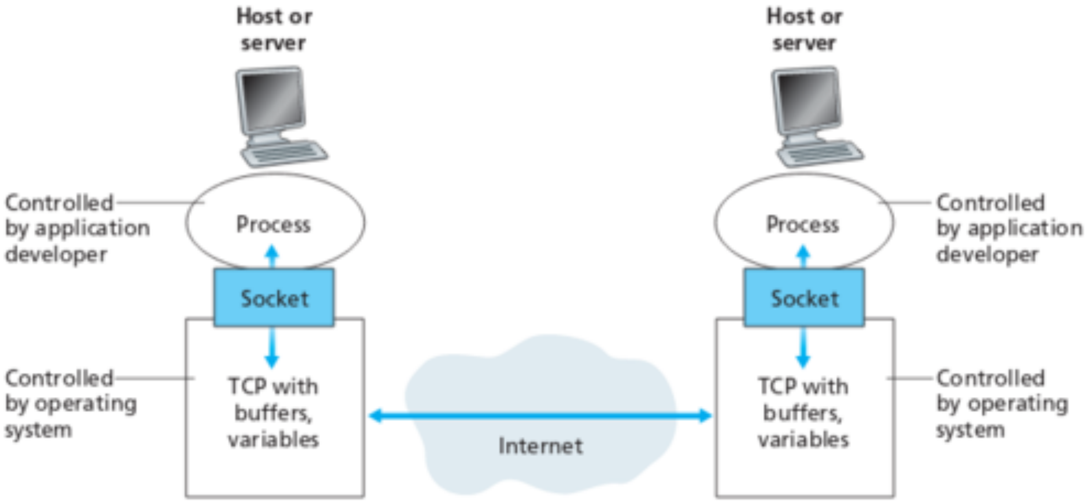
$L = message\ size\ (bit)$
$r = link\ rate\ (bit/s)$
$s = propagation\ speed\ (m/s)$
$d = distance\ (m)$

---

# Chapter 2 - Application layer

Processes must have identifiers, IP-adress and port number.

An application consists of pair of processes which communicate over the network (client-server). These processes send and receive messages, into and from the network through a software interface called **socket**. Considering the analogy presented in the book "Computer Networking: Top Down Approach". There is a house that wants to communicate with other house. Here, house is analogous to a process, and door to a socket. Sending process assumes that there is an infrastructure on the other side of the door that will transport the data to the destination. Once the message is arrived on the other side, it passes through receiver's door (socket) into the house (process). This illustration from the same book can help you:


Host or server
Host or server
Controlled by application developer
Process
Socket
Controlled by operating system
TCP with buffers, variables
Internet
Process
Socket
TCP with buffers, variables
Controlled by application developer
Controlled by operating system

Sockets are part of transport layer, which provides logical communication to applications. This means that from application's point of view both hosts are directly connected to each other, even though there are numerous routers and/or switches between them. Thus a socket is not a connection itself, it's the end point of the connection. Transport layer protocols are implemented only on hosts, and not on intermediate routers.

**Ports** provide means of internal addressing to a machine. The primary purpose is to allow multiple processes to send and receive data over the network without interfering with other processes (their data). All sockets are provided with a port number. When a segment arrives to a host, the transport layer examines the destination port number of the segment. It then forwards the segment to the corresponding socket. This job of delivering the data in a transport layer segment to the correct socket is called **de-multiplexing**. The segment's data is then forwarded to the process attached to the socket.

## DNS Lookup

### Iterative (non-recursive)

Iterative DNS queries are ones in which a DNS server is queried and returns an answer without querying other DNS servers, even if it cannot provide a definitive answer. Iterative queries are also called non-recursive queries.

### Recursive

Recursive DNS queries occur when a DNS client requests information from a DNS server that is set to query subsequent DNS servers until a definitive answer is returned to the client. The queries made to subsequent DNS servers from the first DNS server are iterative queries.

## Types of DNS servers

**Root name server -** Contacted by local name server when it cannot resolve name

Maintains database of TLD (top-level domain) servers
When contacted, returns list of DNS servers for TLD in question

---

**Local DNS name server** - does not strictly belong to hierarchy

each ISP (residential ISP, company, university) has one. Also called "default name server". Part of a host's network configuration

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.

---

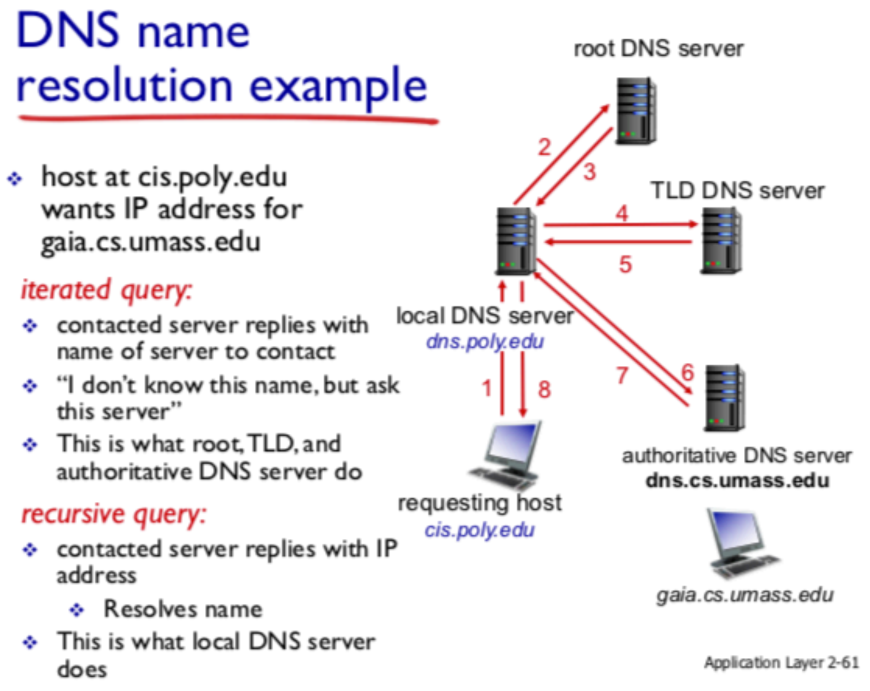
**TLD DNS Server** - responsible for com, org, net, edu, aero, jobs, museums, and all top-level country domains, e.g.: uk, fr, ca, jp.

When contacted, returns list of authoritative DNS servers for organization in question

---

**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.

---



DNS name resolution example

- host at cis.poly.edu wants IP address for gaia.cs.umass.edu

*iterated query:*

- contacted server replies with name of server to contact
- "I don't know this name, but ask this server"
- This is what root, TLD, and authoritative DNS server do

*recursive query:*

- contacted server replies with IP address
  - Resolves name
- This is what local DNS server does

**HTTP response status codes**

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

**301 Moved Permamently** - requested object moved, new location specified later in this message (Location:)

**400 Bad Request -** request message not understood by server

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

**505 HTTP** Version Not Supported

HTTP is stateless - server maintains no information about past client requests.

# Chapter 3 - Transport layer

## Transport layer:

Transport layer: logical communication between processes. Relies on, and enhances, network layer services.

Network layer: logical communication between hosts

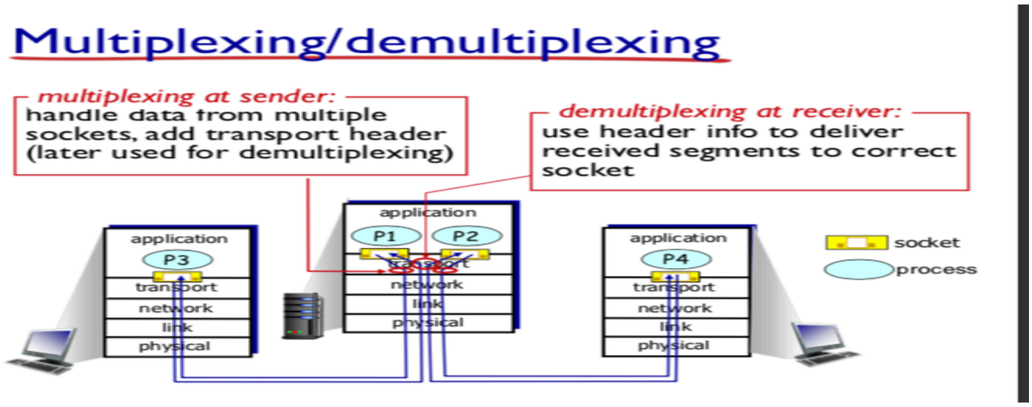
**Transport services and protocols:**

Provide *logical communication* between app processes running on different hosts

Transport protocols run in end systems

Send side: breaks app messages into *segments*, passes to network layer

Rcv side: reassembles segments into messages, passes to app layer



Multiplexing/demultiplexing

*multiplexing at sender:*
handle data from multiple sockets, add transport header (later used for demultiplexing)

*demultiplexing at receiver:*
use header info to deliver received segments to correct socket

Host uses IP addresses and port numbers to direct segment to appropriate socket

---

### Go-back-N

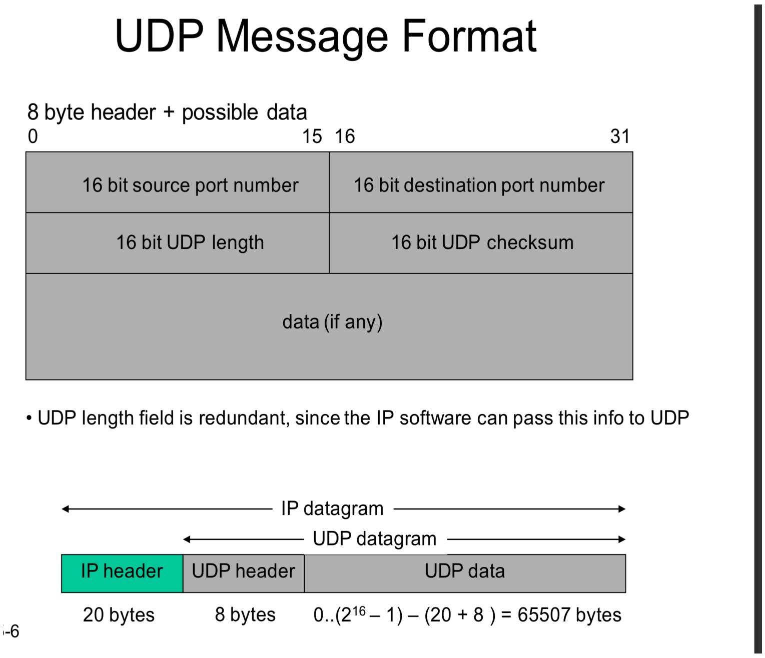
Sender can have up to N unacked packets in pipeline
Receiver only sends cumulative ack
Doesn't ack packet if there's a gap
Sender has timer for oldest unacked packet
When timer expires, retransmit all unacked packets

---

## UDP – User Datagram Protocol

- Datagram-oriented transport layer protocol
- Provides connectionless unreliable service
- Provides optional end-to-end checksum covering header and data
- Provides no feedback to control data rate
- An UDP datagram is silently discarded if checksum errors
- UDP messages can be lost, duplicated, or arrive out of order

---



UDP Message Format

8 byte header + possible data

| 0 15 | 16 31 |
|---|---|
| 16 bit source port number | 16 bit destination port number |
| 16 bit UDP length | 16 bit UDP checksum |
| data (if any) | |

- UDP length field is redundant, since the IP software can pass this info to UDP

IP datagram
UDP datagram

| IP header | UDP header | UDP data |
|---|---|---|
| 20 bytes | 8 bytes | 0..($2^{16}$ – 1) – (20 + 8 ) = 65507 bytes |

-6

---

## TCP – Transmission Control Protocol

- TCP is a connection-oriented transport protocol

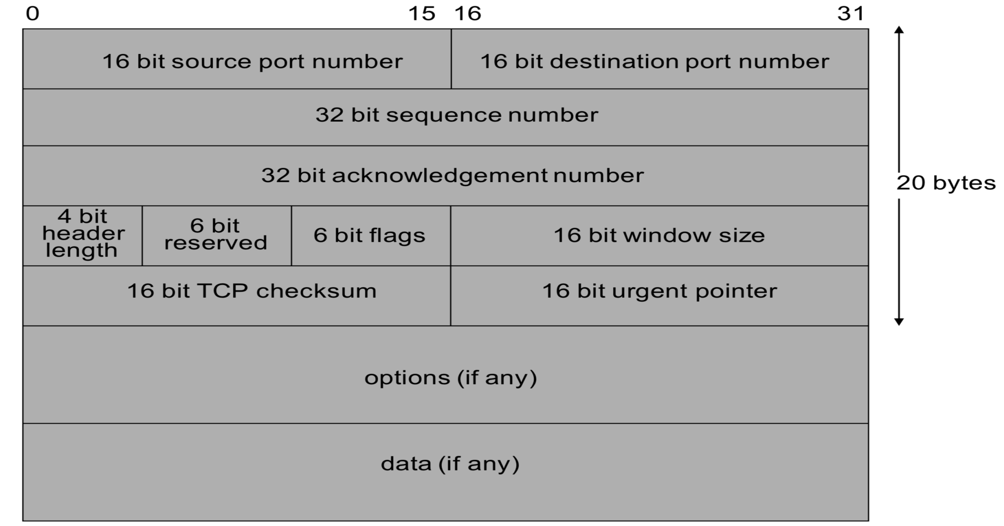
- A TCP connection is a full duplex connection between exactly two end- points

– Broadcast and multicast are not applicable toTCP

- TCP provides a reliable byte stream service
  - – A stream of 8-bit bytes is exchanged across the TCP connection
  - – No record markers inserted by TCP
  - – The receiving (reading) end cannot tell what sizes the individual writes were at the sending end
  - – TCP decides how much data to send (not the application); each unit is a segment

- Lots of applications have been implemented on top of TCP
  - – TELNET(virtual terminal), FTP(file transfers), SMTP(email), HTTP

## TCP Segment



cture 5-6

## TCP Flags The 6 flag bits in the TCP header are:

**ACK** - Acknowledgement number valid

**RST** - Reset the connection

**SYN** - Synchronize sequence numbers to initiate connection

**FIN** - Sender is finished sending data

(**URG** - Urgent pointer valid, set when sender wants the receiver to read a piece of data urgently and possibly out of order)

(**PSH** - Receiver should immediately pass the data to the application, buffers should be emptied, note difference compared to URG)

## Maximum Segment Size

- MSS (Maximum Segment Size) is the largest chunk of data TCP will send to the other side

- MSS can be announced in the options field of the TCP header during connection establishment

- If MSS is not announced, a default value of 536 is assumed
    - 576 bytes is min MTU for IP networks, subtract 20B IP hdr and 20B TCP hdr → default MSS 536 to avoid IP fragmentation

- In general, the larger MSS the better until fragmentation occurs

---

# Flow control

Flow control defines the amount of data a source can send before receiving an acknowledgement from receiver

TCP uses a sliding window protocol to accomplish flow control

For each TCP connection (always duplex), the sending and receiving TCP peer use this window to control the flow

---

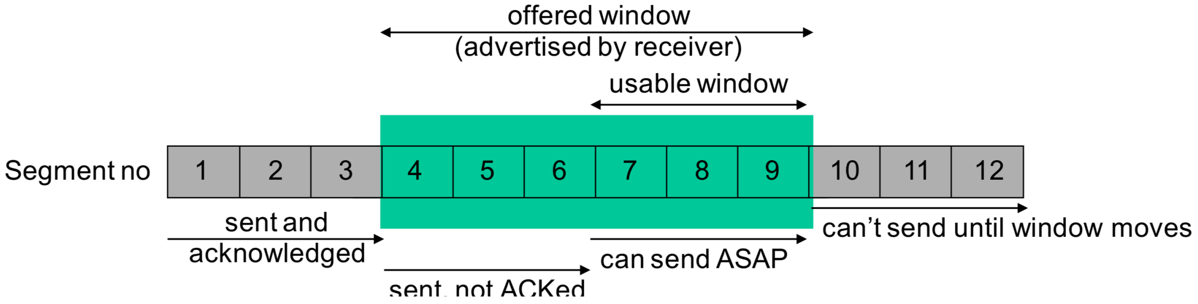
## Sliding windows

- Receiver: offered window

    – acknowledges data sent and what it is prepared to receive
    – receiver can send an ACK, but with an offered window of 0
    – later, the receiver sends a window update with a non-zero offered window size

- Sender: usable window

    - how much data it is prepared to send immediately



## Nagle’s Algorithm

- Each TCP connection can have only one outstanding (i.e. unacknowledged) small segment (i.e. a tinygram)
- While waiting - additional data is accumulated and sent as one segment when the ACK arrives, or when maximum segment size can be filled

## Bandwith-Delay Product

How large should the window be for optimal throughput?
Calculate the capacity of the ”pipe” as:

capacity(bits) = bandwidth(bits/sec) x RTT(sec)

This is the size the receiver advertised window should have for optimal throughput.
Example:

T1 connection across the US: capacity = 1.544Mbit/s x 60ms = 11,580

# Congestion Control

• Congestion Window – Sender’s window size is not only determined by the receiver, but also by the congestion in the network

• Sender maintains 2 window sizes:

  - Receiver-advertised window
  - Congestion window(CWND)

• Actual window size = min(rcv window, CWND)

• To deal with congestion, sender uses several strategies:

  - Slowstart
  - Additive increase of CWND
  - Multiplicative decrease of CWND

## Slow Start

• At beginning of connection, CWND=MSS

• For each ACKed segment, CWND is increased by one MSS

  - Until a threshold of half allowable window size is reached

• Process not slow at all

  - window size grows exponentially

- CWND=1
- Send 1 segment
- Receive 1 ACK, and increase CWND  to 2
- Send 2 segments
- Receive ACK for 2 segments, and increase CWND to 4

## Congestion Avoidance

- Slow start will increase CWND size exponentially until threshold value is reached

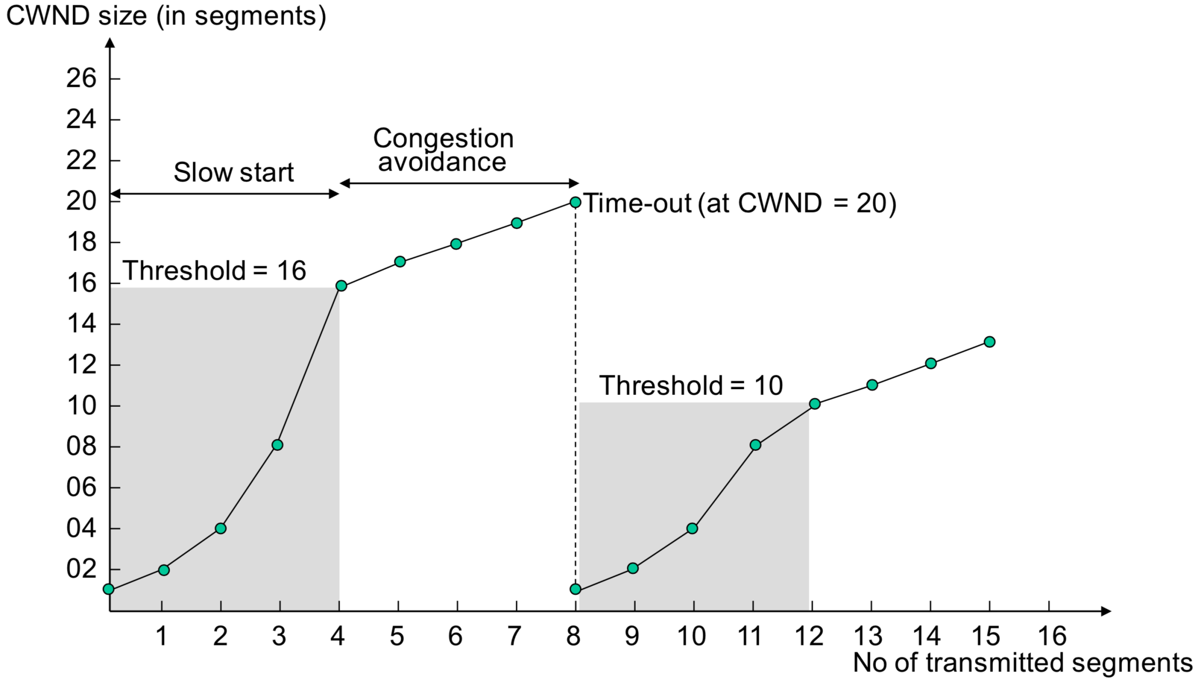
- To avoid congestion this exponential growth must slow down:

  - After the threshold is reached, CWND will be increased additively (one segment per ACK, even if several segments are ACKed)

- This linear growth will continue until either:
  - Receiver-advertised window size is reached, or
  - ACK timeout occurs -> TCP assumes congestion

- When congestion occurs:
  - Threshold value is set to1/2 the last CWND
  - CWND restarts from one MSS again

- Initial threshold value is 65535 bytes, minimum value is 512 bytes

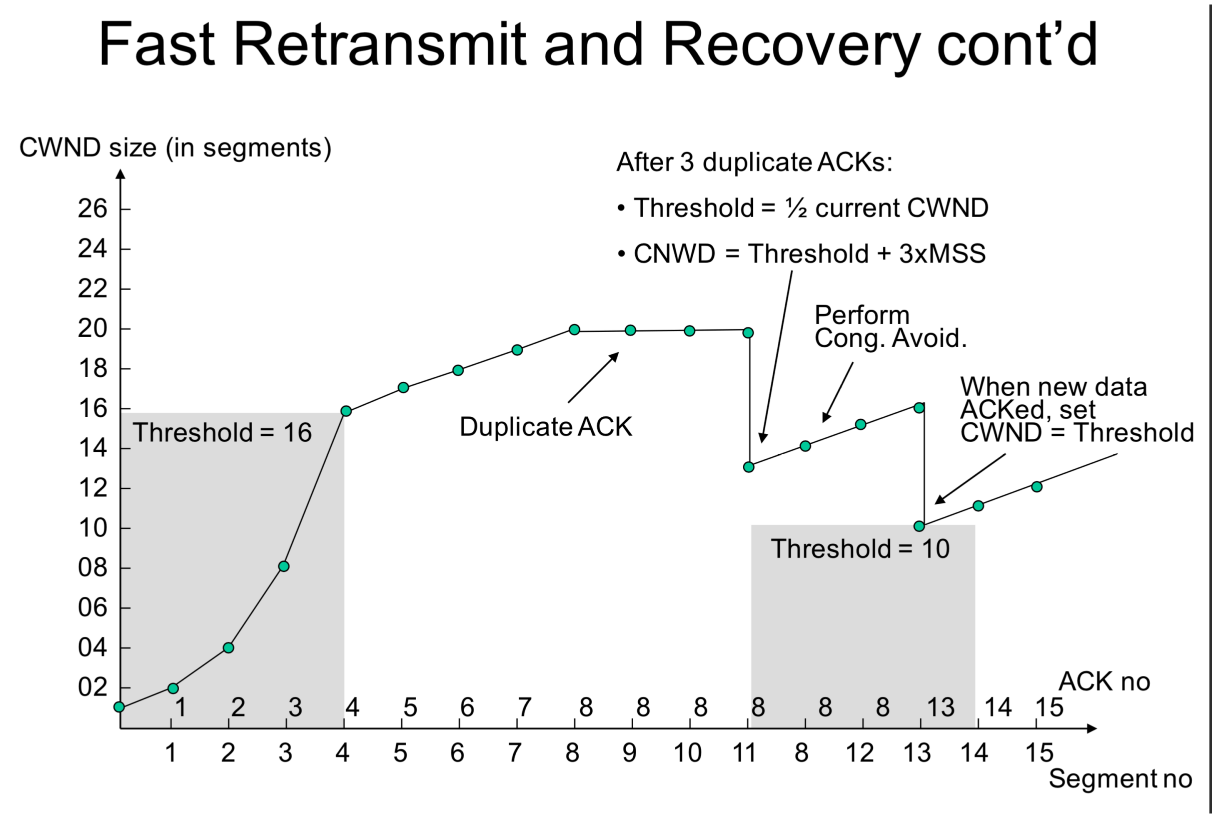


### Fast Retransmit and Fast Recovery

TCP is required to generate an immediate ACK (a duplicate ACK) when an out-of-order segment is received, to inform sender what segment number that is expected.

- Cause of duplicate ACK could be
  - lost segment, or
  - reordering of segments
  - but data is still flowing between the two ends!!

- If 3 (or more) duplicate ACKs are received in a row:
  - Retransmit immediately (before time-out) - Fast Retransmit
  - Do congestion avoidance (not slow start) - Fast Recovery



## Fast Retransmit and Recovery cont’d

After 3 duplicate ACKs:

• Threshold = ½ current CWND

• CNWD = Threshold + 3xMSS

Flow control is a way for the receiver to regulate the sender’s transmission pace. The purpose is to prevent the receiver from becoming overwhelmed with more data than it can process. Congestion control deals with the situation when packets are dropped by routers along the path between sender and receiver. The purpose of TCP congestion control is to regulate the sender’s transmission pace based on the current network conditions.

# Chapter 4 - Network layer

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

**Routing**: determine route taken by packets from source to destination

**Analogy:**

**Routing**: process of planning trip from source to destination

**Forwarding**: process of getting through single interchange

## IP Fragmentation / reassembly

- Network links have MTU (max. transmission unit) - largest possible link-level frame
- Different link types, different MTUs
    - Ethernet – 1500 byte
- IP datagram needs to be divided ("fragmented") if it is larger than link MTU
    - one datagram becomes several datagrams
    - "reassembled" only at final destination
    - IP header bits used to identify fragments and put them into order

## Subnets:

192.168.60.55/20 (255.255.240.0)

| | | | | |
|---|---|---|---|---|
| Network ID: | 192 | 168 | 48 | 0 |
| Broadcast IP: | 192 | 168 | 63 | 255 |
| Usable IPs: | 192.168.48.1 to 192.168.55.254 | | | |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |

```
IP =        11000000.10101000.00111100.00110111
Mask =      11111111.11111111.11110000.00000000
                             .00110000
```

To get the broadcast ID, look at the last 1 in the mask. In this case it represents the number 16. Add 16 to 48 and subtract 1. That's the broadcast ID.

**172.10.85.60/22**

**network ID = 172.10.84.0**
**broadcast ID = 172.10.87.255**
**IPs = 172.10.84.1 to 172.10.87.254**

```
IP     = xxxxxxxx.xxxxxxxx    .01010101.
Mask   = 11111111.11111111.11111100.00000000
                              .01010100
                                 84
```

---

# Routing

**Popular Routing Protocols:**

**RIP**
**OSPF**
**BGP**

---

**Routing Information Protocol – RIP**

- Metric is Hop Counts
  - 1: - Directly connected
  - 16: - Infinity
  - RIP cannot support networks with diameter > 15
- RIP uses distance vector
  - RIP messages contain a vector of hop counts.
  - Every node sends its routes to its neighbours
  - Route information gradially spreads through the network – Every node selects the route with smallest metric.

---

**Disadvantages with RIP**

- Slow convergence
  - Changes propagate slowly
  - Each neighbor only speaks ~ every 30 seconds; information propagation time over several hops is long
- Instability
  - After a router or link failure RIP takes minutes to stabilize.
- Hops count may not be the best indication for which is the best route.
- The maximum useful metric value is 15
  - Network diameter must be less than or equal to 15.

- RIP uses lots of bandwidth
    - It sends the whole routing table in updates.

---

## Open Shortest Path First - OSPF

- OSPF is a link-state protocol
    - Builds Link State Advertisements (LSAs)
    - Distributes LSAs to all other routers
    - Computes delivery tree using the Dijkstra algorithm

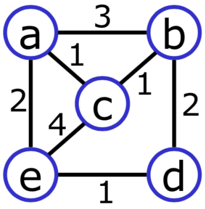
- OSPF uses IP directly (protocol field = 89)
    - Not UDP or TCP

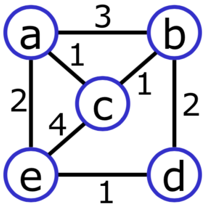
- OSPF networks are partitioned into areas to minimize cross-area communication.

---

## Dijkstra Algorithm (Shortest Path First)

Find shortest paths from ”a” to all other nodes!





| *M* | $D_b$ (path) | $D_c$ (path) | $D_d$ (path) | $D_e$ (path) |
|---|---|---|---|---|
| {a} | 3 (a-b) | 1 (a-c) | ∞ (--) | 2 (a-e) |
| {a, c} | 2 (a-c-b) | **1 (a-c)** | ∞ (--) | 2 (a-e) |
| {a, c, b} | **2 (a-c-b)** | **1 (a-c)** | 4 (a-c-b-d) | 2 (a-e) |
| {a, c, b, e} | **2 (a-c-b)** | **1 (a-c)** | 3 (a-e-d) | **2 (a-e)** |
| {a, c, b, e, d} | **2 (a-c-b)** | **1 (a-c)** | **3 (a-e-d)** | **2 (a-e)** |

---

## Border Gateway Protocol - BGP

- Inter-domain routing
- Simple cases: use static routing
- Main purpose: Network reachability between autonomous systems
- BGP version 4 is the exterior routing protocol used in the Internet today.
- BGP uses TCP

- TCP is reliable: reduces the protocol complexity

- BGP uses path-vector - enhancent of distance-vector.
- BGP implements policies – chosen by the local administrator.

---

Autonomous Systems

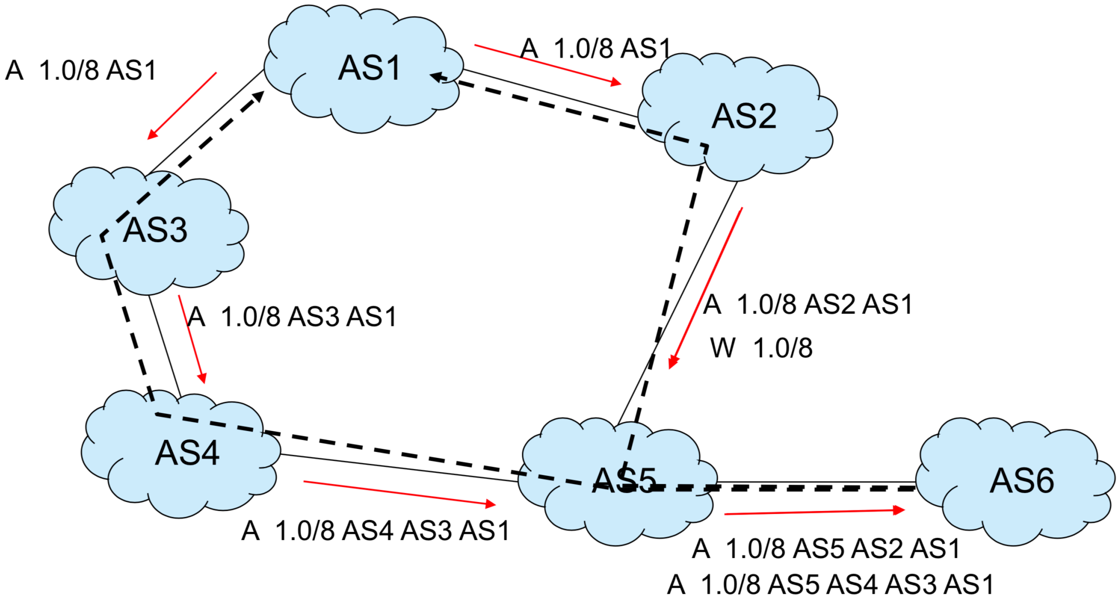
- An Autonomous system is generally administered by a single entity.
  - Operators, ISPs (Internet Service Providers)

- An AS contains an arbitrary complex sub-structure.
- Each autonomous system selects the routing protocol to be used within the AS.
- Policies or updates within an AS are not propagated to other AS:s.
- An AS-number is (currently) a 16-bit unique identifier
- Interconnwection between AS:s
  - Service Level Agreements (SLA:s)
  - Internet Exchange Points (IX:s)
  - Network Access Points (NAPs)

---

## BGP Simple example

- AS1 has a network 1.0.0.0/8 that it announces



---

# Chapter 5 – Link Layer

Layer-2 packet: encapsulate datagram into frame, adding header and trailer

| Name | EDC Length | Description | Examples |
|---|---|---|---|
| **Even Parity** | **1 bit** | Set so total number of one's is even. Detects single bit errors. | **Memory, RAID Disks.** |
| **Internet checksum** | **16 bits** | **16-bit sum** | **IP, TCP, UDP** |
| **CRC-32** | **32 bits** | Cyclic code (polynomial division). Detects all burst errors up to 32 bits long (and some longer). | **Ethernet** |

**Multiple Access Protocols**

Two types of links:

- Point-to-Point
  - point-to-point link between Ethernet switch and host

- Broadcast (shared wire or medium)
  - Old Ethernet
  - 802.11 wireless LAN

**Multiple access protocol**

- Distributed algorithm that determines how nodes share channel, i.e. determines when a node can transmit

Examples of random access MAC protocols:

- slotted ALOHA
- ALOHA
- CSMA, CSMA/CD, CSMA/CA

## CSMA (carrier sense multiple access)

CSMA: sense (listen) before transmit:

- If channel sense idle: transmit entire frame
- If channel sensed busy, defer transmission

- Human analogy: don't interrupt others.

### CSMA Collisions

- Collisions can still occur:
  - propagation delay means two nodes may not hear each other's transmission

- Collision: entire packet transmission time wasted
  - distance & propagation delay play role in in determining collision probability

---

## 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 and received signals
  - difficult in wireless LANs: received signal strength overwhelmed by local transmission strength

---

## Summary of MAC protocols

- Channel partitioning, by time, frequency or code
  - Time Division, Frequency Division

- Random access (dynamic),
  - ALOHA, S-ALOHA, CSMA, CSMA/CD
  - Carrier sensing: easy in some technologies (wire), hard in others (wireless)
  - CSMA/CD used in Ethernet
  - CSMA/CA used in 802.11

- Taking turns
    - Polling from central site, token passing
    - bluetooth, FDDI, 802.11 with PCF (Point Coordination Function), token ring

---

## ARP: address resolution protocol

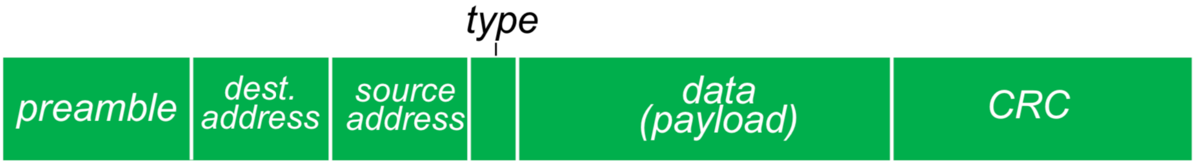
- ARP table: each IP node (host or router) on a LAN has a table:
    - IP/MAC address mappings for nodes on the LAN:
    - < IP address; MAC address; TTL>

    - TTL (Time To Live): time after which address mapping will be forgotten (typically 20 min)

---

## Ethernet frame structure

sending adapter encapsulates IP datagram (or other network layer protocol packet) in Ethernet frame



- Preamble:
    - 7 bytes with pattern 10101010 followed by one byte with pattern 10101011
    - used to synchronize receiver and sender clock rates

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

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

---

**Ethernet: unreliable, connectionless**

- connectionless: no handshaking between sending and receiving NIC

- unreliable: receiving NIC doesn’t send acknowledgements (positive or negative) to sending NIC

- Ethernet’s MAC protocol: unslotted CSMA/CD with binary back-off

- Back-off: wait before trying again after a collision
    - Time to wait is randomized
    - After c collisions, wait between 0 and 2c-1 time units

---

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

---

# Chapter 6 - Wireless and Mobile Networks

- 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

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

---

IEEE 802.11: multiple access

- Avoid collisions: 2 nodes or more transmitting at same time
- 802.11: CSMA – sense before transmitting
    - don’t collide with ongoing transmission by other node

- 802.11: *no* collision detection!
    - difficult to receive (sense collisions) when transmitting due to weak received signals (fading)
    - can’t sense all collisions in any case: hidden terminal, fading
    - goal: *avoid collisions:* CSMA/C(ollision)A(voidance)

---

**IEEE 802.11 MAC Protocol: CSMA/CA**

- 802.11 sender

- 1 if sense channel idle for **DIFS** then
    - transmit entire frame (no Collision Detect!)

- 2 if sense channel busy then
    - start random backoff time
    - timer counts down while channel idle
    - transmit when timer expires
    - if no ACK, increase random backoff interval, repeat

- 802.11 reciever
    - if frame received OK
    - return ACK after **SIFS** (ACK needed due to hidden terminal problem)

---

**Avoiding collisions (more)**

idea: allowsenderto“reserve”channel rather than random access of data frames: avoid collisions of long data frames

- 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

---