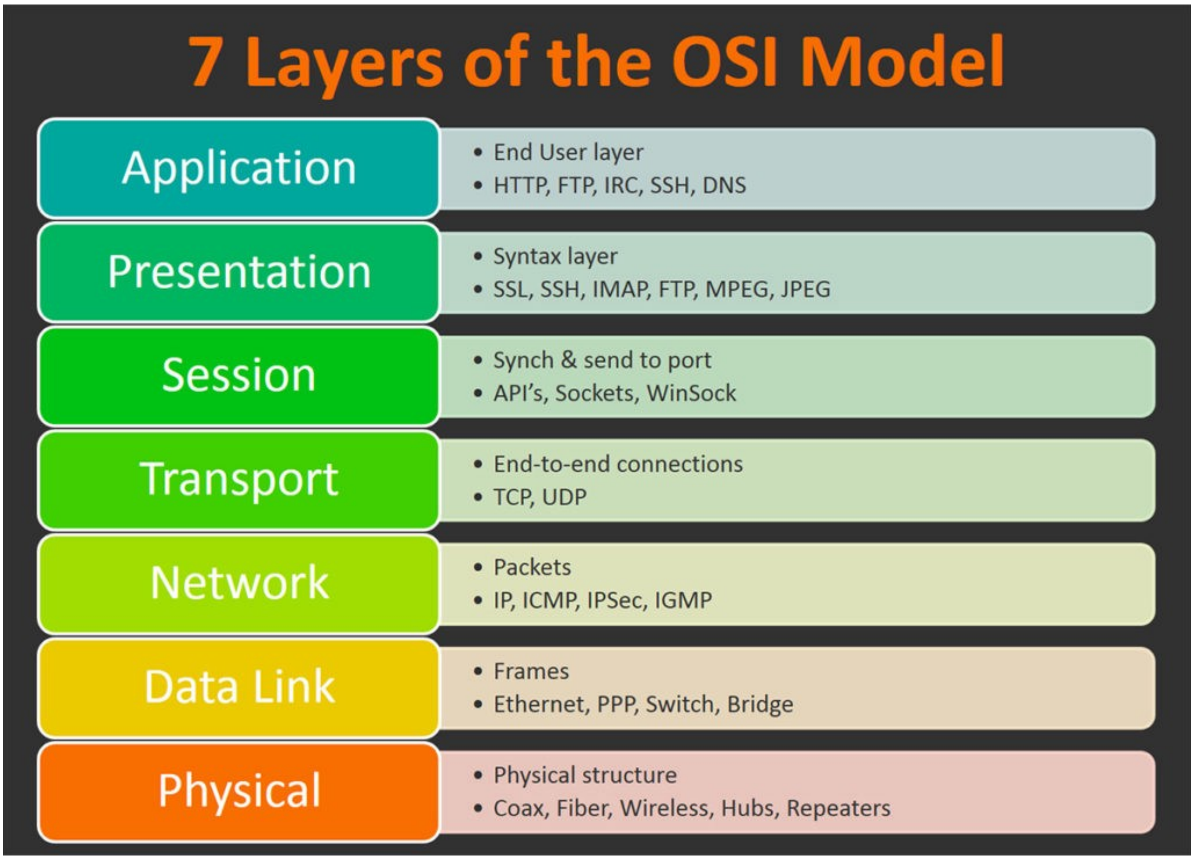
Computer Networks

**Quick refresher



1. Chapter 1 Application Layer

1.1. RFC

RFC (Request for comments) is a memo published by the **IETF** (Internet Engineering Task Force). Not really a request for comments but a politically correct way of publishing.

1.2. Telnet

telnet is a command line interface for communication with a remote device or server. Telnet stands for teletype network.

Telnet can be used to

- Connect to a remote server or device and give commands just like you would be physically sitting in front of the computer
- Manage and configure network devices such a routers and switches
- Check if ports are open and closed on a server.

Telnet is sent in clear text so there is no encryption, if would use it today anyone could grab your data your sending. It should not be used over the public internet. Telnet itself runs on port 23

1.3. File Transfer Protocol (FTP)

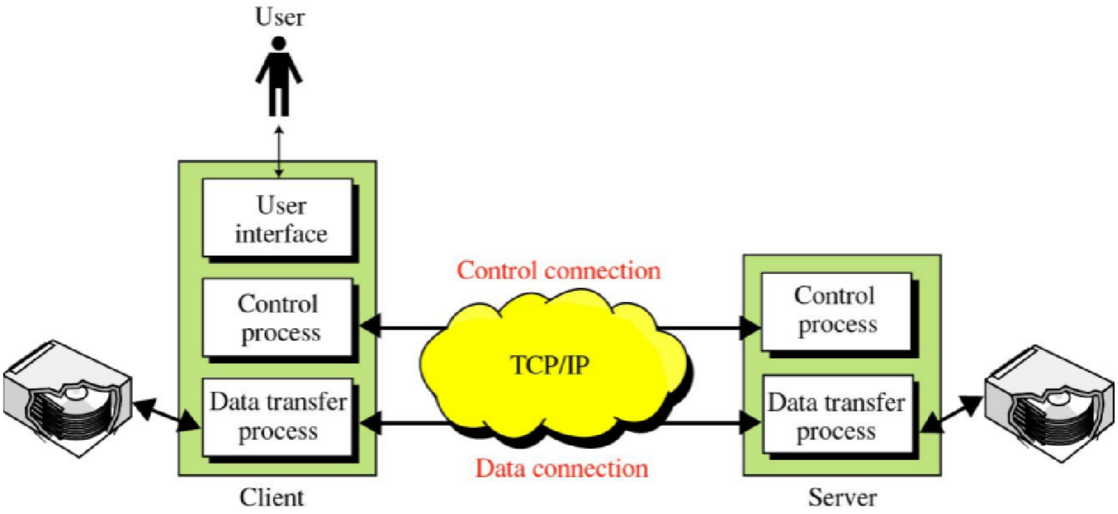
Designed to transfer files and navigate directory listings.

Used two TCP port:

- 21 for control

- 20 for data transfers

Control port uses Telnet protocol to negotiate session



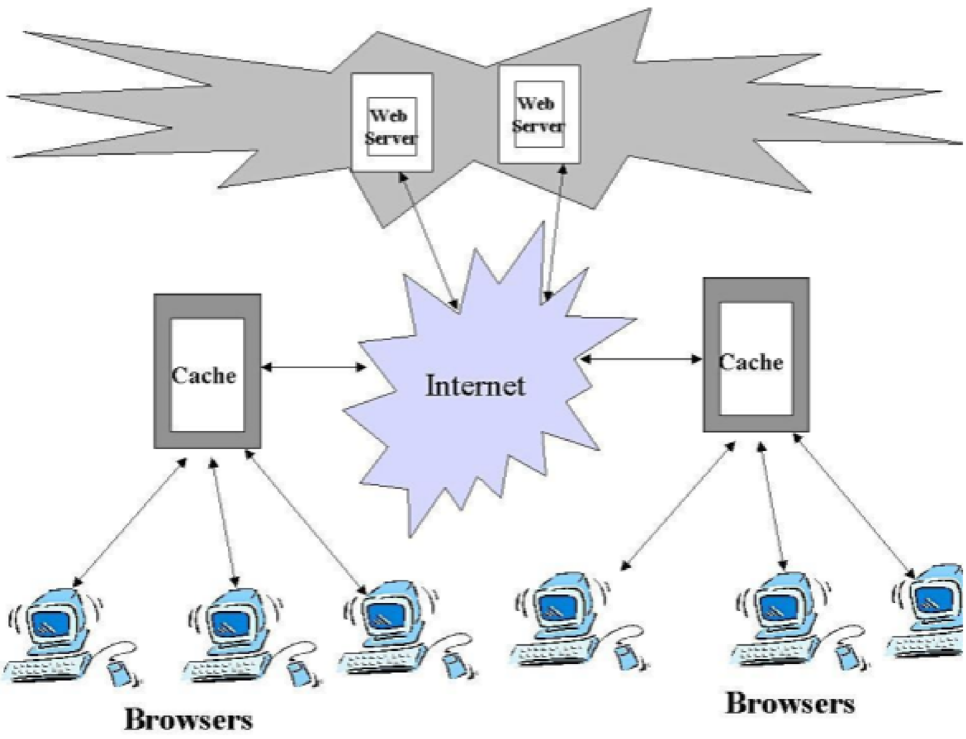
FTP Cheat Sheet

USER – Send username to the FTP server
PASS – Send the password (Anonymous servers need email address)
CWD – Change the working directory on the server
PASV – To enter the passive mode (To let client connect to the server)
RETR – To retrieve a remote file from the server
QUIT – To terminate the connection to the server

1.4. Architectural View of the WWW

Quick refresher for some important parts:

- Clients use web browsers and **HTTP** to retrieve **HTML** files or web pages that are stored on servers at a specified URI.
- Caches improve performance by locally storing the results of client requests. the cache will serve repeat request rather than downloading material again from the servers.



1.5. Uniform Resource Identifiers (URIs)

Web resources need a human-readable identifier in order to support easy retrieval -> **URI**

Most popular form of a **URI** is the **Uniform Resource Locator (URL):**

1.6. Hypertext Markup Language (HTML)

- HyperText Markup Language is a subset of Standardized General Markup Language (SGML)
- Documents use elements to “mark up” or identify sections of text for different purposes or display characteristics.
- Mark up elements are not seen by the user when page is rendered by the web browser.

1.7. Hypertext Transfer Protocol (HTTP)

- The **Hyper Text Transfer Protocol (HTTP)** supports client-server communication using a simple request/response protocol
- It is an application-layer protocol, that operates over **TCP on port 80**
- The protocol is connectionless or stateless and has no notion of session.

1.7.1. HTTP 1.0/1.1 Commands

- **GET** – retrieve document specified by URL
- **PUT** – store specified document under given URL
- **HEAD** – retrieve header from document at URL
- **OPTIONS** – retrieve information about available options
- **POST** – give information to the server under given URL
- **DELETE** – remove document specified by URL

PUT -> create new resource or overwrite it

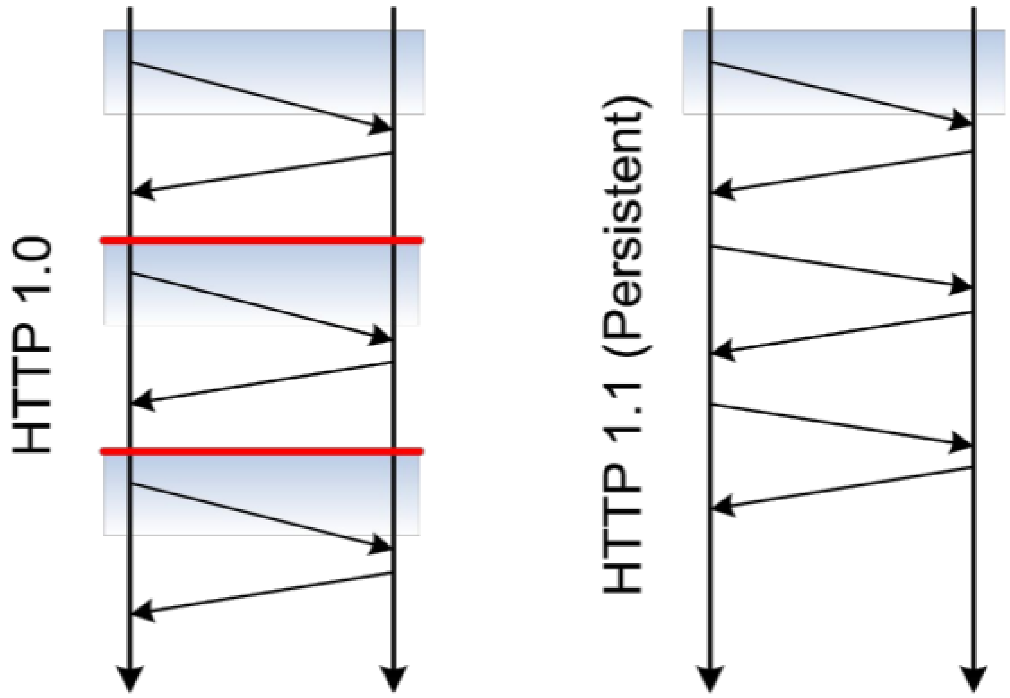
POST -> update existing resource

1.7.2. Inefficiency in HTTP 1.0

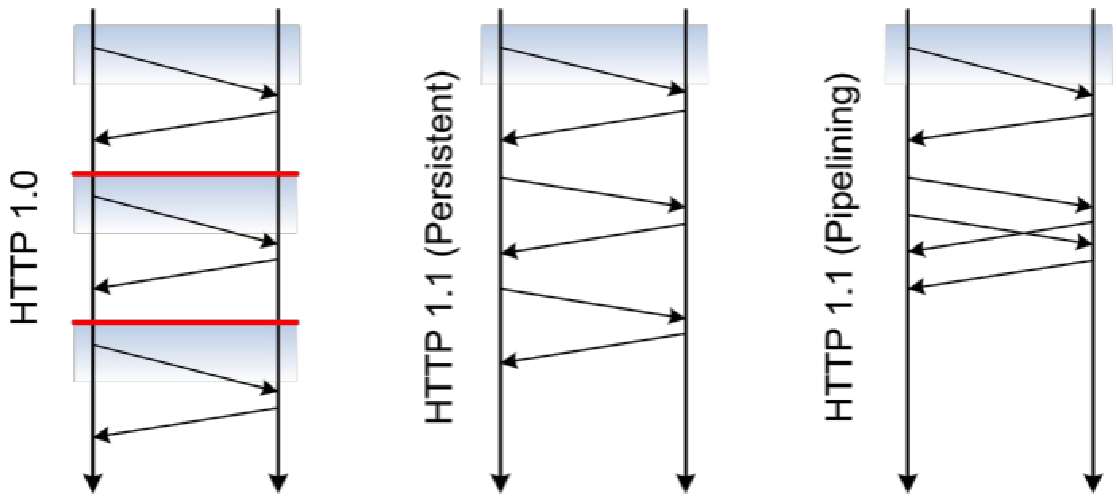
- Each web page contains an average of 40 embedded object(images, audio, video).
- Opening a TCP connection for each object is inefficient, requiring 1 round trip per object
- **TCP** congestion algorithm slowly increases speed, so transfer rate is slower on new connections.

1.7.3. HTTP 1.1 VS HTTP 1.0

Persistent TCP in HTTP 1.1



1.7.4. Pipelining in HTTP1.1



- Final optimization: download each embedded object as soon as we know that we need it, rather than waiting until transfer of its parent completes (**pipelining**).

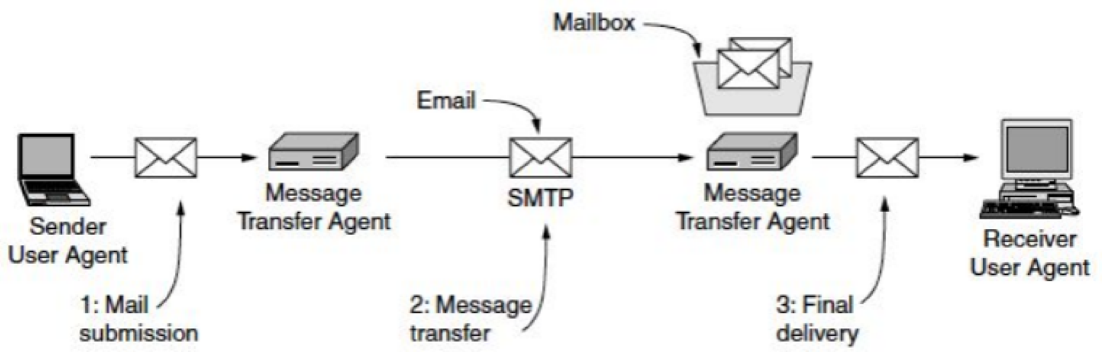
1.8. Simple Mail Transfer Protocol (SMTP)

Email as we know it today separates email exchange into two parts:

- **Mail submission and delivery protocols**, which allow clients to connect to the mail server
- **Mail transfer protocol** which allows mail servers to communicate

→ This specialization allows us flexible access to email, while ensuring standard communication

1.8.1. The Architecture Of Email



1.8.2. Message Format

Header	Meaning
To:	Email address(es) of primary recipient(s)
Cc:	Email address(es) of secondary recipient(s)
Bcc:	Email address(es) for blind carbon copies
From:	Person or people who created the message
Sender:	Email address of the actual sender
Received:	Line added by each transfer agent along the route
Return-Path:	Can be used to identify a path back to the sender

1.8.3. Interaction in command/response mode

Three phases:

1. Handshaking
2. Transfer of messages
3. Closure

1.9. Post Office Protocol (POP)

- A mechanism used by mail clients to talk to mail servers
- A simple **unsynchronized** protocol, difficult to manage on multiple machines.

1.10. Internet Message Access Protocol (IMAP)

- Another mechanism for mail clients to communicate with mail servers.
- All folders (inbox, outbox, drafts, etc...) are **synchronized** on the client and the server.
- Better suited because we use a lot of different devices but higher overhead.

1.11. Web-mail Over HTTP

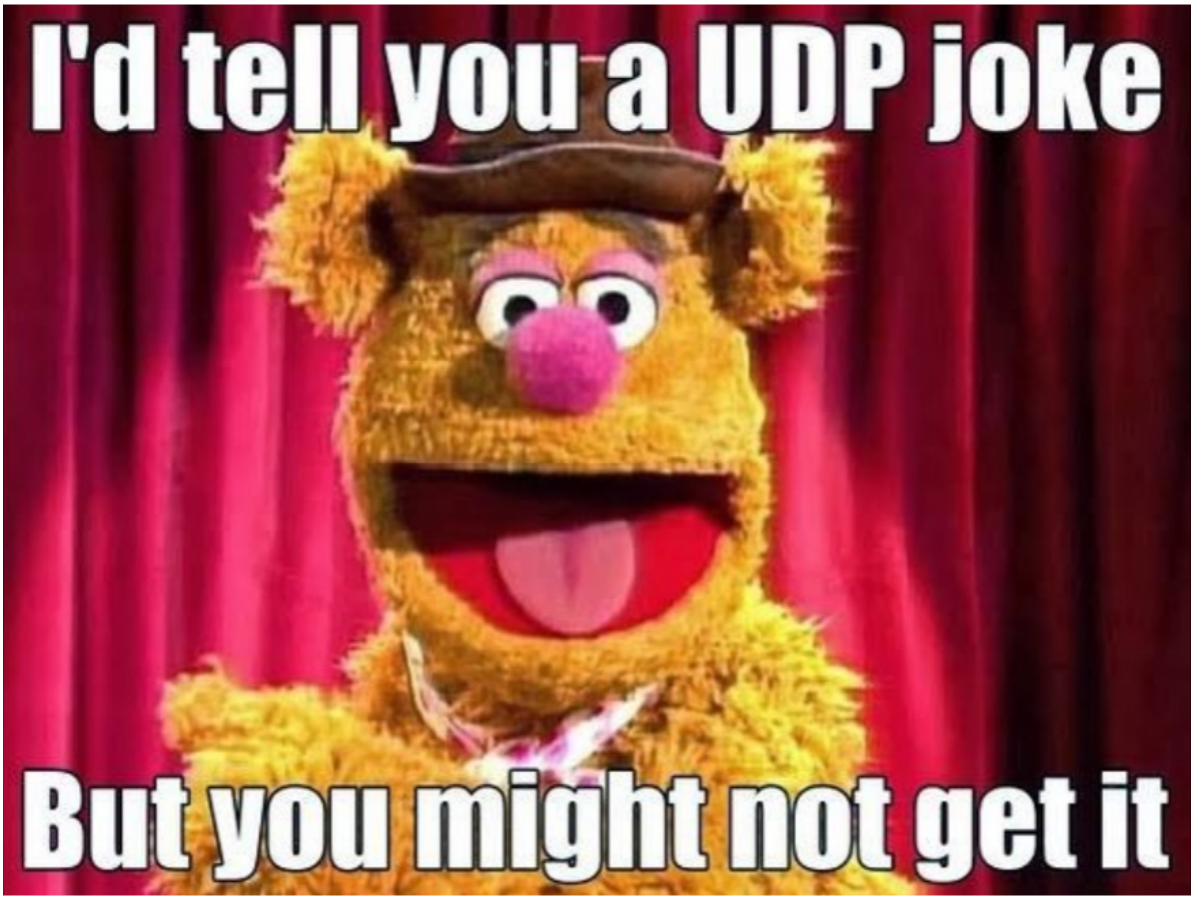
- In this case the web browser is the mail client
- No need for **synchronization** as all files are stored on the server.
- **Lightweight** and allows use on multiple devices, but **requires a consistent Internet connection**

1.12. Dynamic Host Configuration Protocol (DHCP)

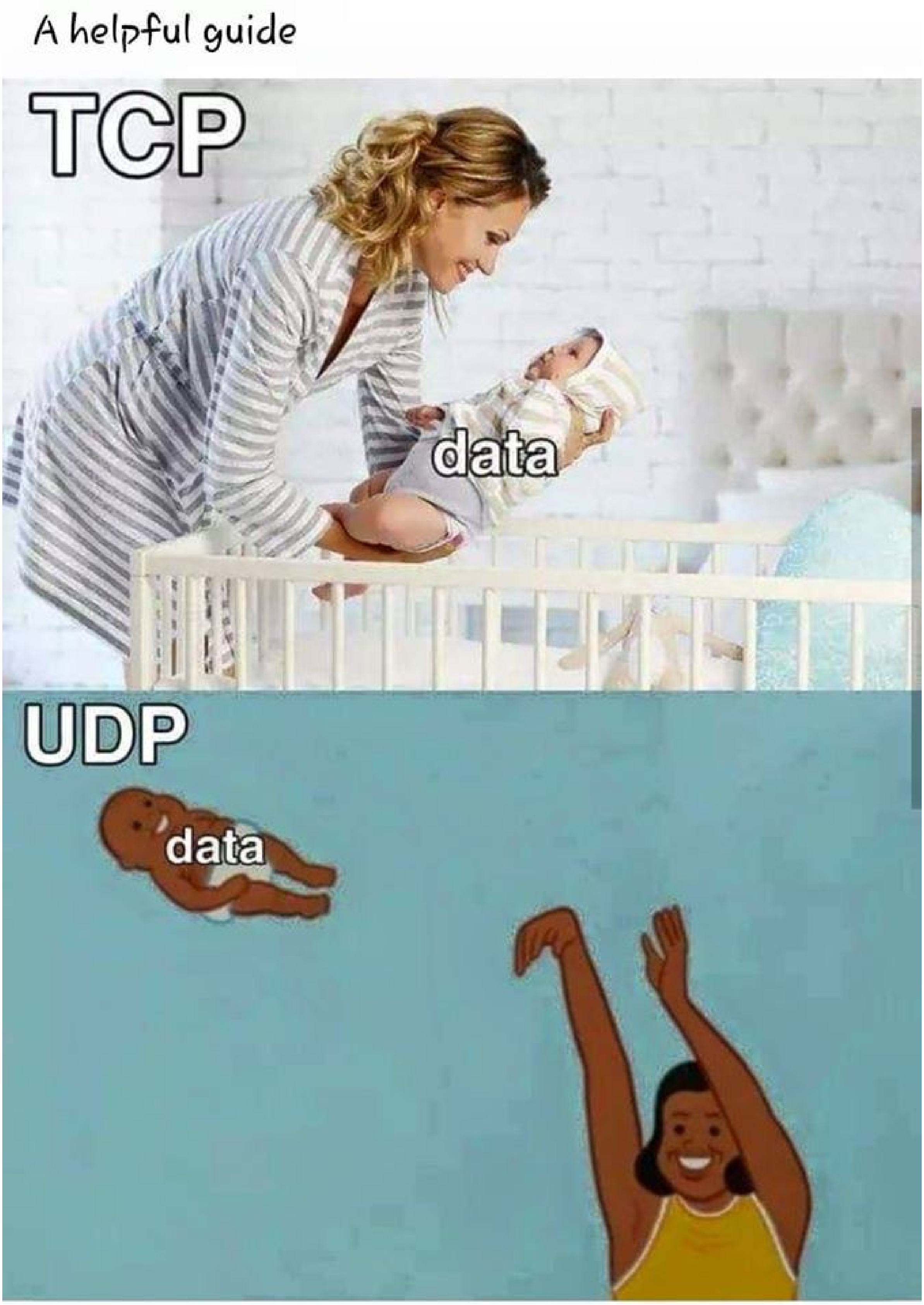
Provides automated assignment of IP addresses:

- IP addresses are assigned on-demand
- Avoid manual IP configuration
- Supports the mobility of a lot of smaller devices

USES UDP

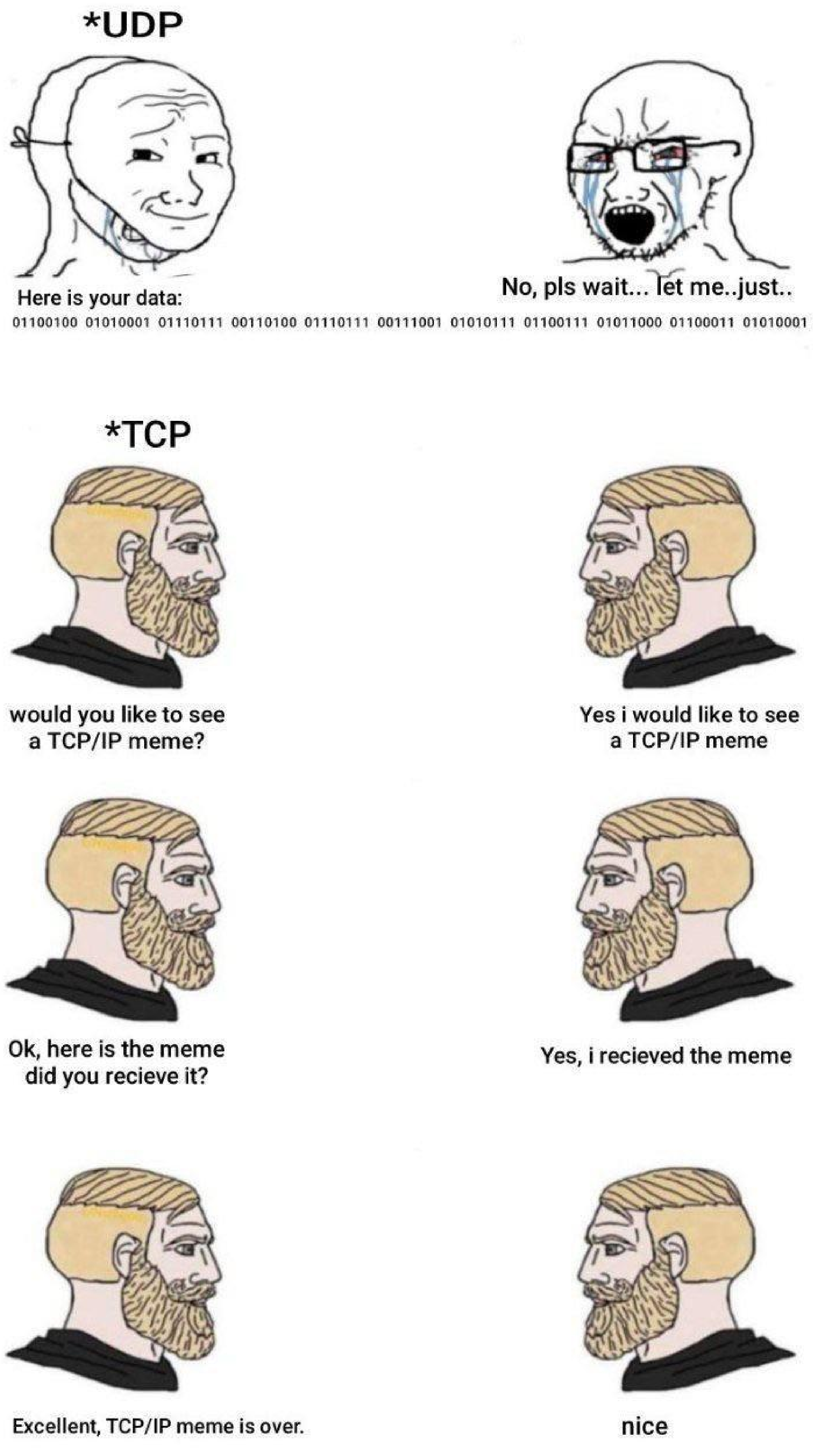


HA MEMES



1.12.1. Verschil tussen UDP en TCP

Als ge u afvroeg wat het verschil is tussen het virgin UDP protocol en het CHAD TCP protocol kijk dan hieronder.


*UDP
Here is your data:
01100100 01010001 01110111 00110100 01110111 00111001 01010111 01100111 01011000 01100011 01010001
No, pls wait... let me..just..
*TCP
would you like to see a TCP/IP meme?
Yes i would like to see a TCP/IP meme
Ok, here is the meme did you recieve it?
Yes, i recieved the meme
Excellent, TCP/IP meme is over.
nice

Dit is Tim nadat hij een TCP packet naar uw mama heeft gestuurd.

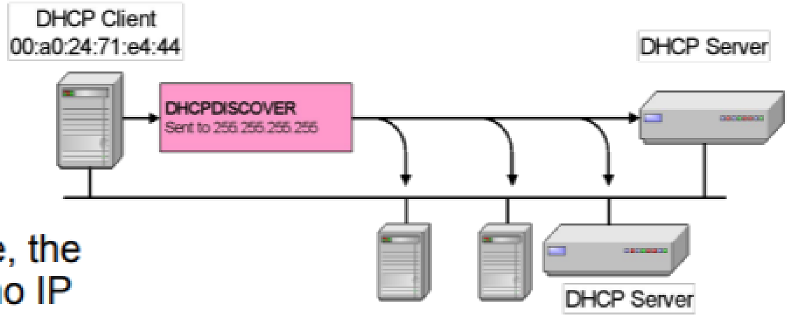
1.12.2. Address Leases

- **DHCP** is not for permanent address allocation
- Addresses are leased for a specific period depending on the environment
  - Shorter leases more efficiently allocate space.
  - –Longer leases lower load on the DHCP server.

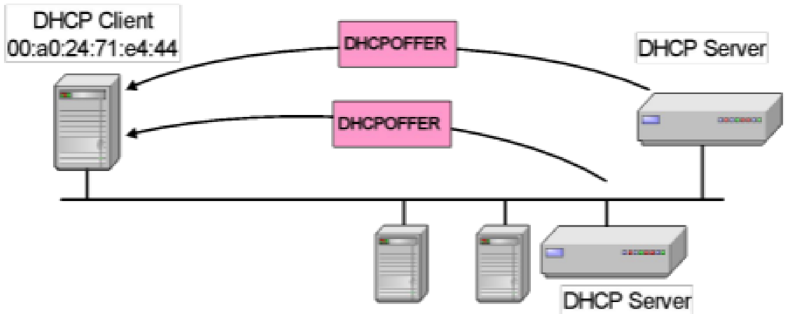
1.12.3. DHCP Message Format

Don't think this is important since there are more than 100 options defined.

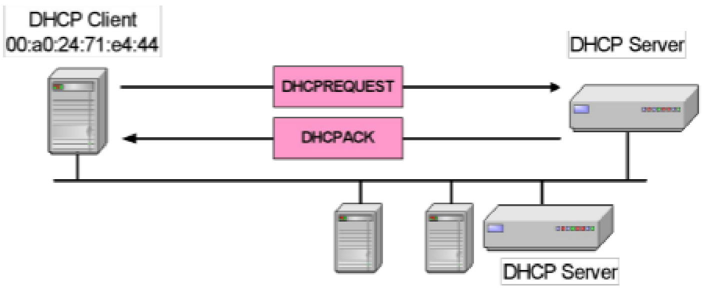
1.12.4. DHCP Discovery



Note: at this stage, the DHCP client has no IP address. All messages are broadcast.

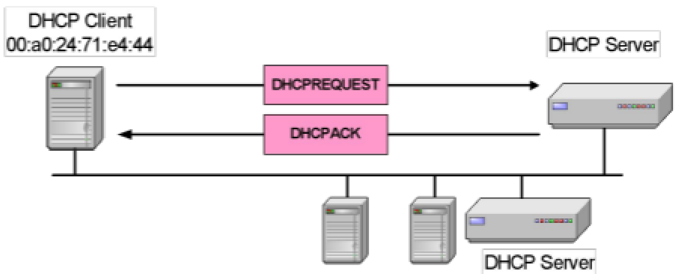


Note: the client now has an IP address and all further communication is via IP.

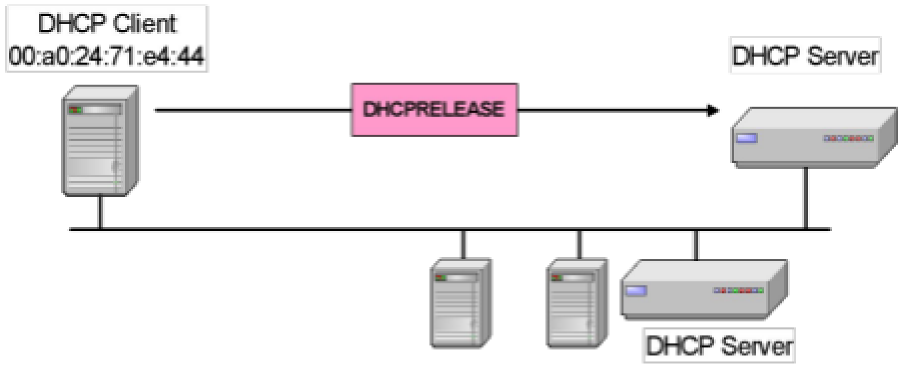


1.12.5. Renewing a Lease

- Leases are renewed when 50% has expired. If DHCP server sends DHCPNACK, then address is released.



1.12.6. Releasing an Address



1.12.7. Duplicate Address Detection

After assignment of an IP address, hosts will perform **duplicate address detection**

⟶ if the clients detects a duplicate address, it refuses the offered IP address with a **DHCPDECLINE** message.

We use **ARP** for this.

1.13. Domain Name System (DNS)

A few notes

- DNS names are not case sensitive, yet URL resource identifiers are:
  - WWW.GOOGLE.COM/index.html (VALID)
  - www.google.com/index.html (VALID)
  - www.google.com/INDEX.HTML (INVALID)
- All DNS names technically end in a dot. We can omit this:
  - http://www.google.com.
  - http://www.google.com

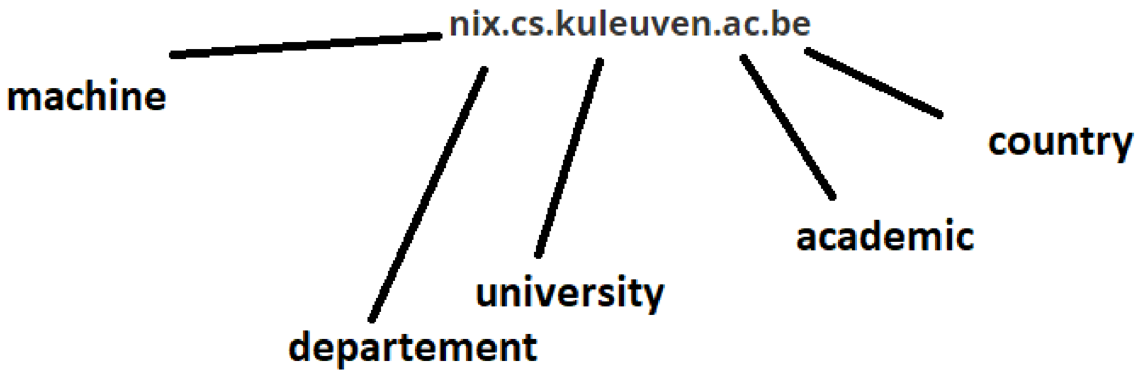
1.13.1. Hierarchical Domain Addresses

We use hierarchy to manage the complexity of DNS lookup and to enable distribution.

Consider a domain name:

nix.cs.kuleuven.ac.be

$\rightarrow$ Hierarchy is embedded in the name from biggest on right(country), to smallest on left(machine).



1.13.2. Top Level Domains

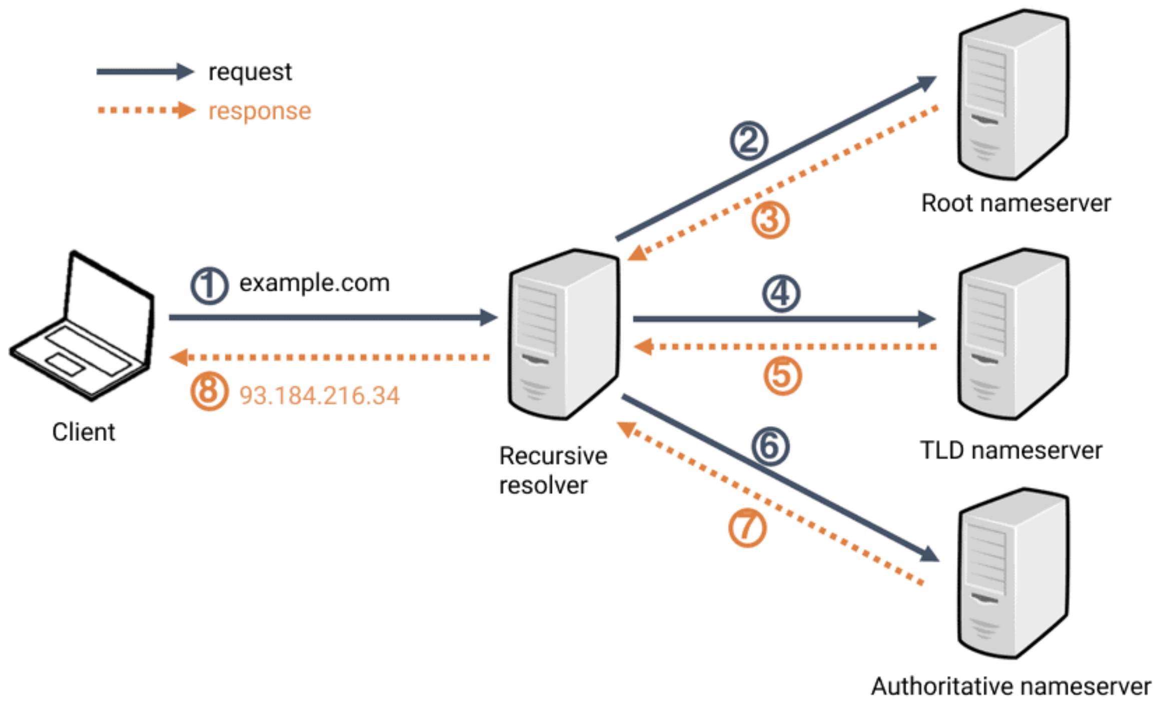
3 groups of Top Level Domains

- 2-letter country codes ex. BE, FR, NL
- generic names (similar organisations)
    - com - commercial organisations
    - org - non commerical organisations
- names of organisations within USA
    - edu - universities
    - gov - US government
    - mil - US army

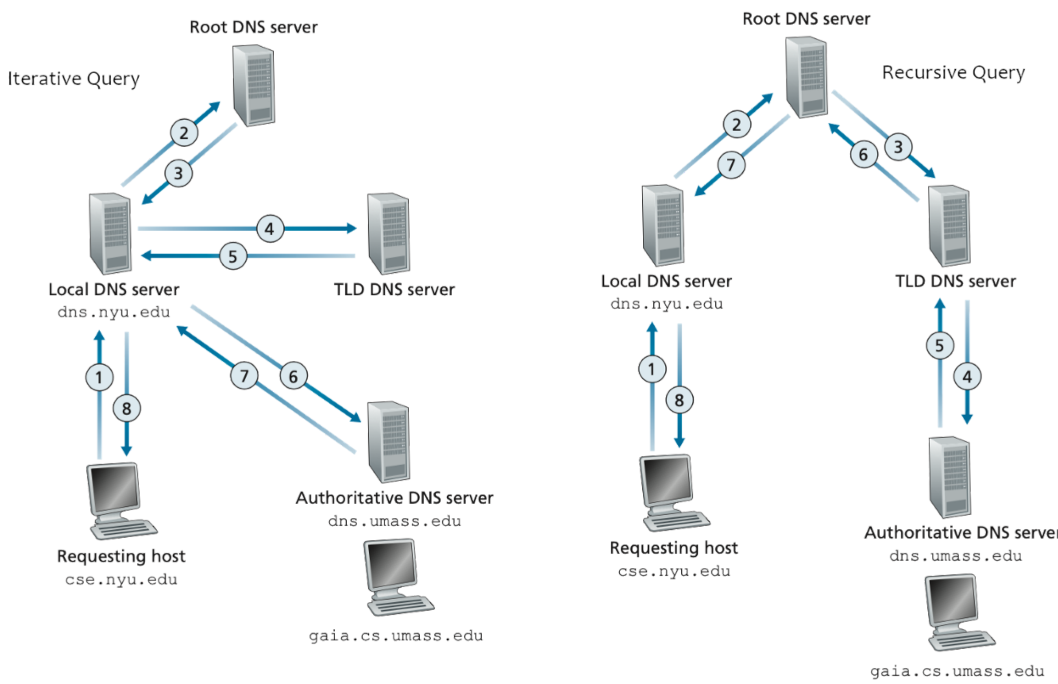
$\rightarrow$ administration for the next level down is delegated to registrars to manage complexity

1.13.3. Benefits of Hierarchy

- We build on the hierarchy of the name-space when storing DNS data, avoiding bottle-necks
- We reduce management complexity by delegating responsibility
- We allow for appropriate regional controls of this global resource



1.13.4. Side note for Recursive and Iterative DNS



When the DNS lookup is **iterative**, the resolver or Local DNS server will itself make every request to the different DNS servers in the hierarchy. As you can see above. ↑

When the DNS lookup is **recursive**, each server along the route will make the request itself to the next server in the hierarchy. As you can see above. ↑

Recursive: the local name-server does not return partial information

Iterative: The root name server only returns partial information

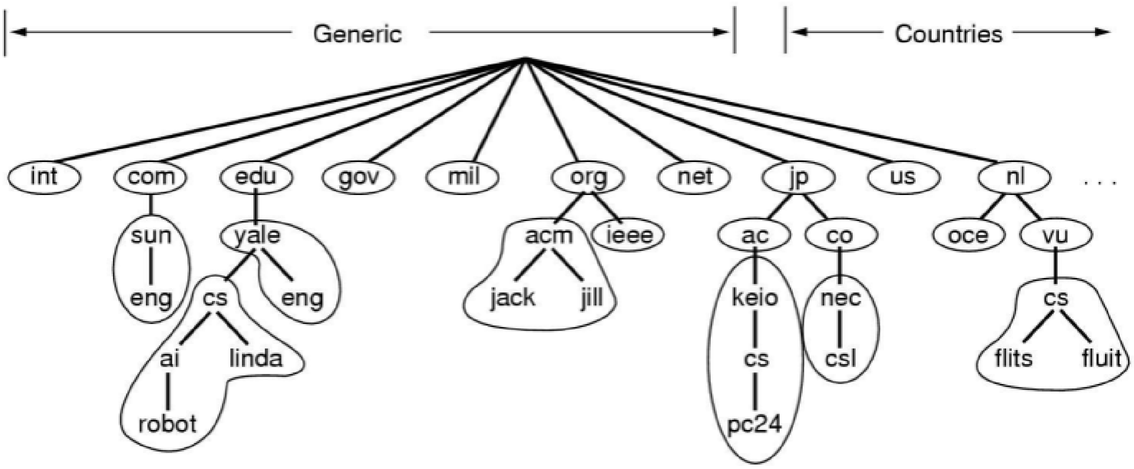
Note: NS other than the local NS may operate recursively, but it is rare due to overhead.

1.13.5. Non-Overlapping Zones

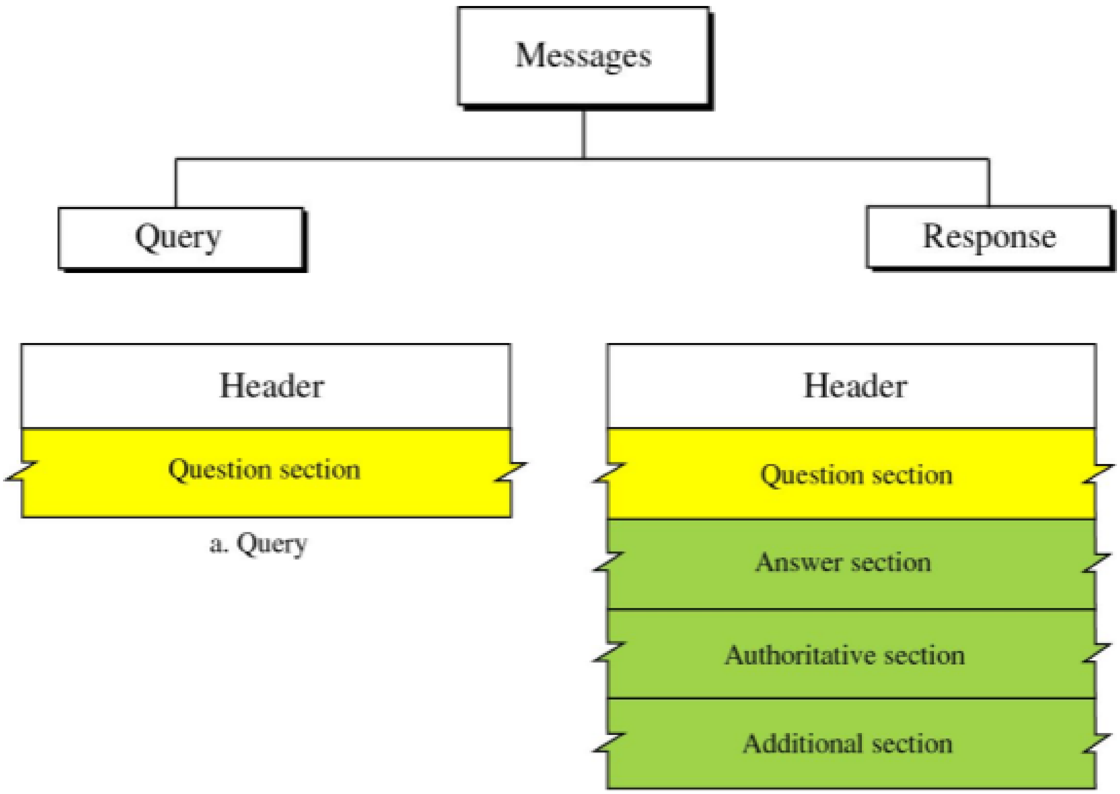
We divide the name-space into non-overlapping zones.

Each zone's administrator is responsible for drawing the boundaries of internal zones:

- Load balancing
- managing overhead of many servers
- May follow organizational boundaries



1.13.6. DNS Messages



a. Query

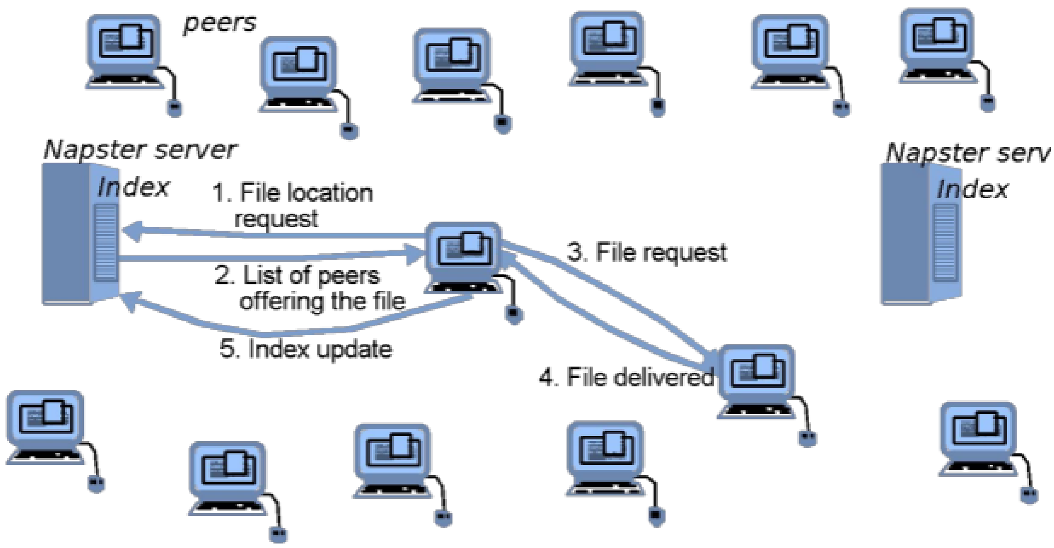
b. Response

1.14. Discovery Protocols (Napster, Gnutella)

Quick example:

1.14.1. Napster

- Was one of the earliest P2P applications
- Napster was able to implement a large-scale music download service without paying for (many) servers and high bandwidth connections.
- The infrastructure was provided largely for free by the users



Steps in **Napster**

1. Ask server for list of locations of files
2. List of peers offering the files
3. Request file from certain peer
4. File delivery
5. Index update

1.14.2. The P2P Application Pattern

Peer to peer is nice because it barely requires any servers. The incentive is that you can download files for free from peers while in exchange other peers can do the same from you.

1.14.2.1. P2P App $\neq$ P2P Network

- A P2P application re-uses edge resources to provide a service. It may, or may not use a P2P network
- Centralization limits scalability
- If you know about data, you may be responsible for it
- What happens to the services when the server fails?

1.14.2.2. Important Note

While application-level networks provide a conceptual routing substrate, each application-level hop may map to several hops at the network layer. Thus it is critical that we keep the amount of application-level hops to a minimum.

1.14.3. Gnutella 0.4

- Gnutella supports peer-to-peer resource discovery.
- Gnutella builds an **unstructured decentralised overlay network** on top of **TCP/IP**

$\rightarrow$ Gnutella does the same job as Napster without any servers:

- No single point of failure, or attack
- No need to provision powerful indexing servers.

Legally speaking, there is no single **Gnutella** entity to target so there is no one to sue.

Privacy: Limited anonymity is provided as each user only has the details of its neighbours.

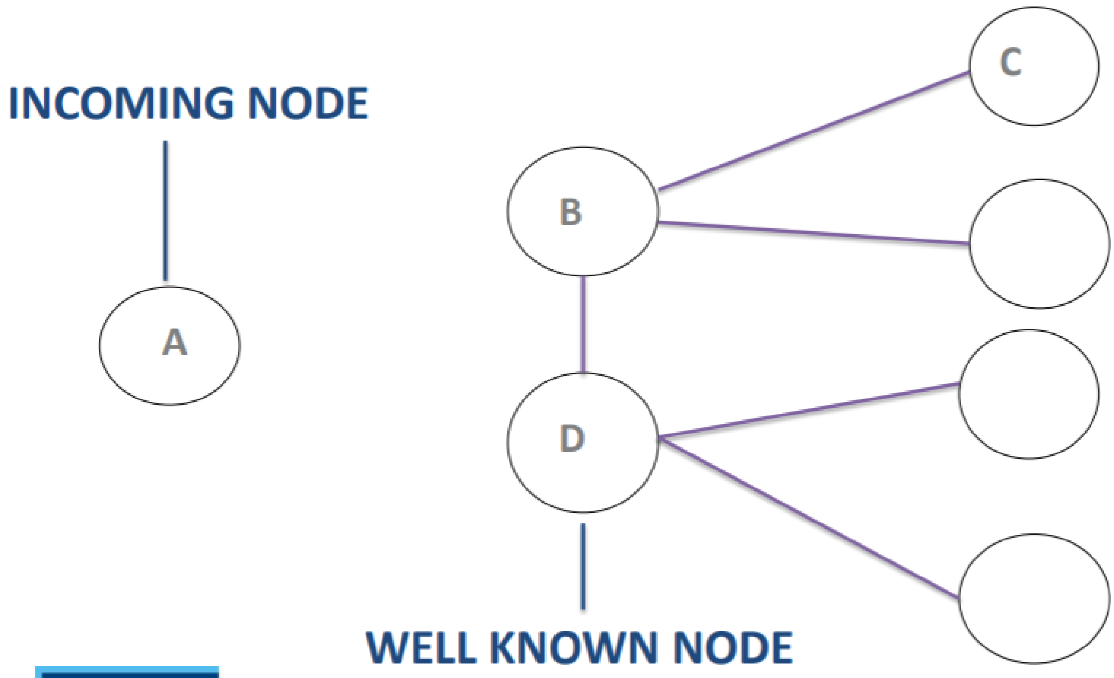
1.14.3.1. Gnutella Messages

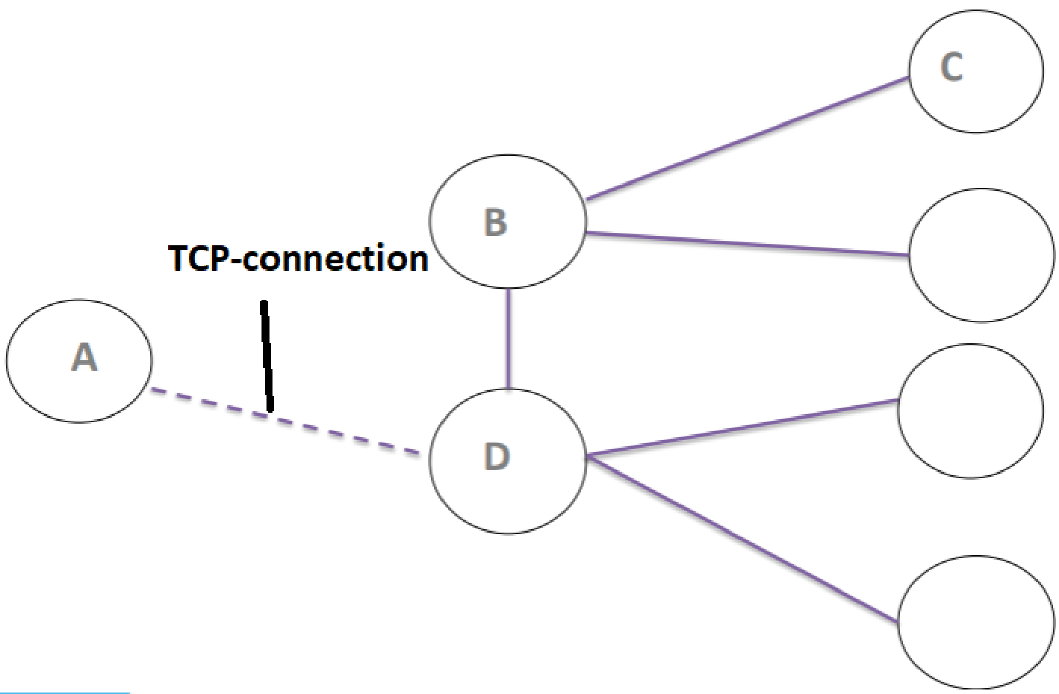
- **PING** is a broadcast message used in peer discovery. A peer that receives a PING and is capable of accepting an incoming connection should respond with a PONG message.
- **PONG** is a response to a PING. It contains the responding peer's connection details and meta-data such as the amount of data the peer is sharing.
- **QUERY** is a broadcast search message. If a peer receiving a QUERY has matching data, it generates a QUERYHIT search response message.
- **QUERYHIT** is a response to a QUERY. It contains information required to acquire the requested data and meta data (number of shared files, speed of connection etc.)
- **PUSH** is a mechanism to support downloads from fire-walled peers.

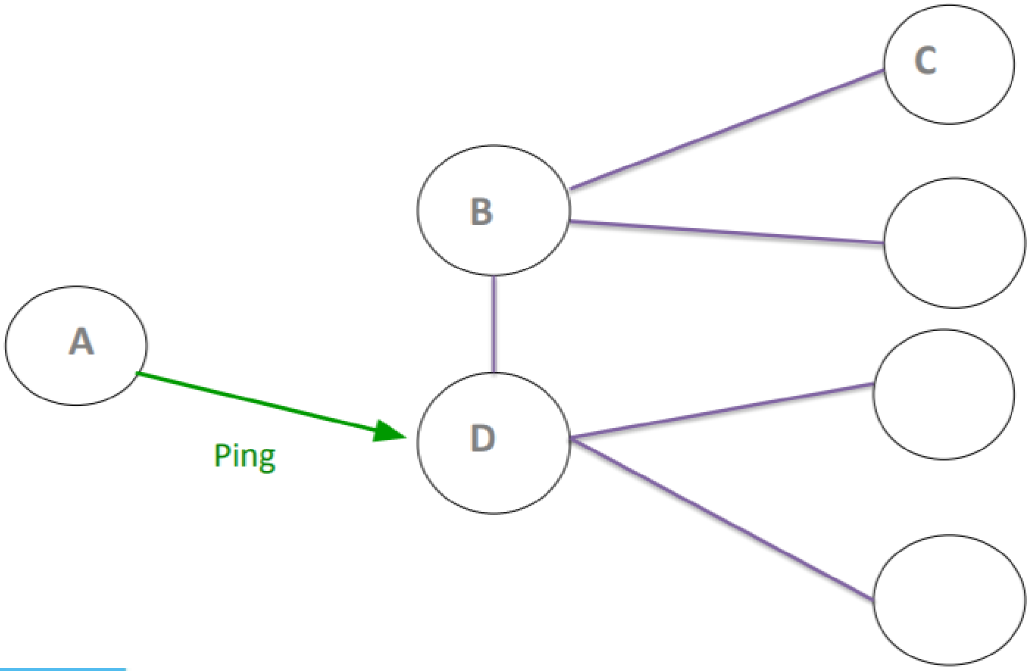
There are 3 phases to the Gnutella cycle

- Connection
- Search
- File Transfer

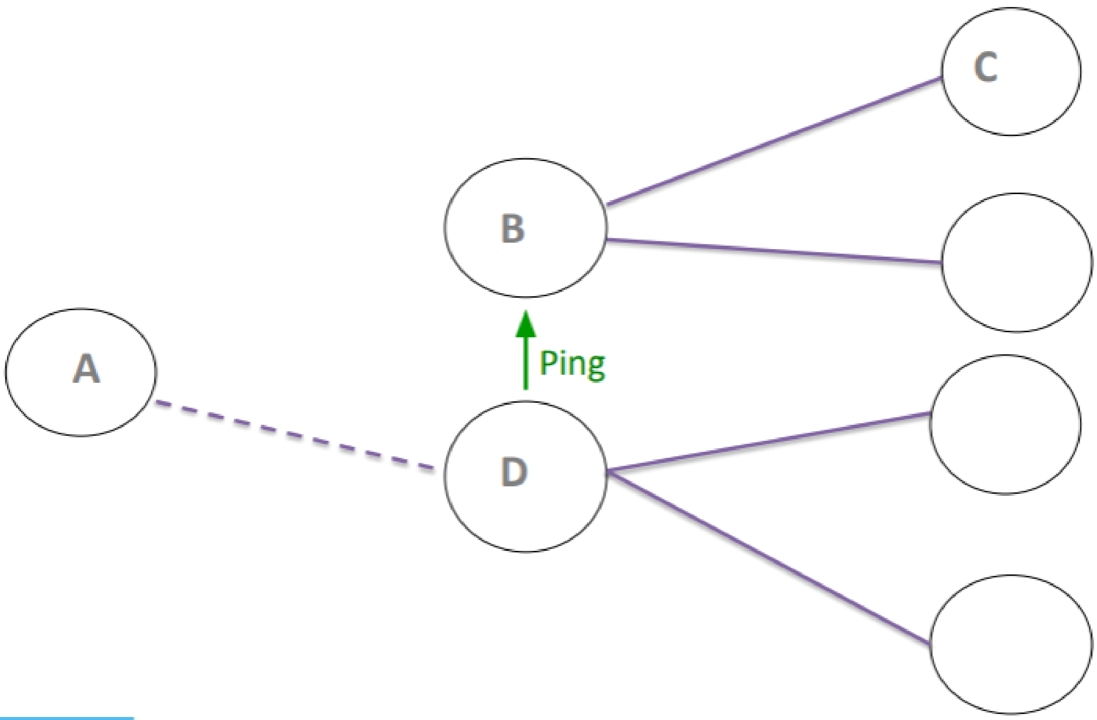
1.14.3.2. Connecting a New Peer:


INCOMING NODE
C
B
A
D
WELL KNOWN NODE

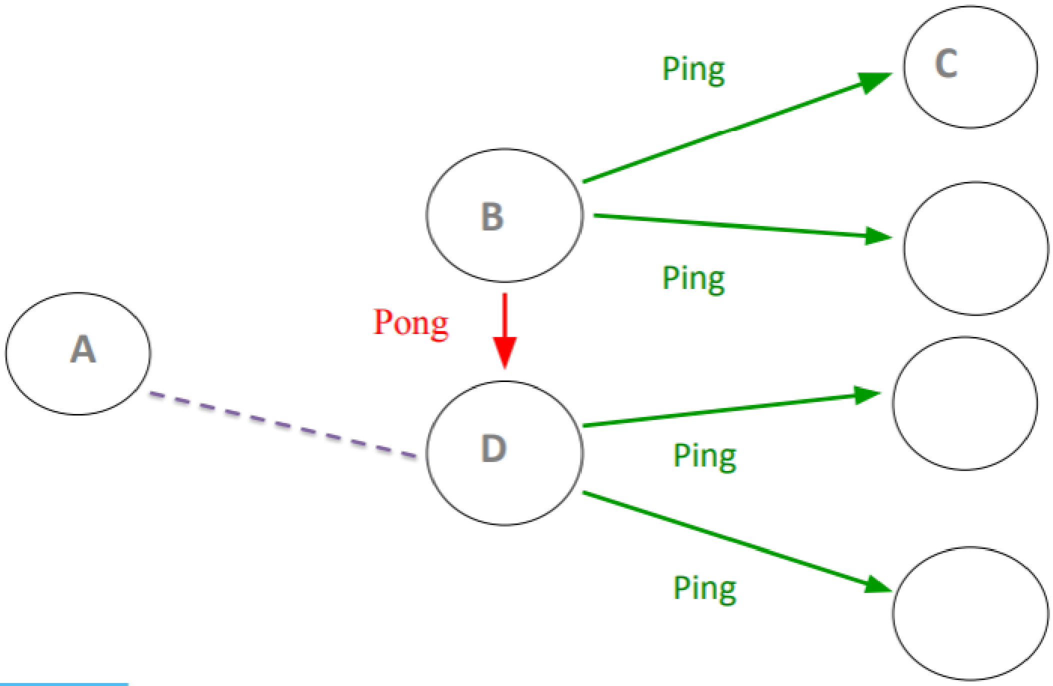

C
B
TCP-connection
A
D



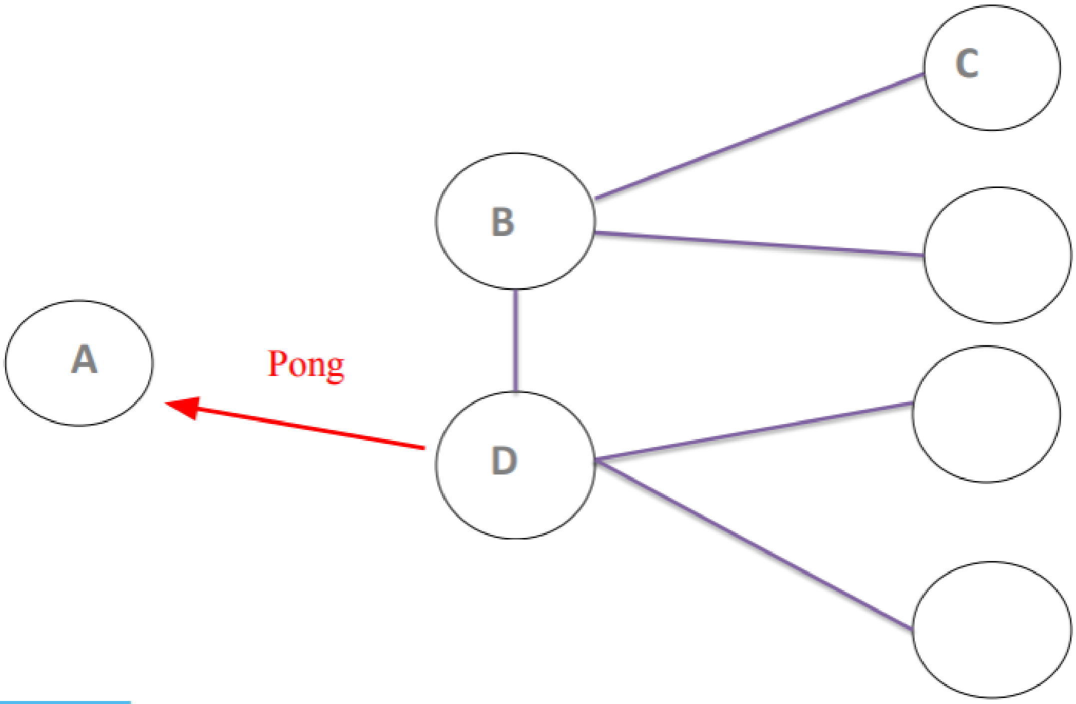
KU LEUVEN
DISTRINET RESEARCH GROUP



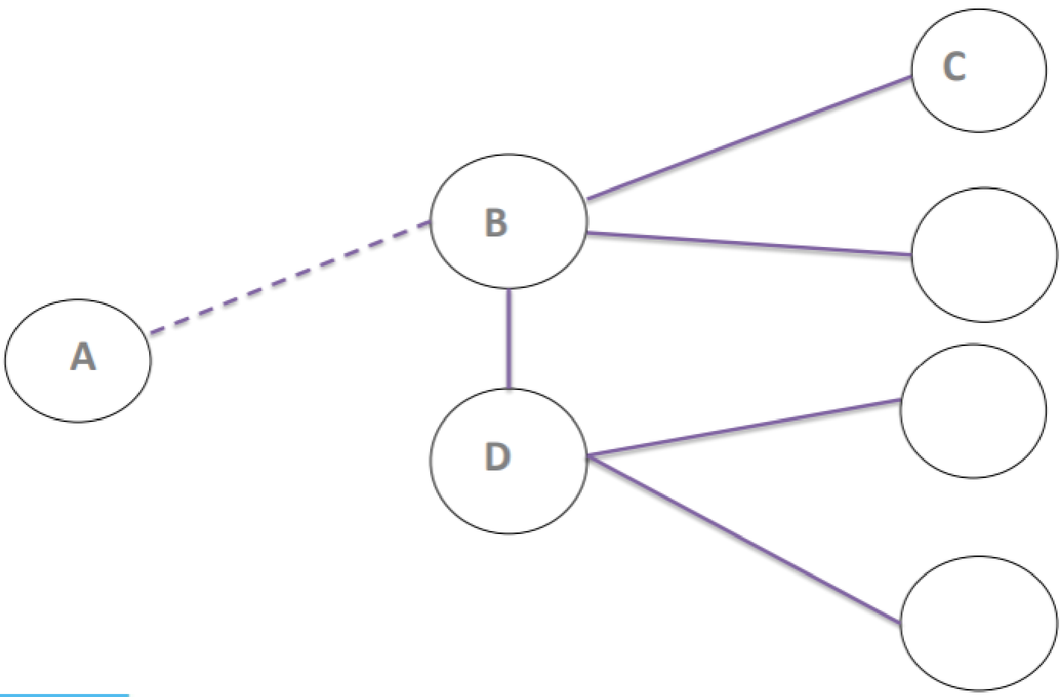
KU LEUVEN
DISTRINET RESEARCH GROUP



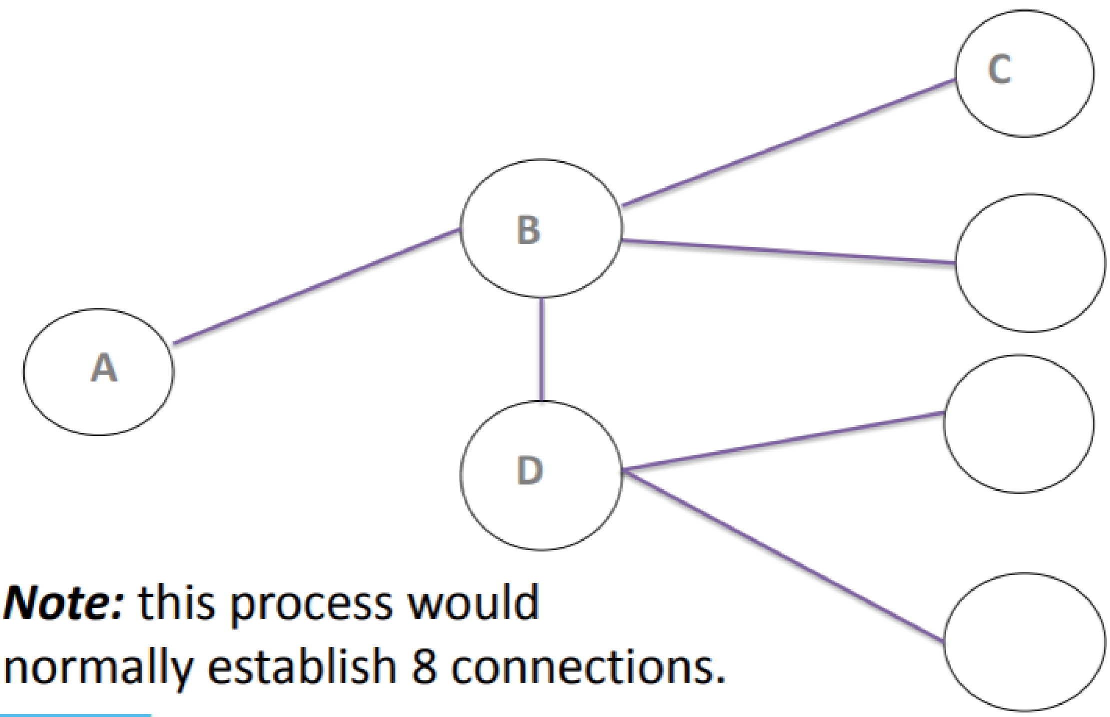
KU LEUVEN
DISTRINET RESEARCH GROUP



KU LEUVEN
DISTRINET RESEARCH GROUP



KU LEUVEN
DISTRINET RESEARCH GROUP



KU LEUVEN
DISTRINET RESEARCH GROUP

Short explanation for connecting to the Gnutella network

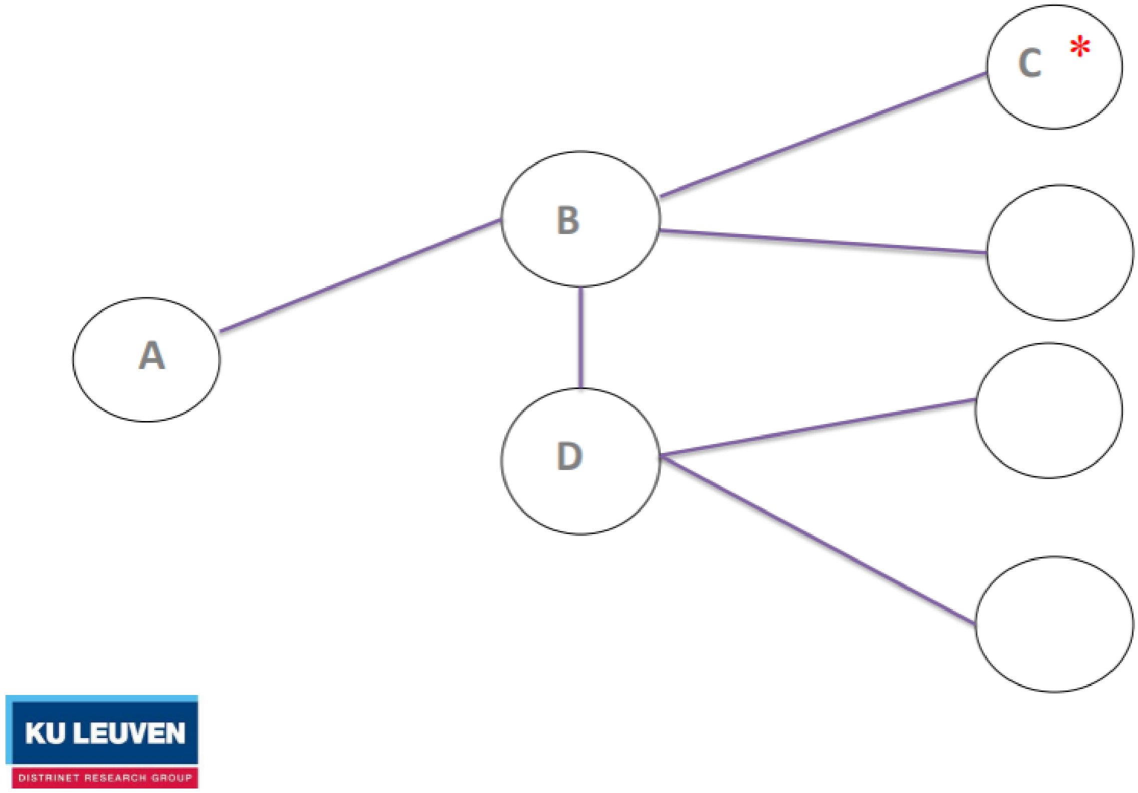
So in short, A new-arriving peer connects to an initial peer by initiating TCP connections to that host. This peer will then broadcast a PING message which 'floods the netwerk'

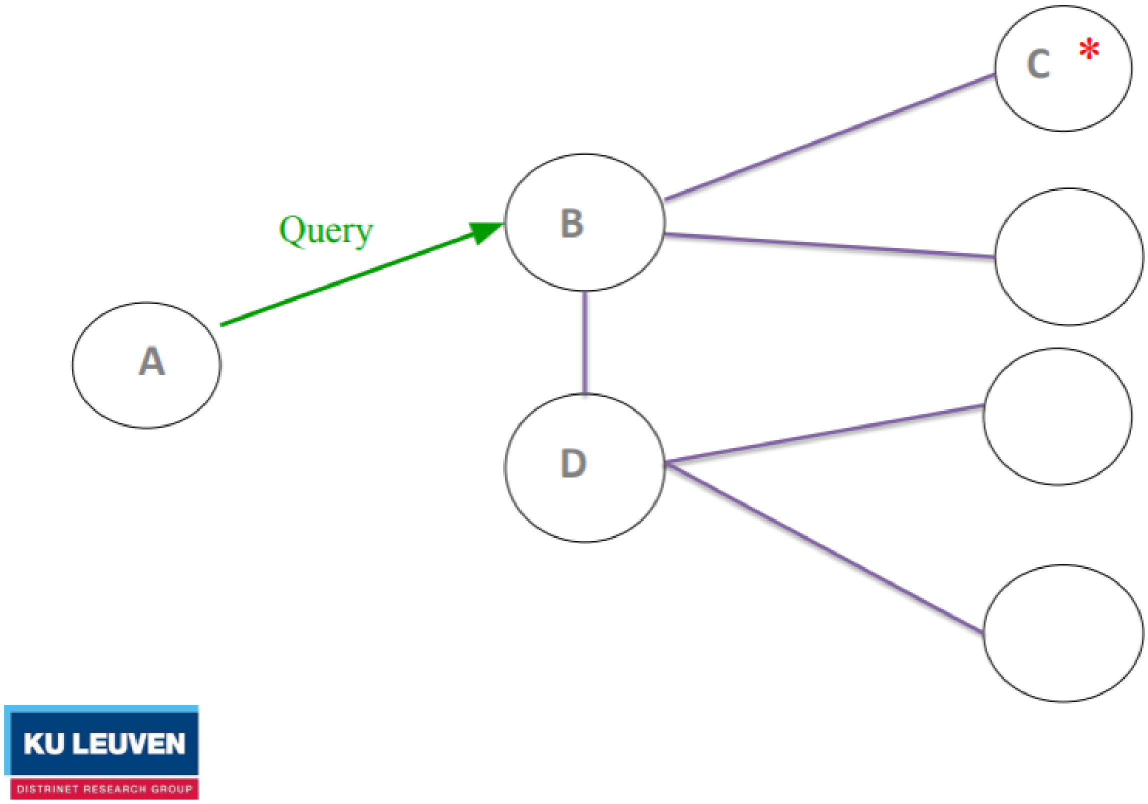
Available peers that receive a ping should respond with a PONG, which contains the network address and port on which the sending peer is listening for Gnutella connections. The Pong message is forwarded back along the path of the incoming PING

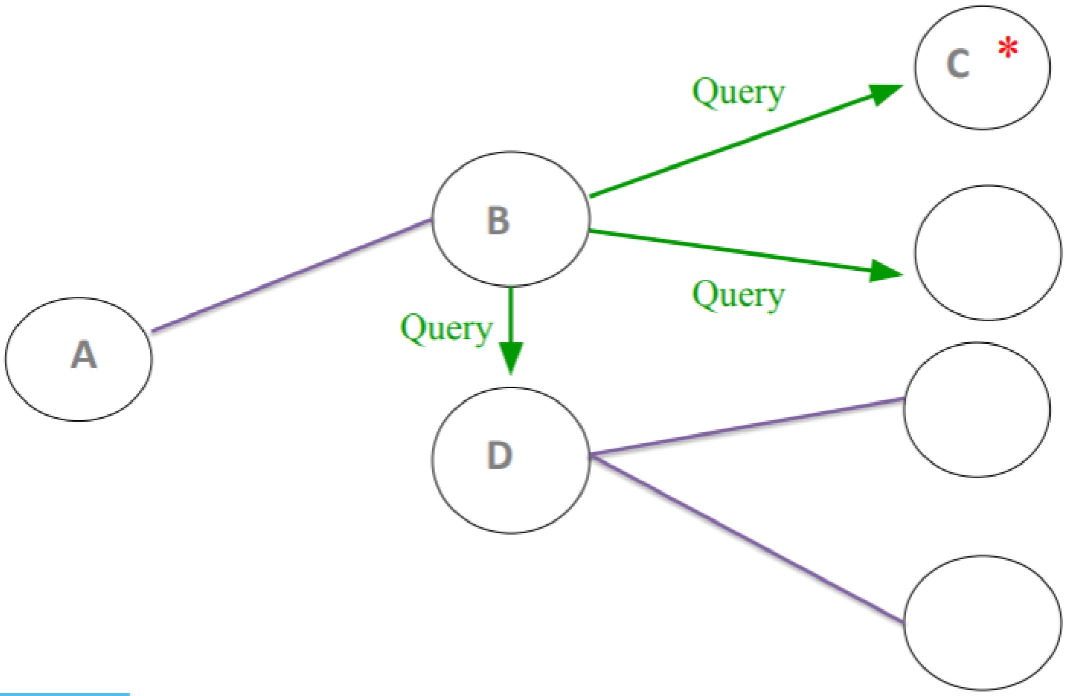
Broadcast messages are tagged with a TTL. Peers decrement the TTL, discarding where TTL = 0

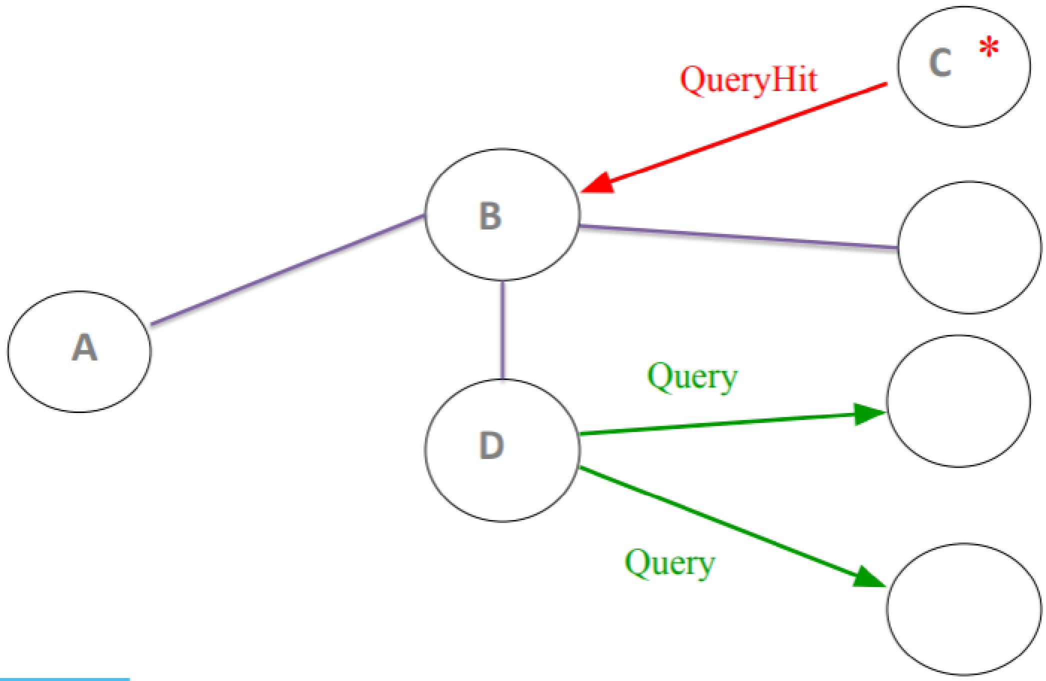
Incoming peers will establish eight connections.

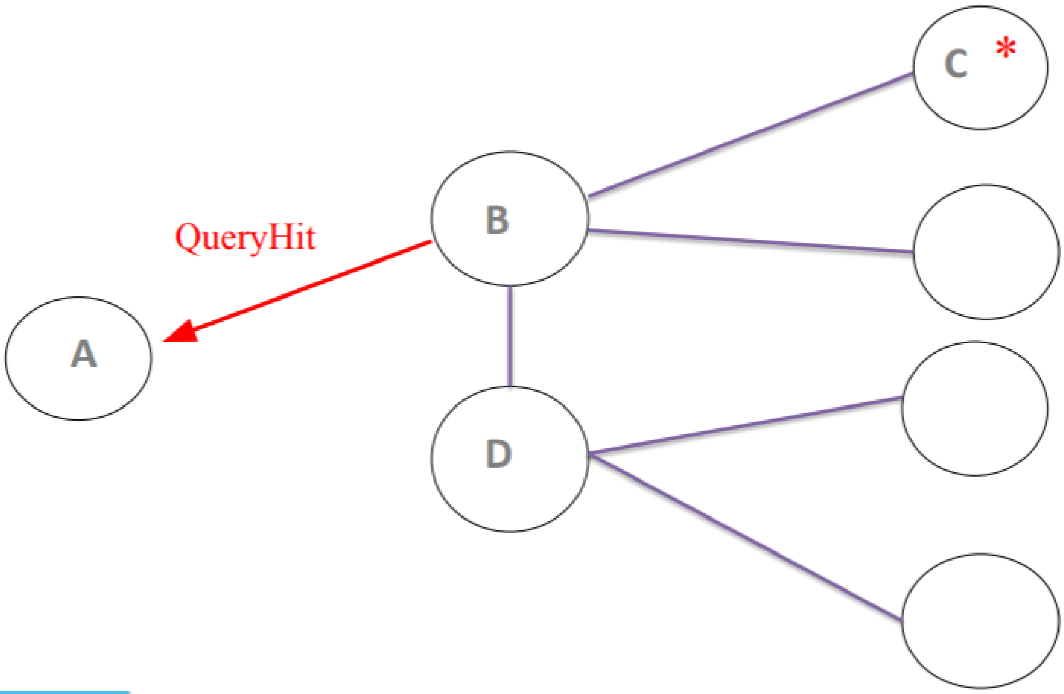
1.14.3.3. Searching for a file






C *
Query
B
Query
Query
A
D


C *
QueryHit
B
A
Query
D
Query



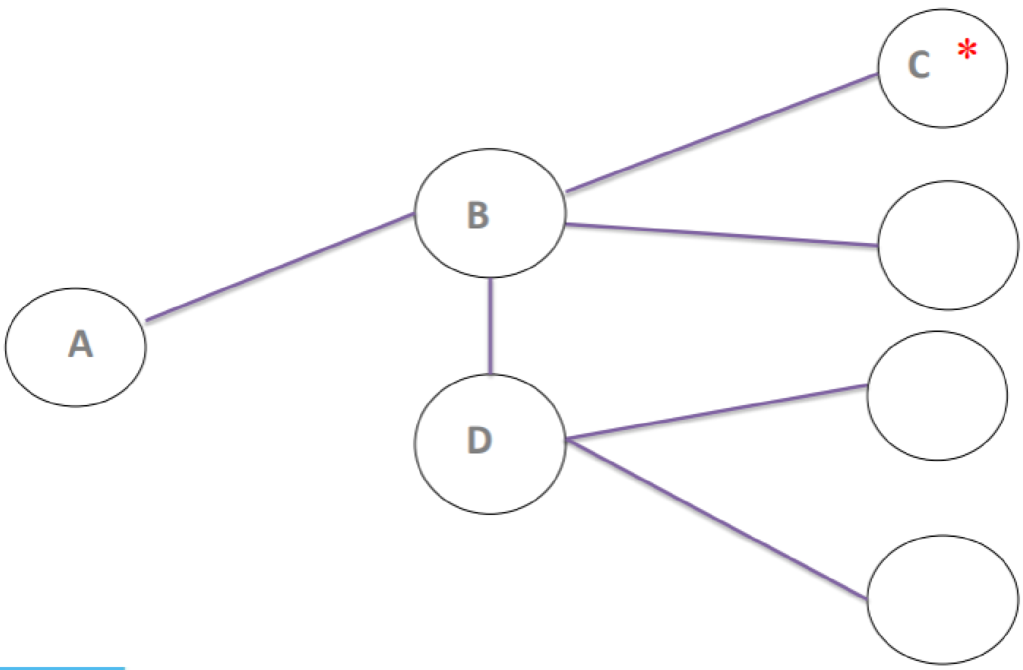
Short explanation for searching for a file in the Gnutella network

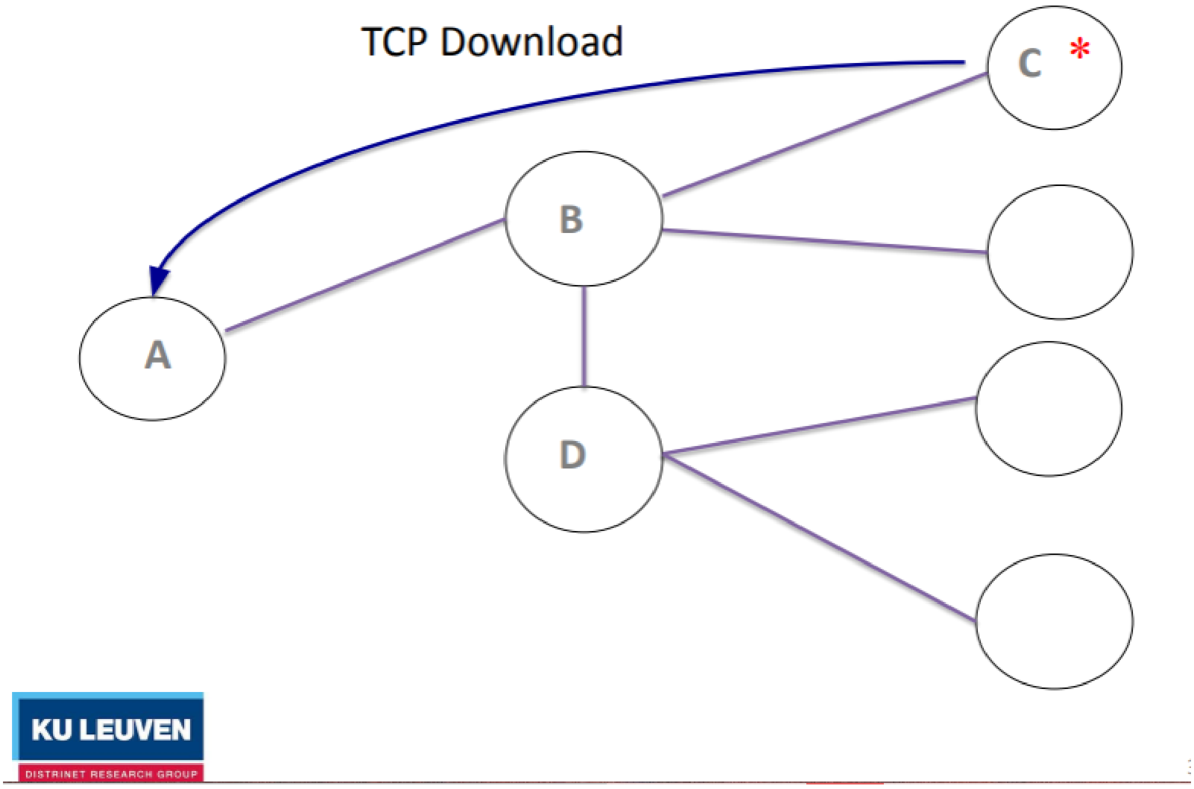
Peers listen for incoming QUERY messages, and contribute to their broadcast across the network by flooding them to each of their neighbours, while decrementing their TTL value.

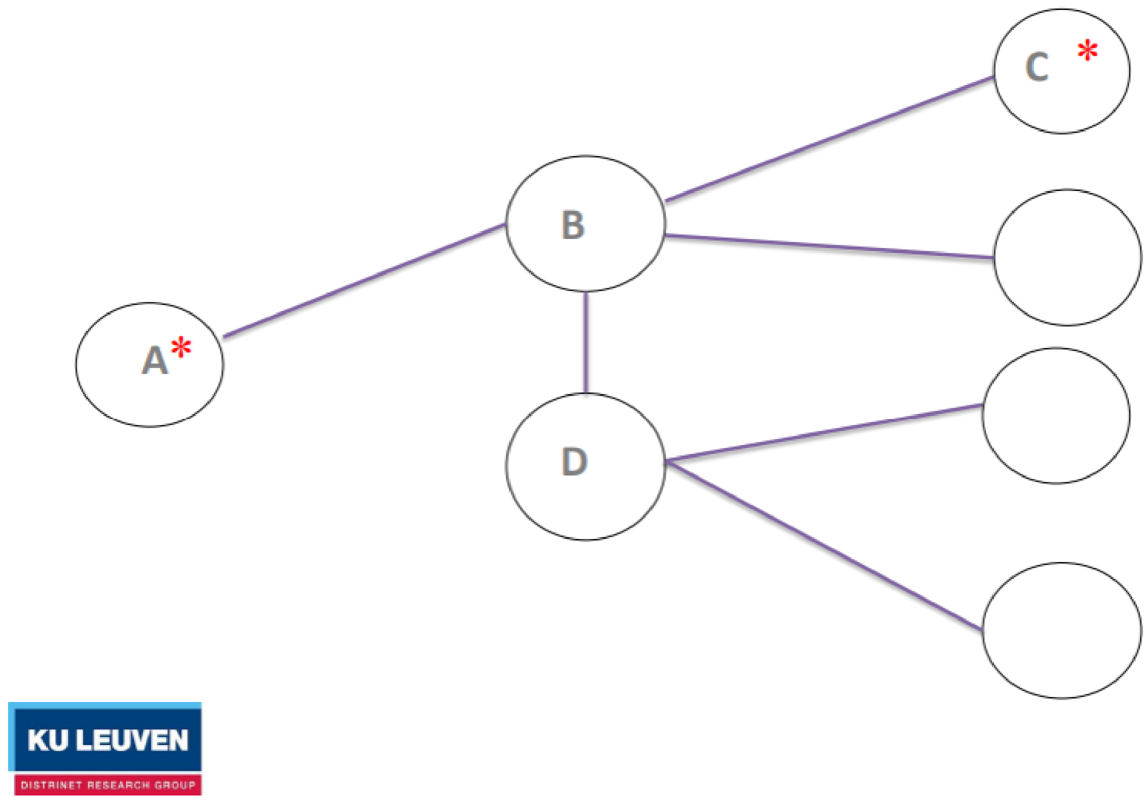
If a peer has the requested file from the QUERY, it responds by sending a QUERYHIT message back along the path of the incoming QUERY.

QUERYHIT messages contain the network address and port on which the responding peer is listening for HTTP file-transfer connections.

1.14.3.4. File Transfer from peer to peer







Short explanation for the file transfer

When requesting peer receives a QUERYHIT message, it can attempt to initiate a direct download, from the target peer (whose port and IP address were specified in the QUERYHIT message) via HTTP.

However, if the target peer is behind a firewall, the requesting peer can instead send PUSH message to the target, containing details of the file request.

On receiving a PUSH, the target peer establishes the HTTP connection and pushes the file.

1.14.3.5. Shortages of Gnutella 0.4

- The flat network structure of Gnutella results in high load due to broadcast.
- Nodes are heterogeneous. Should my phone do the same share of work as my server?

1.14.4. Gnutella 0.6

1.14.4.1. Upgrade of Gnutella 0.6

- Only ultra-peers participate in peer discovery. Leaf-nodes always connect to an ultra-peer
- When a leaf node connects to an ultra-peer it uploads a complete list of its resources
- File discovery messages are only sent to leaf-nodes where they host a matching file.

$\rightarrow$ This way **leaf nodes** do not participate in discovery, reducing their load.

1.14.4.2. Requirements for Ultrapeers:

- No firewall
- Sufficient Bandwidth
- Sufficient Uptime
- Sufficient RAM and CPU

Gnutella 0.6 improves on the scalability of Gnutella 0.4

- By exploiting the resources available on strong nodes
- conserving the resources available on weak nodes

1.15. File Transfer Protocols (Bittorrent)

BitTorrent is thet most popular P2P network today, accouting for up to 35% of Internet traffic

- Gnutella is designed for resource discovery
- BitTorrent is designed for **efficient content distribution**

1.15.1. The Complete File Sharing Problem

1.15.1.1. Q1: How to discover the location of files to download

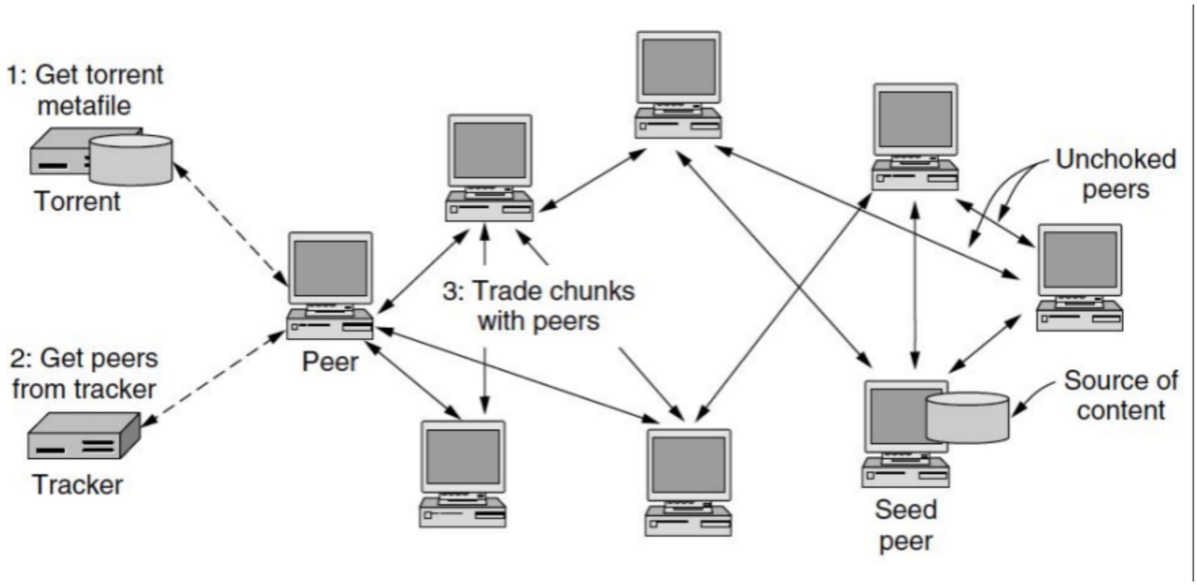
A1: Each torrent provides connection details of 'trackers' that know addresses of peers downloading this file: **the swarm**. Each file is broken into many small chunks. Each chunk is identified by an SHA-1 hash key. On connection, addresses of the swarm are transferred from the tracker

1.15.1.2. Q2: How to optimally replicate content to peers

A2: Peers should both download and upload chunks. Peers with all chunks are seeders. Peers share lists of chunks and download the rarest first, optimizing availability.

1.15.1.3. Q3: How to encourage users to upload

BitTorrent tackles the problem of Free-riding by implementing a 'tit-for-tat' trading scheme. Peers measure download performance. Trading continues only with peers offering high download speed. Over time this matches peers with similar speeds. Free-riders are cute off, or "choked"



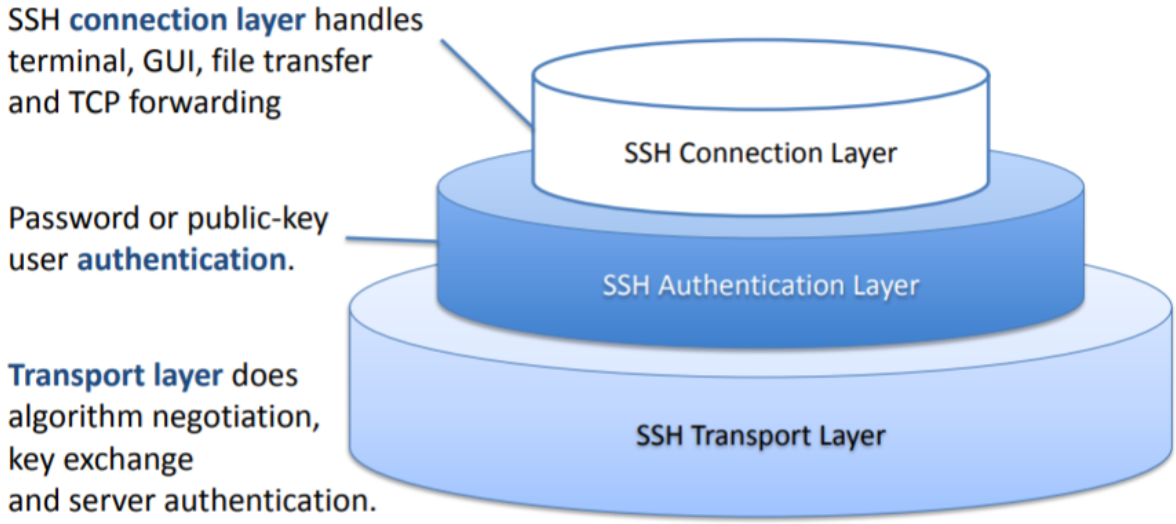
1.16. Secure Shell (SSH)

SSH solves two key problems:

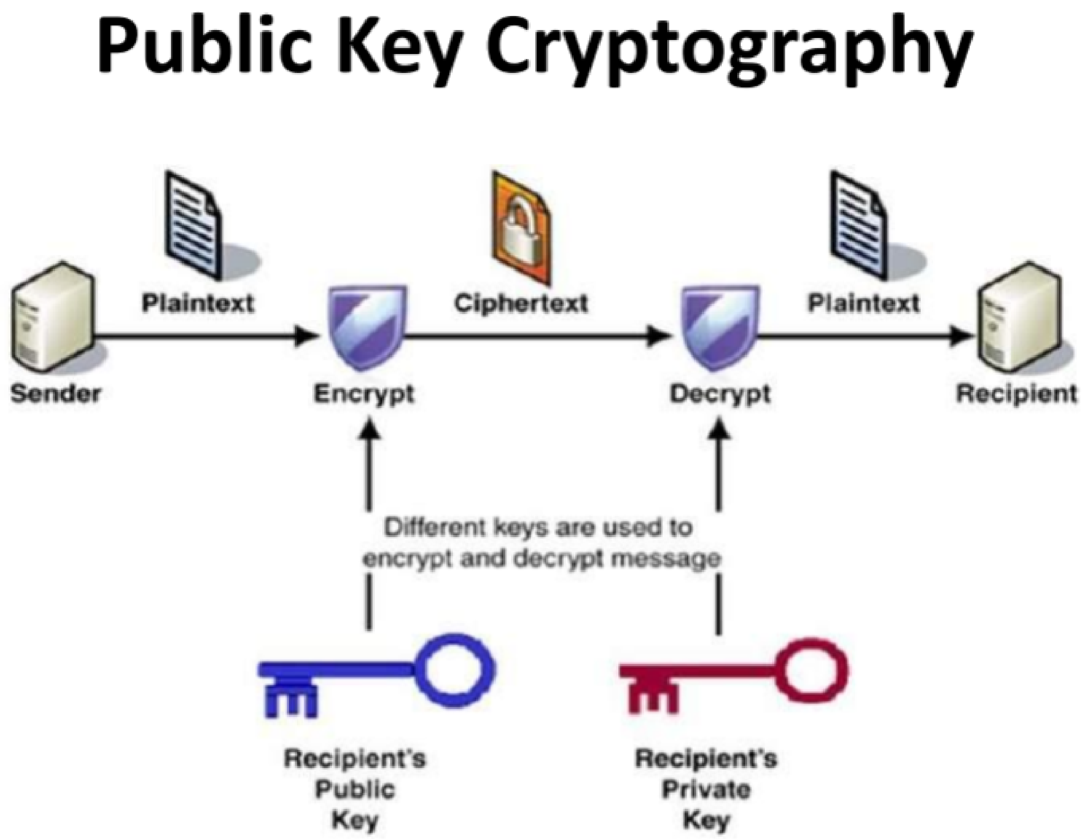
- Telnet does not offer server authentication
- transmits the user credentials in the open

1.16.1. SSH Connection

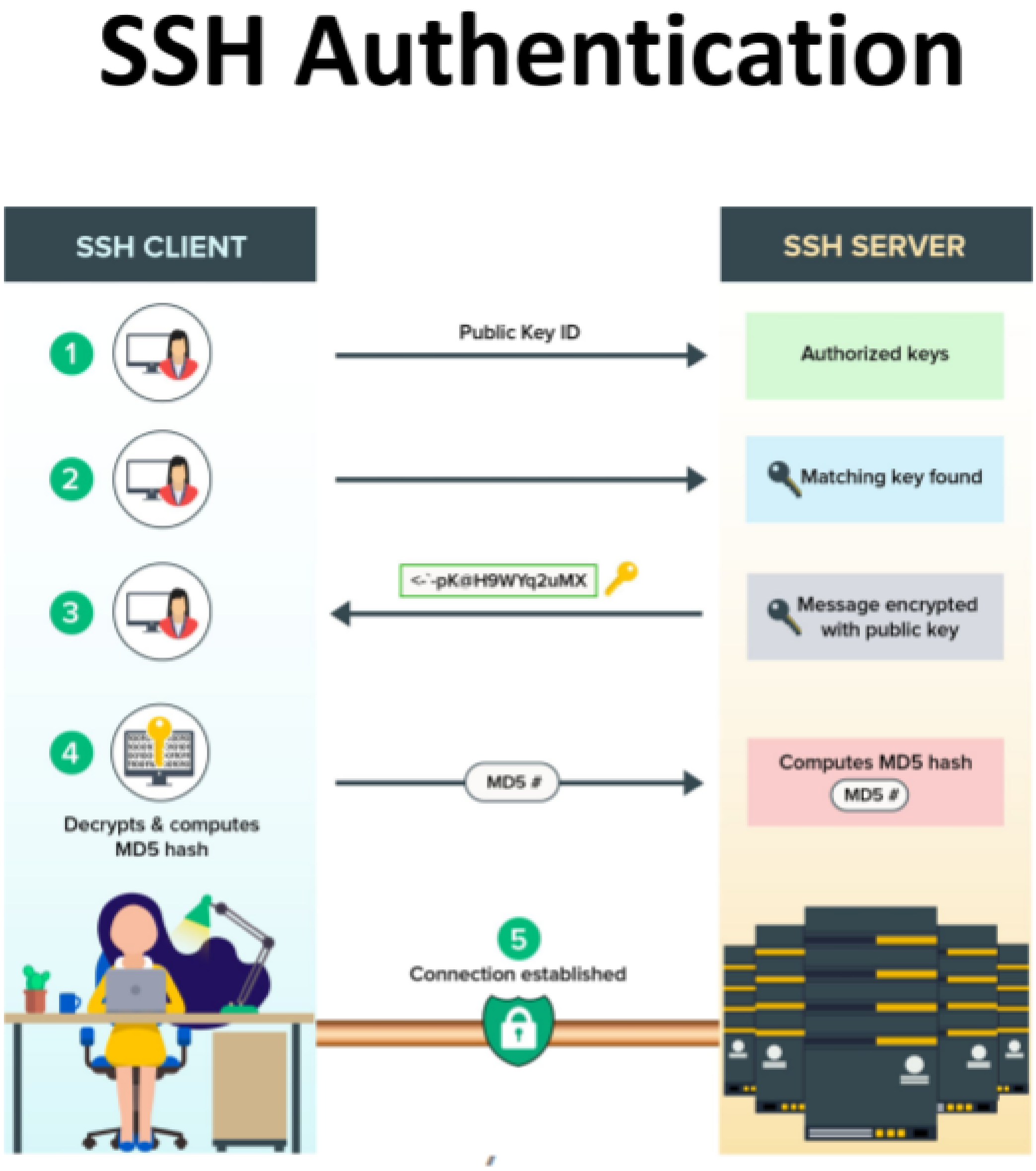
1. Open up a TCP connection between the two machines (doesn't need to be TCP but needs to be reliable)
2. Packet is made with the packetlength, paddingamount, payload, padding and message authentication code(To see the packet hasn't been changed).
3. Packet is encrypted except for the packetlength and the message authentication code.



1.16.2. Public Key Cryptography



1.16.3. SSH Authentication



1.16.4. Anonymity

- Anonymity in P2P protects the user from censorship
- On the other hand, it provides a cover for illicit behaviour

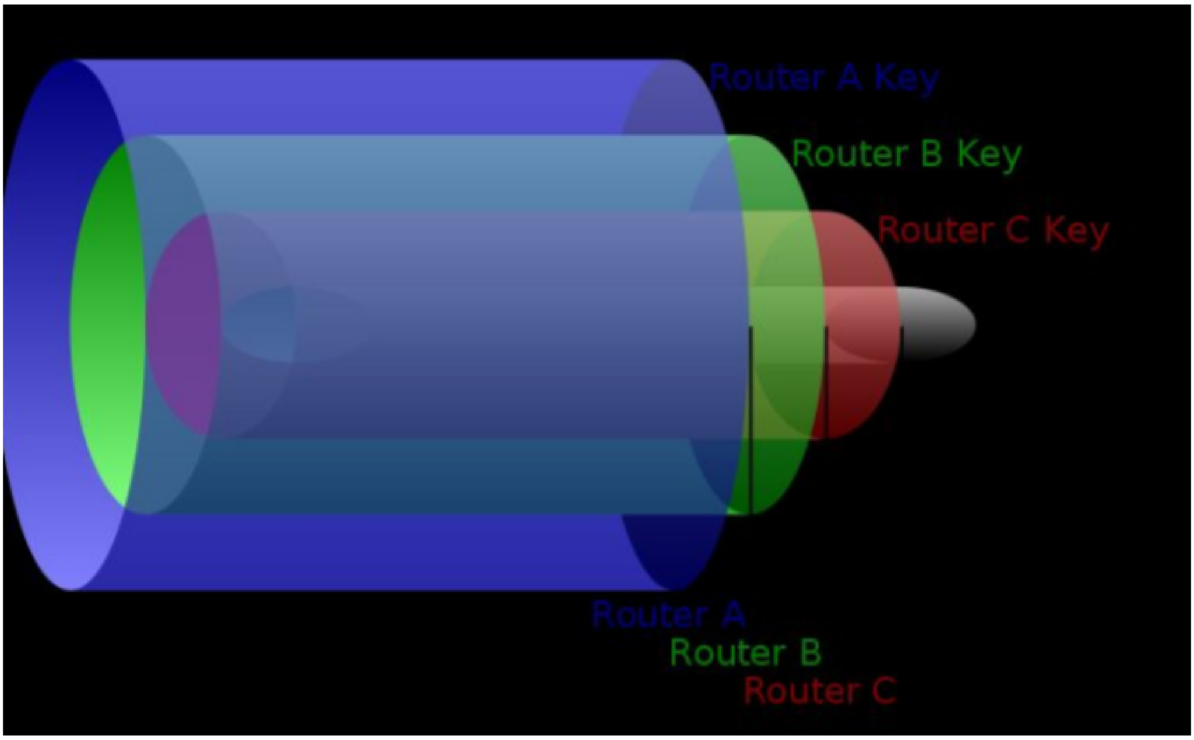
⟶ Let us look at an anonymous P2P system

1.17. Onion Routing Technique

In onion routing, messages are encrypted as they travel along each point in a circuit of nodes called onion routers

Each router removes one layer of encryption to reveal next-hop routing instructions and routes the message along its way.

Intermediate nodes do not know the origin, destination or content; only the next hop. Like an onion we have **many layers of privacy**



1.18. TOR: General Approach

TOR client obtains a lot of TOR routers from the directory server

TOR client picks a random path through the TOR network, terminating at an exit node.

- All links are encrypted except for the outgoing link from the exit node.

When the client downloads a different file, they create a whole new path.

1.18.1. Security

- Cannot prevent a **Sybil attack**, but is designed to make this difficult.

A **Sybil attack** is one where an attacker pretends to be so many people at the same time. It is one of the biggest issues when connecting to a P2P network. It manipulates the network and controls the whole network by creating multiple fake identities. To a single view, these different identities look like regular users, but behind the scenes, a single entity is called an unknown attacker who controls all these fake entities at once.

- An adversary can:
    - Generate, modify, delay and delete traffic
    - Operate onion routers
    - Compromise many onion routers
- TOR protects against traffic analysis attacks and thus is resistant to monitoring and censorship

1.18.2. TOR Architecture

- All connections between nodes are TLS secured
- Each client node runs **Onion Proxy(OP)**:
  - Offers a **SOCKS** interface to applications
  - Discover routers by querying servers
  - Establishes circuits on the overlay
- A client != an **Onion Router(OR)**

1.18.3. Onion Routers

Each router maintains a set of keys that are used to sign descriptors of the router for the directory

Onion routers may act as an exit point, or a forwarding point along a circuit (as dictated by the Onion Proxy client)

TOR security is based on public key cryptography

1.18.4. TOR Cells

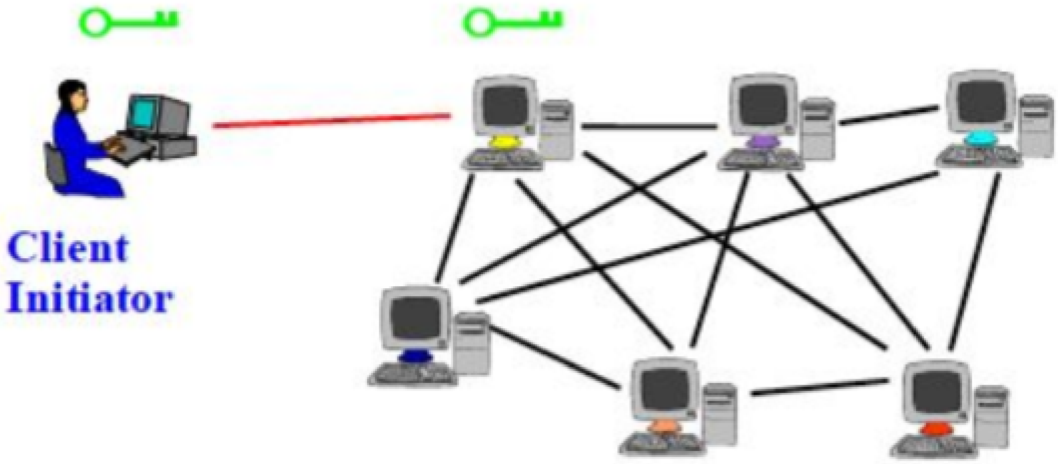
- Traffic is encapsulated in fixed-size cells, with header containing:
  - Circuit ID header (circID) to determine which circuit the cell refers to.
  - Descriptor of cell's payload type
- Control cells are interpreted by the receiving node
- Relay cells carry end-to-end data and an 2nd header containing necessary meta-data

1.18.5. Important Commands

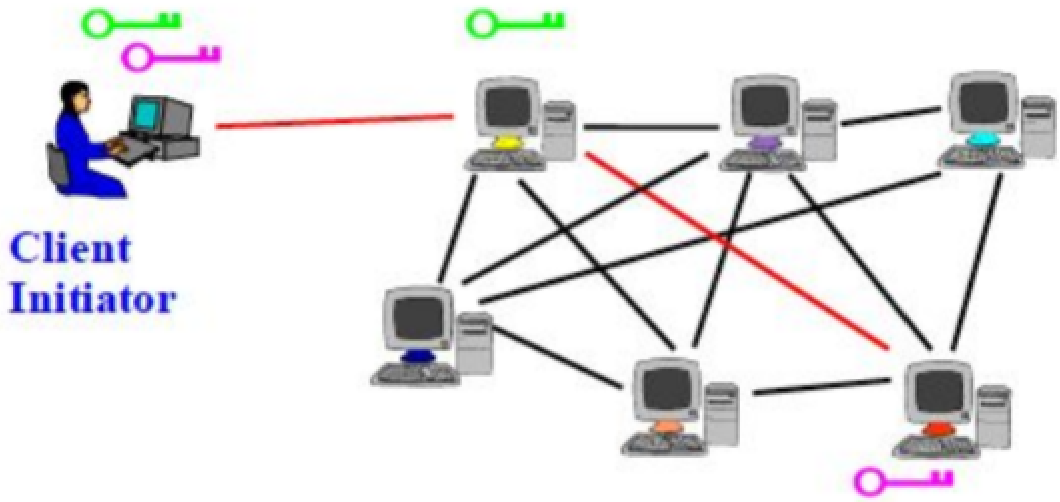
- *Relay begin*: to open a new stream.
- *Relay end*: to close a stream cleanly.
- *Relay connected*: opening notification.
- *Relay extend*: extend circuit by a hop.
- *Relay extended*: extension notification.

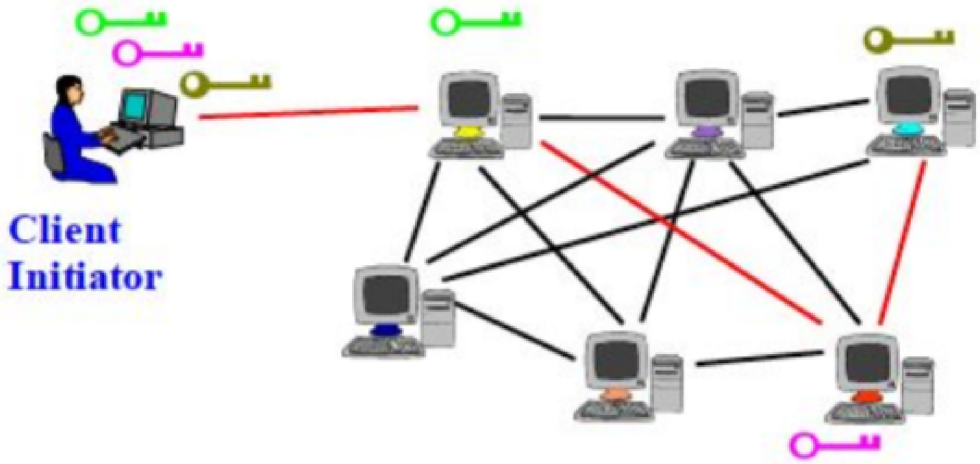
1.18.6. TOR Circuits

1. We begin by establishing a connection to known OR using 'relay begin'

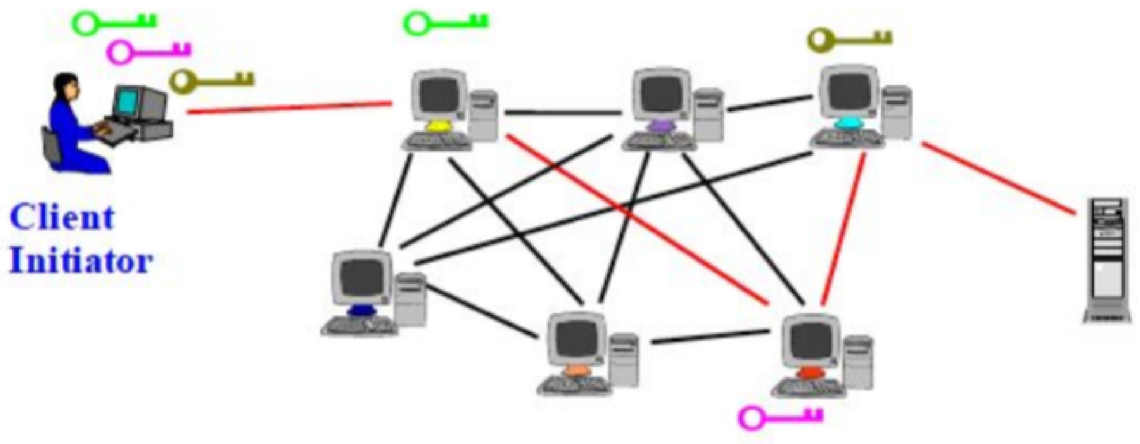


2. We then extend as necessary with relay extend





3. Once established, we can tunnel our traffic to the exit router



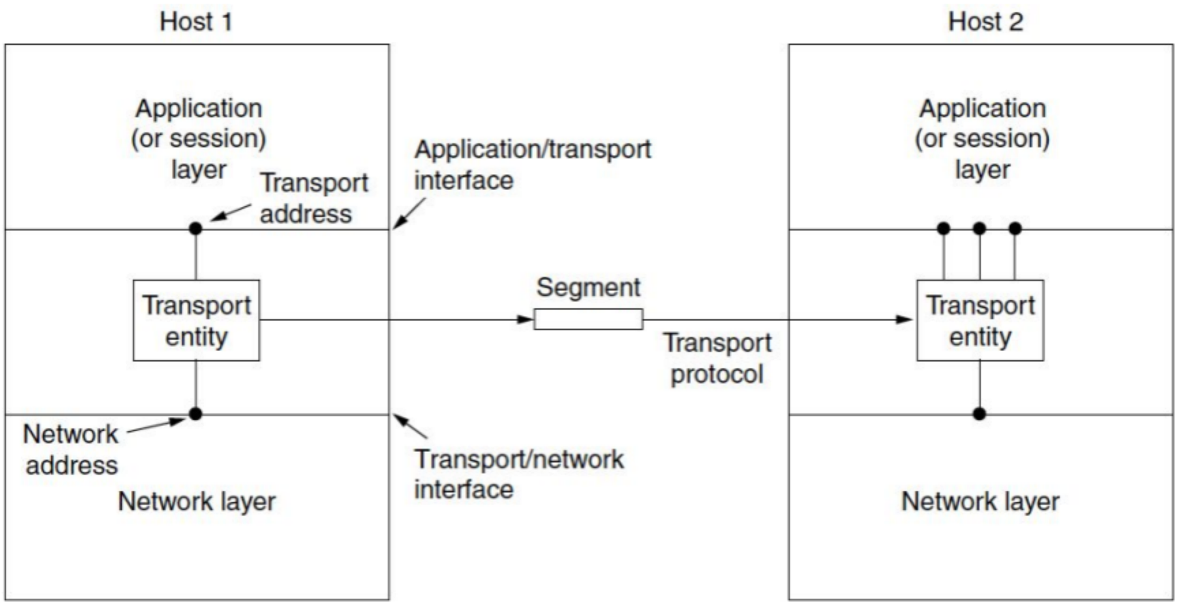
1.18.7. Establishing a TCP Connection

1. App. obtains TCP connection via SOCKS proxy.
2. The OP builds/chooses a circuit & exit node OR
3. The OP then sends a relay begin cell to host, the host responds with relay connected cell.
4. The OP accepts data from the TCP stream, which is packaged into relay data cells and passed to the exit OR.

2. Chapter 2 The Transport Layer

2.1. The Transport Layer

- Provides end-to-end delivery of data encapsulated in segments using virtual circuits
- Strives for efficiency, relatiability and cost effectiveness
- Transport layer is implemented by libraries running on hosts



2.1.1. Connectionless Transport

Connectionless transport just wants to be fast and simple, no extra stuff and that shows in the limited error control and flow control.

2.1.2. Connection-oriented Transport

- Focused upon providing reliable communication. Handles lower-layer errors
- Minimizes complexity and ensures a good separation of concerns
- Three phases of interaction

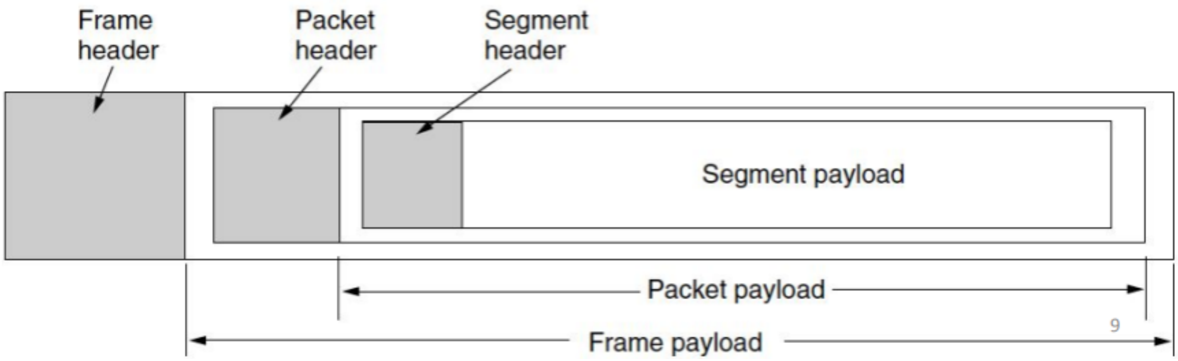
Connection-oriented transport is different because it focuses on providing reliable communication and does handle errors. Also tries to minimize complexity

Three phases of interaction:

- **Establishment** (connection setup)
- **Data Transfer** (application interaction)
- **Release** (connection tear-down)

2.1.3. Segment Encapsulation

- units of transmission at the transport layer => segments
- **Segments** at the transport layer are contained in **packets** at the network layer, which are in turn contained in **frames** at the data link layer



2.1.4. Generalized Transport Primitives

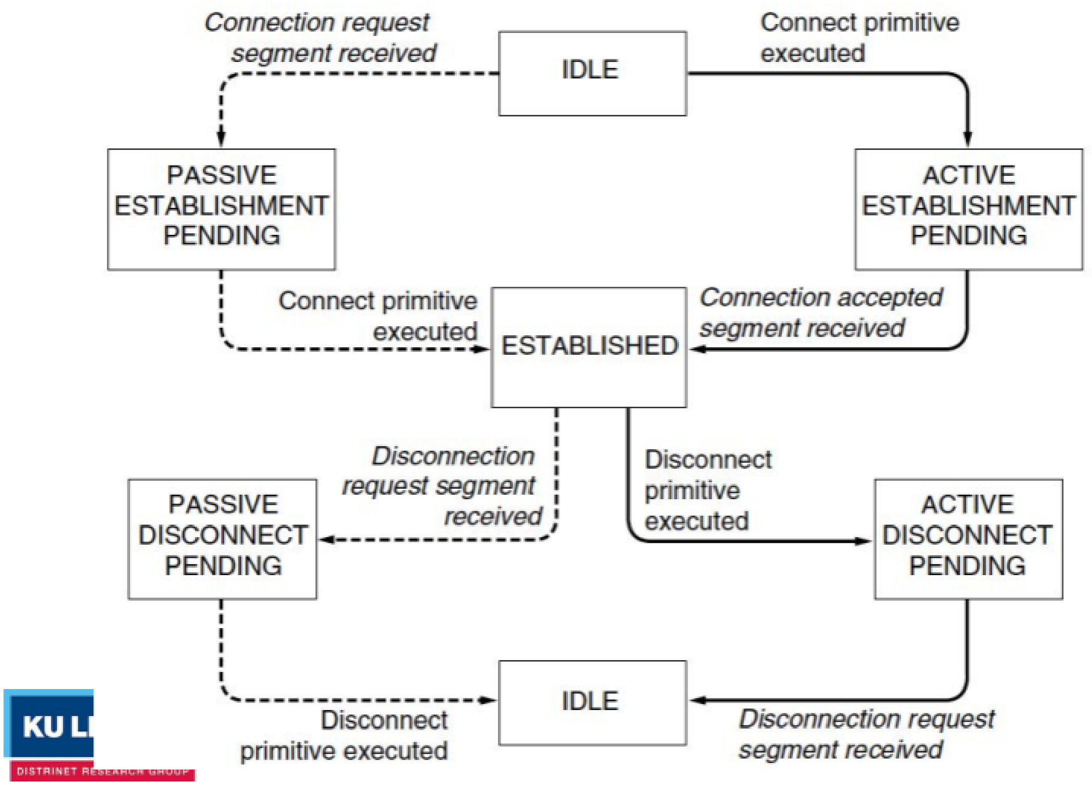
Primitive	Packet sent	Meaning
LISTEN	(none)	Block until some process tries to connect
CONNECT	CONNECTION REQ.	Actively attempt to establish a connection
SEND	DATA	Send information
RECEIVE	(none)	Block until a DATA packet arrives
DISCONNECT	DISCONNECTION REQ.	Request a release of the connection

STATE = ESTABLISHED

2.2. Berkeley Sockets (BS)

Just a normal socket uwu

2.2.1. BS State Machine

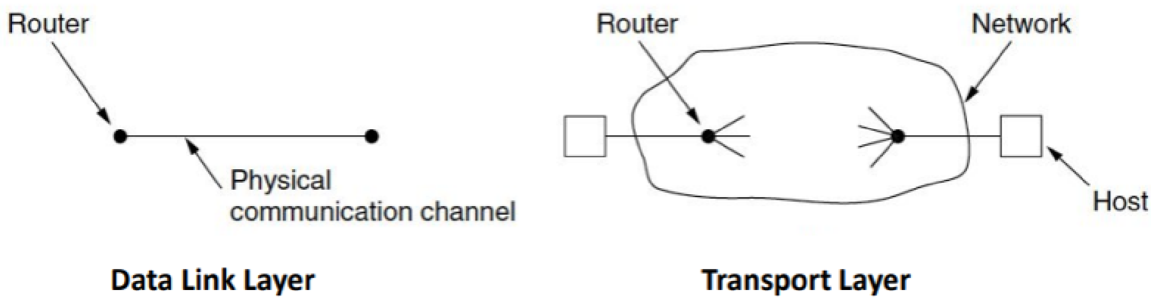


2.3. Compare to the Data Link Layer

The Transport Layer provides many similar services to the Data Link Layer such as error control, flow control and sequencing

The key difference is:

- The Data Link Layer provides communication between two hosts on the same physical link, while...
- Hosts at the Transport Layer may be separated by a whole network



2.4. Elements of Transport Layer Protocols

We will look at four critical elements of transport layer protocols

1. Addressing
2. Connection Management
3. Error Control
4. Flow Control

2.4.1. Addressing

2.4.1.1. Transport Service Access Points i.e PORTS

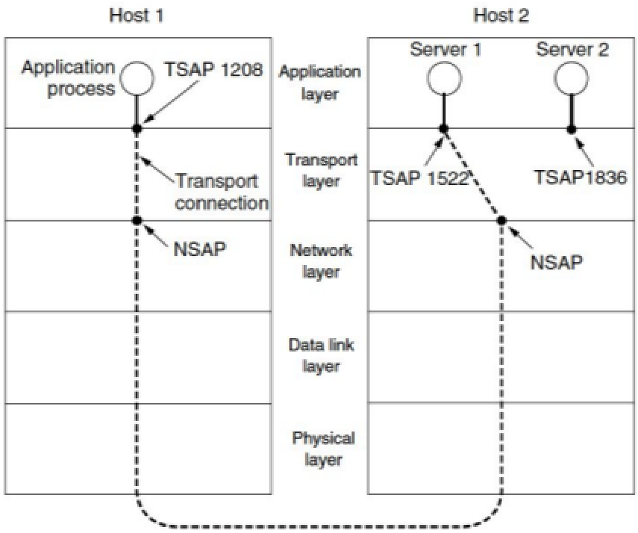
- Transport Service Access Points (TSAPs) define an end-point for transport layer traffic.
- IP addresses are not enough because multiple processes running on the same host may concurrently exchange data.
- Ports are an example of a TSAP

2.4.1.2. Network Service Access Points i.e IP-addresses

2.4.1.3. Network Service Access Points (NSAPs) define an end-point for network layer traffic

- IP addresses are an example of NSAP
- Transport Service Access Points (TSAPs) define an end-point for transport layer traffic.

2.4.1.4.



- **Note:** one NSAP serves multiple TSAPs. Analogy is telephone switchboard with 1 external number.

The explanations for this is quite simple. Across our network we identify routers and clients by their IP-address but once we reach the destination of the IP we are still left wondering for which application the message is. Then we use the port to identify where the message has to go next as can be seen on the image.

2.4.1.5. TSAP Address Assignment

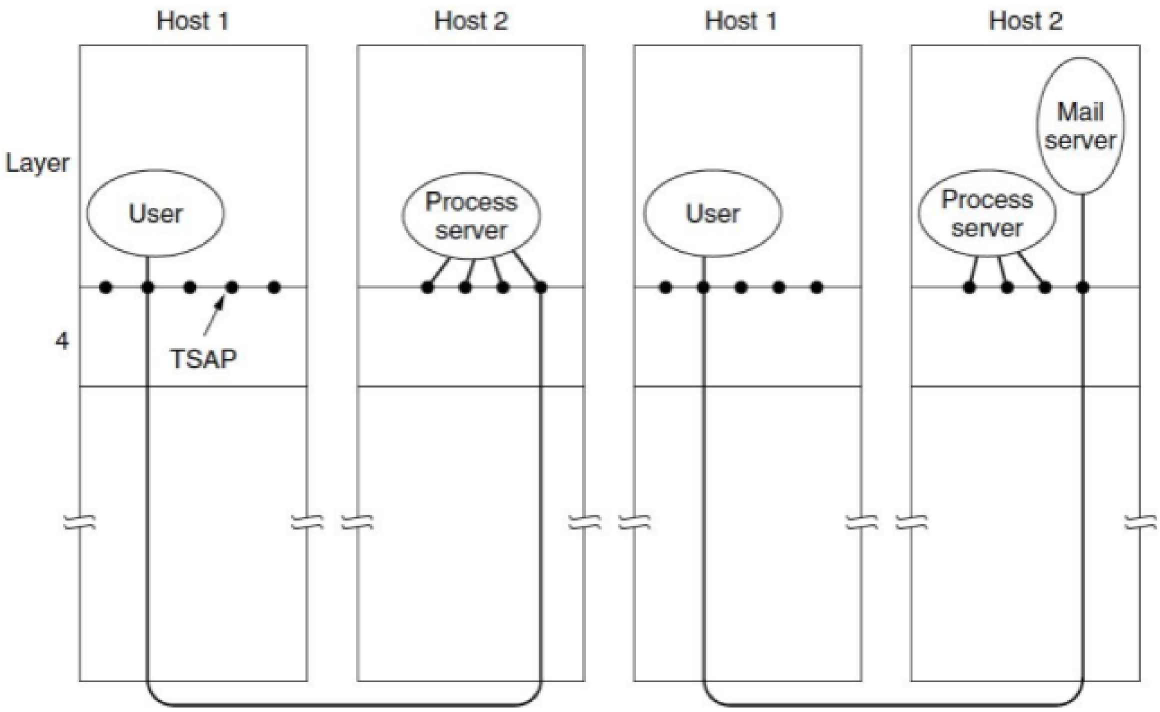
- Some ports are well known and permanently assigned:
    - HTTP, SMTP, Telnet, SSH

2.4.1.6. Initial Connection Protocol

Some servers may only operate sporadically. So why run them all the time?

$\longrightarrow$ Initial Connection Protocol (ICP) conserves resources:

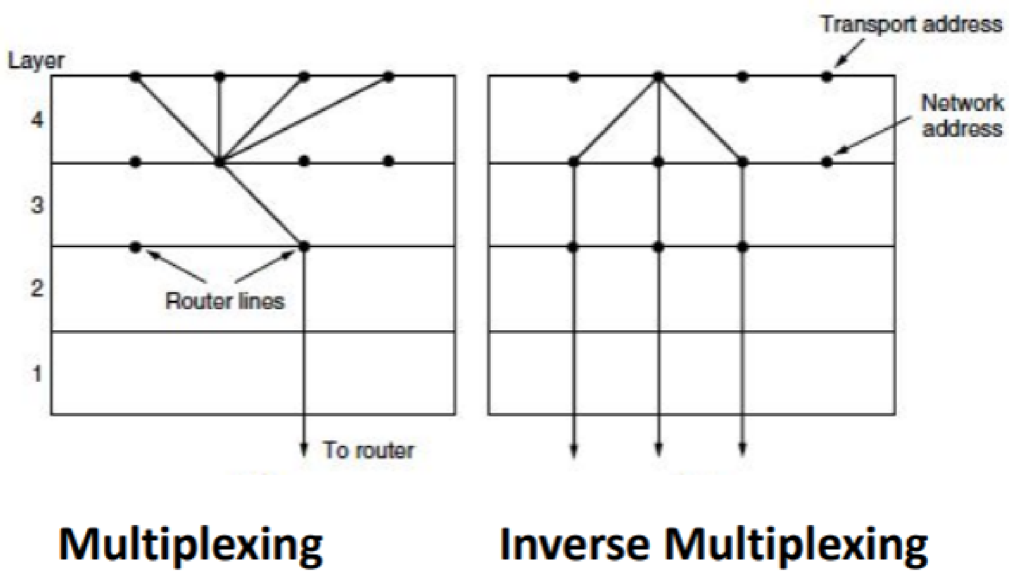
- Process Server listens for connections on well known TSAP addresses
- When a connection is received, the server process is started and passed the TSAP address.



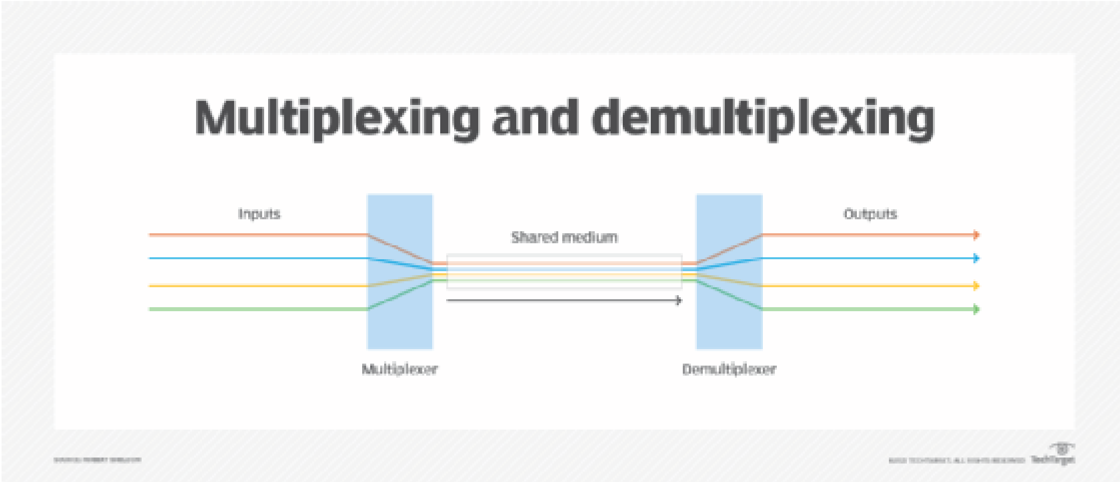
Basically, why keep servers constantly running if they're not constantly needed. That's why we use the Initial Connection Protocol.

So if a host has the ICP service enabled and it receives a connection for a well known port, then the host wakes up the correct service and passes the connection to the now running correct service.

2.4.1.7. Multiplexing Using TSAPs



Multiplexing is actually just taking signals from a lot of different cables and then going to one single cable and transmitting over that, as can be seen below.

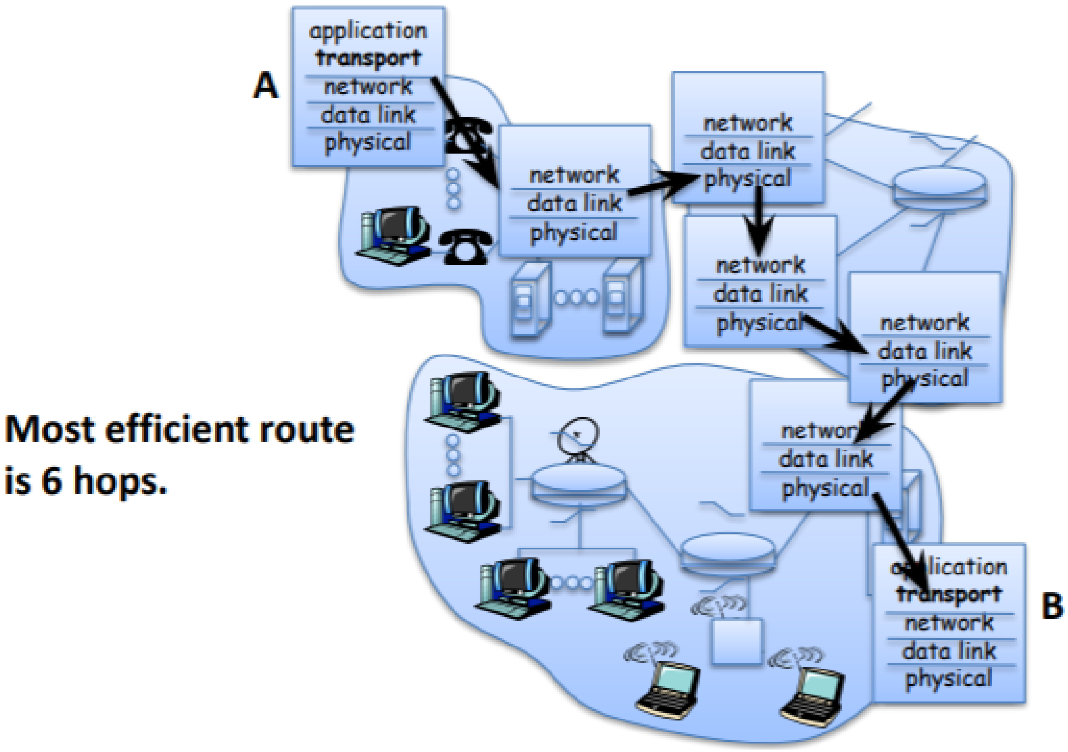


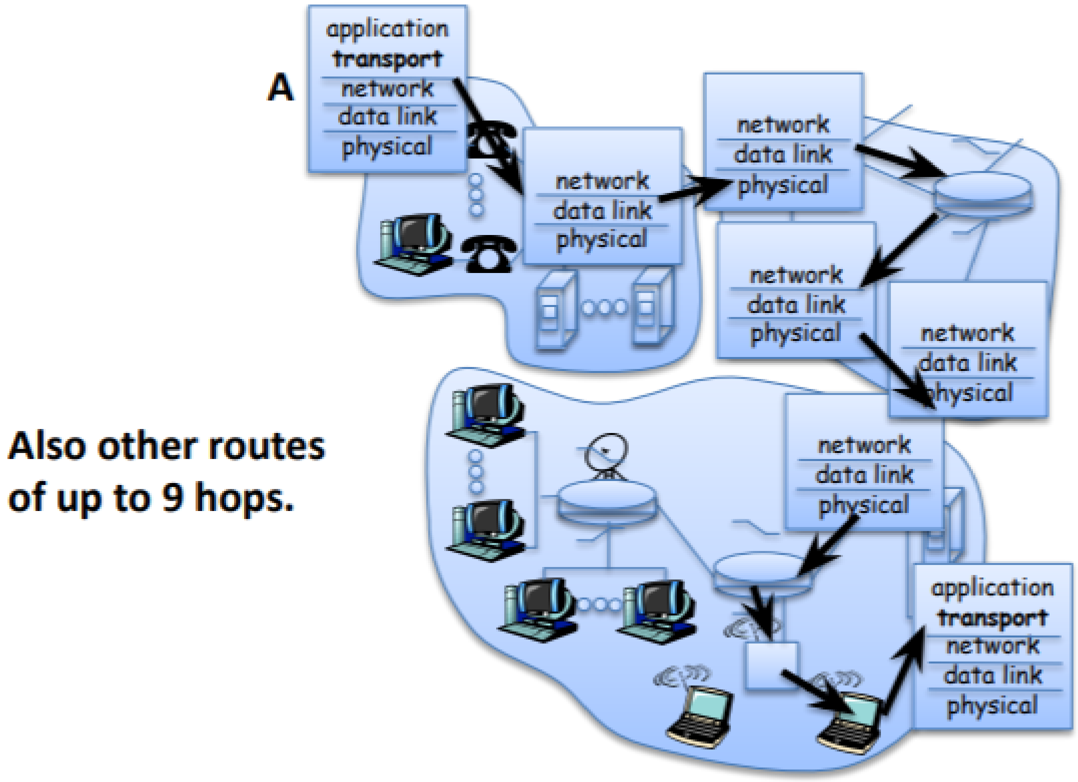
2.4.2. Connection Management

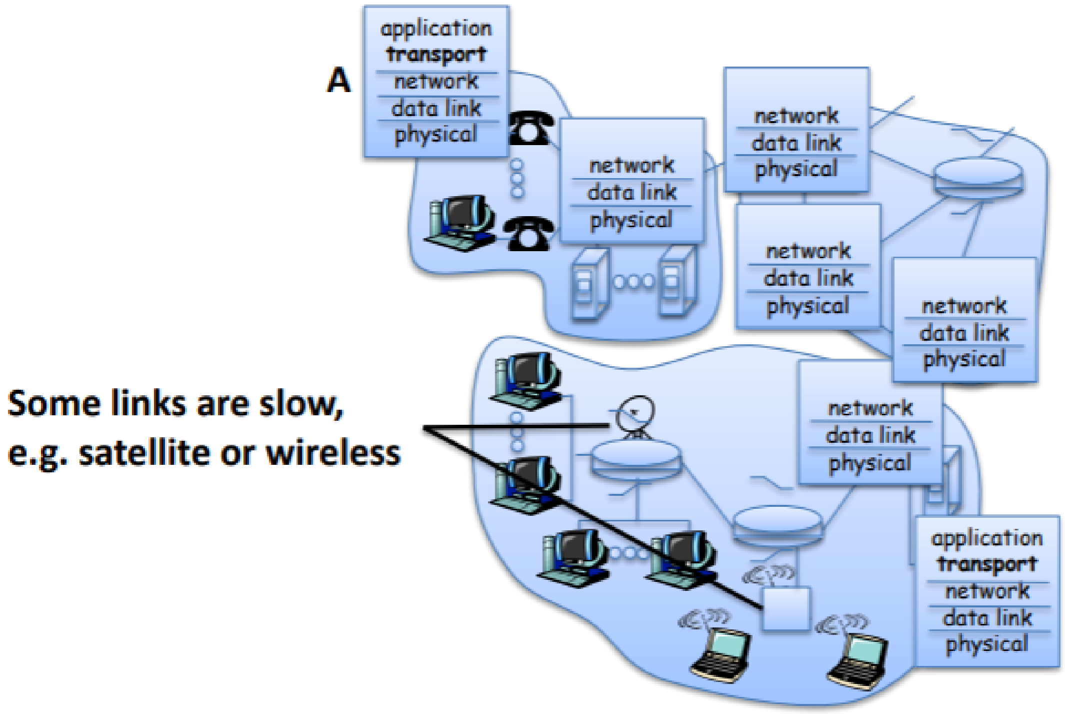
2.4.2.1. Problems of Connection Management

- Transport Layer protocols operate over a network with significant storage potential.
- Some packets may travel slower or longer routes than others

2.4.2.2. Out of Order Packets







All the things above can cause the packets to arrive out of order, be lost entirely or duplicated.

→ A big problem - consider a bank transfer

We send a deposit to an un-trusted party. The packets take the scenic route and we assume they are lost. We resend the packets. The slow packets then arrive resulting in two transfers.

We can solve this by adding a life-time to the packet so that we can reject delayed or duplicated packets. This is now widely used in TCP.

- Bounded Packet Lifetime

Each packet carries a **sequence number** that will not be re-used within **T** seconds (in reality some multiple of packet lifetime)

The standard Internet packet lifetime is 120s. Sequence numbers wrap around to 0. If a machine should crash it would lose the sequence numbers. We solve this by using a real time lock to provide initial sequence numbers during connection. Thus after a crash, hosts continue with a higher sequence number than before.

2.4.2.3. The Two Generals Problem

We have a partial solution. This problem cannot be cleanly solved by the transport layer alone. Higher-level protocols must be designed in such a way that they can tolerate abrupt disconnection. Where this is unacceptable, we must explore probabilistic approaches.

2.4.3. Error Control

2.4.3.1. End-to-End Checksums

Checksums are fixed-size codes generated from larger data using simple hash functions. a consistent hash function is used that will generate the same checksum given the same data. Errors are detected by re-computing segment check-sum and comparing it to the provided value. It is highly unlikely that a packet could be corrupted and still match the provided checksum.

Each individual link is secure, so why isn't our end-to-end connection secure? Because packets can be corrupted within a malfunctioning router.

2.4.3.2. Acknowledgements

Reliable packets require an acknowledgement so we must retransmit packets until an acknowledgement **(ACK)** is received.

If the receiver loses a sequence number, they may negatively acknowledge **(NACK)**, requesting resend.

This requires that the sender stores old messages until they have all been acknowledged

###

2.4.4. Flow Control

2.4.4.1. Sliding Windows !Important!

- Sliding window protocols tackle both error and flow control
- Senders and receivers each maintain a window of messages for which no ACKs have been received.
- The window is a sequence of message IDs, with a lower and upper bound.
- By configuring the window size we can control the flow of packets into the network.

2.5. Sliding Window Protocol

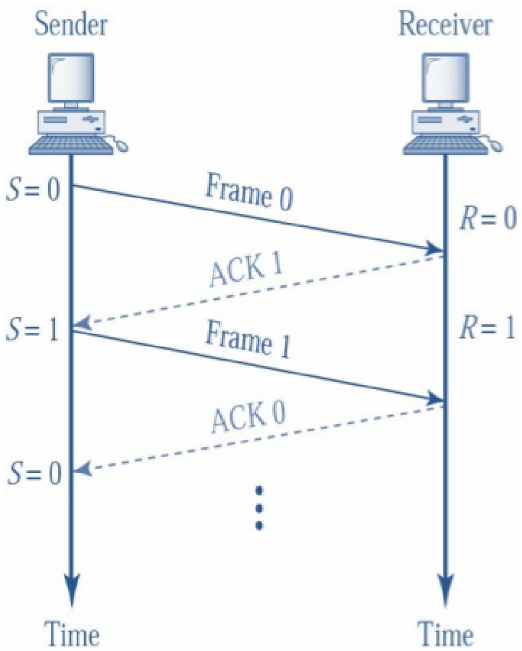
2.5.1. Window Size

- Sliding window protocols may use a fixed window size, or adapt a window size.
- The simplest approach is window of fixed size 1, or **'stop and wait'**
  - Send one segment and wait until it is acknowledged
  - A one-bit sequence number is used to detect duplicates

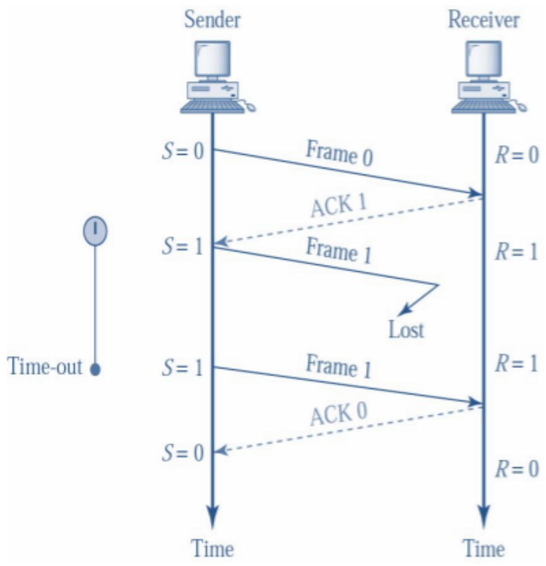
2.5.2. Stop-and-Wait Operation

- Sender keeps last segment until it receives ACK
- Both data and ACKs are numbered alternately 0 and 1
- Sender stores (**S**) with number of the last segment sent
- Receiver stores (**R**) with number of next segment expected
- Sender starts timer on segment send. If an ACK is not received before expiry, send assumes loss or damage and resends.
- Receiver sends ACK with number of next segment when segment is intact.

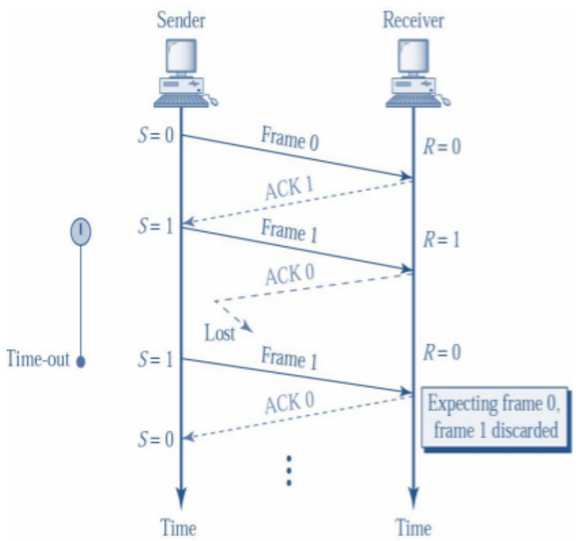
In the following images frame should be segment because we are in the transport layer



- Here we see **normal interaction** with no errors between sender and receiver.

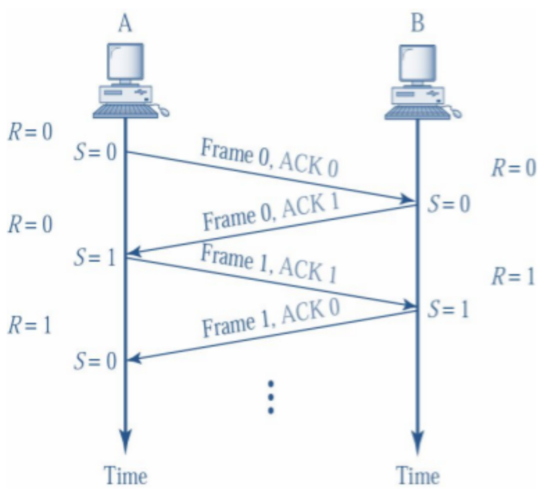


- **Data segment is lost** during transmission or discarded by receiver due to damage.
- Timeout on sender occurs with no ACK.
- Segment is resent, resulting in ACK.



- **ACK segment is lost** during transmission or discarded by sender due to damage.
- Timeout on sender occurs with no ACK.
- Segment is resent, receiver discards and responds with an ACK.

Piggybacking Optimization



- For bi-directional connections, we can do better.
- Why not include the ACK with our data segment?
- This reduces the number of segments sent and therefore traffic.

2.5.2.1. Summary of Stop-and-Wait

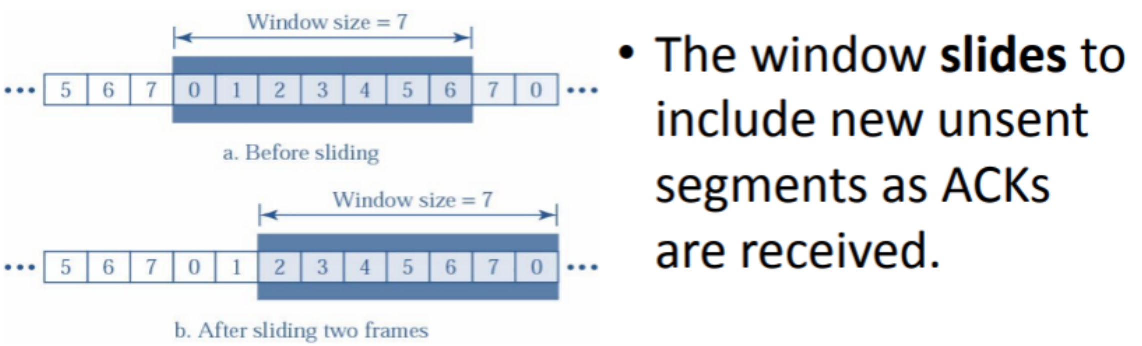
- The simplest sliding window protocol to implement
- Uses very few resources (i,e buffer space is 1 on sender, 0 on receiver)
- But it is inefficient if we have high RTT compared to speed

2.5.3. Go-Back-N

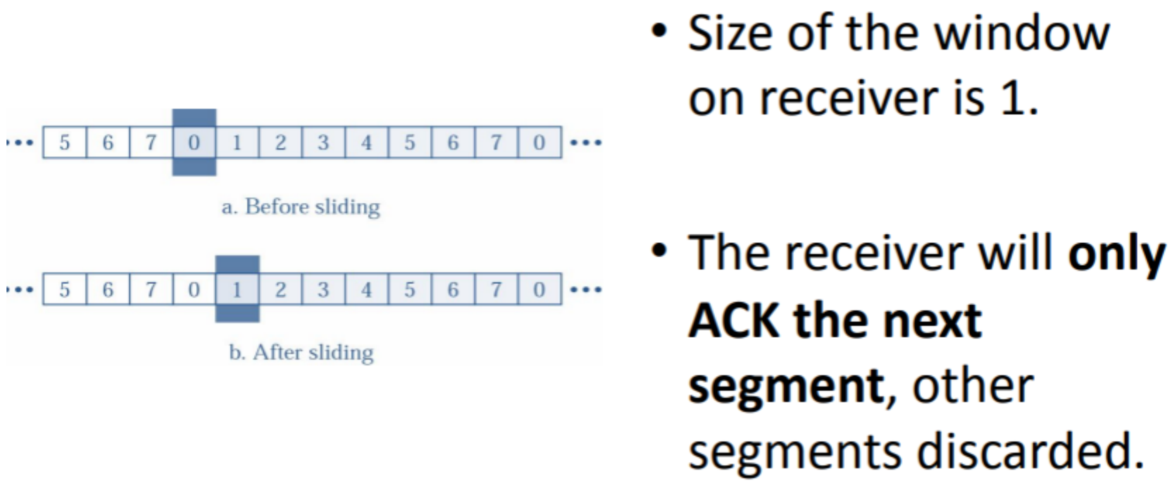
2.5.3.1. Principles

- Segments have a larger range of sequence numbers, that wrap around to 0
- We send **W** segments before requiring an **ACK**
- Keep a copy of the segments until **ACKs** arrive
- This requires extended data structures and more variables than stop-and-wait

2.5.3.2. Sender Window



2.5.3.3. Receiver Window



2.5.3.4. Acknowledgement

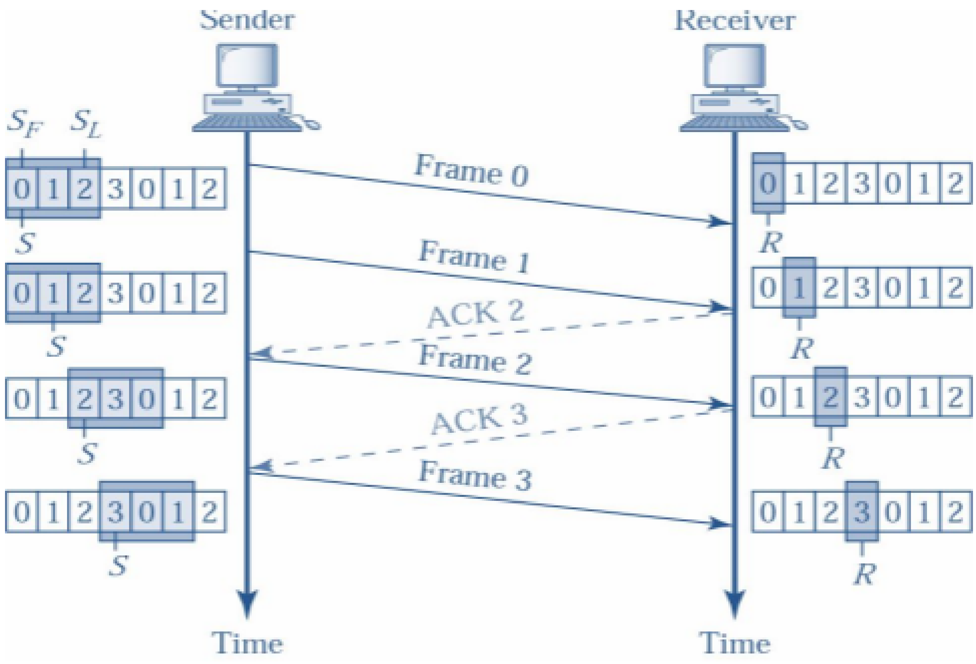
- Sender starts a **per-segment timer**.

- Receiver sends ACK only on next expected segment arriving safely. All damaged or out of order segments are discarded.

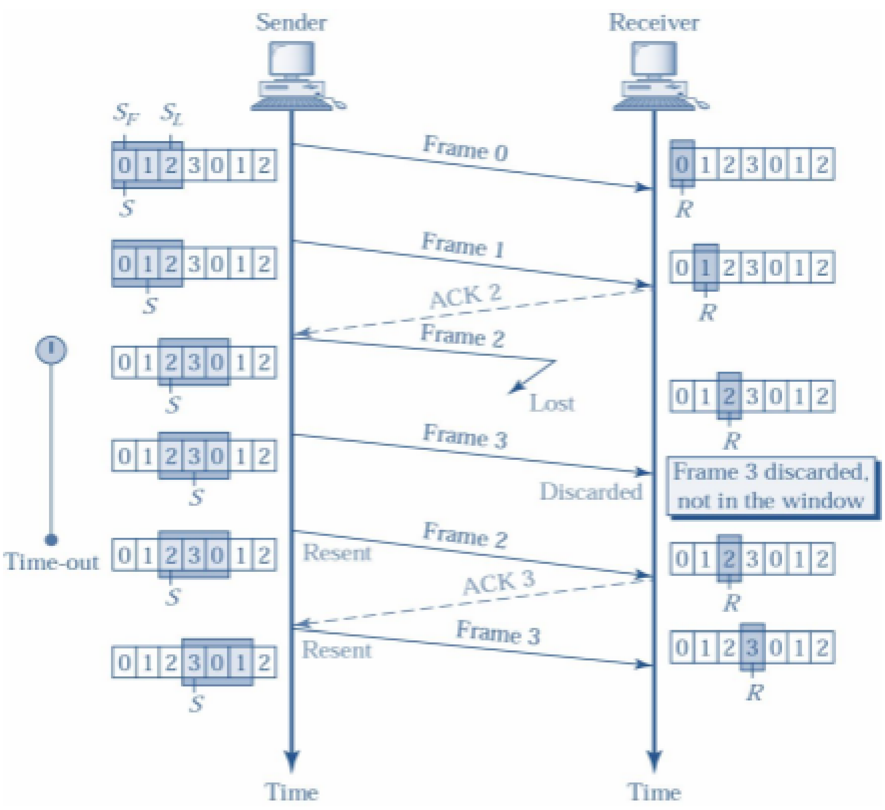
- If timer expires on a segment, all unacknowledged segments in the window are resent.
  - **E.g. if SF = 1, SL=5 and S3 timer expires, sender resends S3, S4, S5 (i.e. it goes back N).**

- Cumulative ACKs acknowledge all packets lower than the specified sequence number.

2.5.3.5. Normal Operation



2.5.3.6. Segment Loss



quick vid:https://www.youtube.com/watch?v=9BuaeEjIeQI

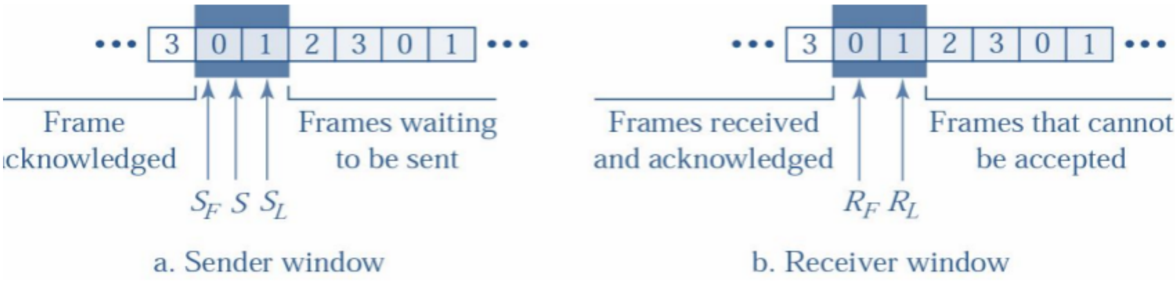
So how does it actually work? Our sender can send an amount of segments indicated by the N in Go-Back-N. It will transmit that amount of segments before waiting for an **ACK** from the receiver. The receiver will only send one ACK at the time. Should a segment be lost or the ACK be lost then our sender will wait till a time-out and then resend all the segments in the current sliding window. If the sender receives an ACK for a segment it will slide the window to the right.

2.5.4. Selective Repeat

2.5.4.1. Principles

- **Go-Back-N** is bandwidth inefficient and slows down the transmission due to the repeated transmission of received packets
- **Selective Repeat** eliminates wasted retransmission of received packets. This is bandwidth efficient but
  - Requires buffers at both send and receiver
  - Requires new variables and book-keeping
- A **Negative ACK (NAK)** reports the sequence number of a damaged or out-of-order segment.

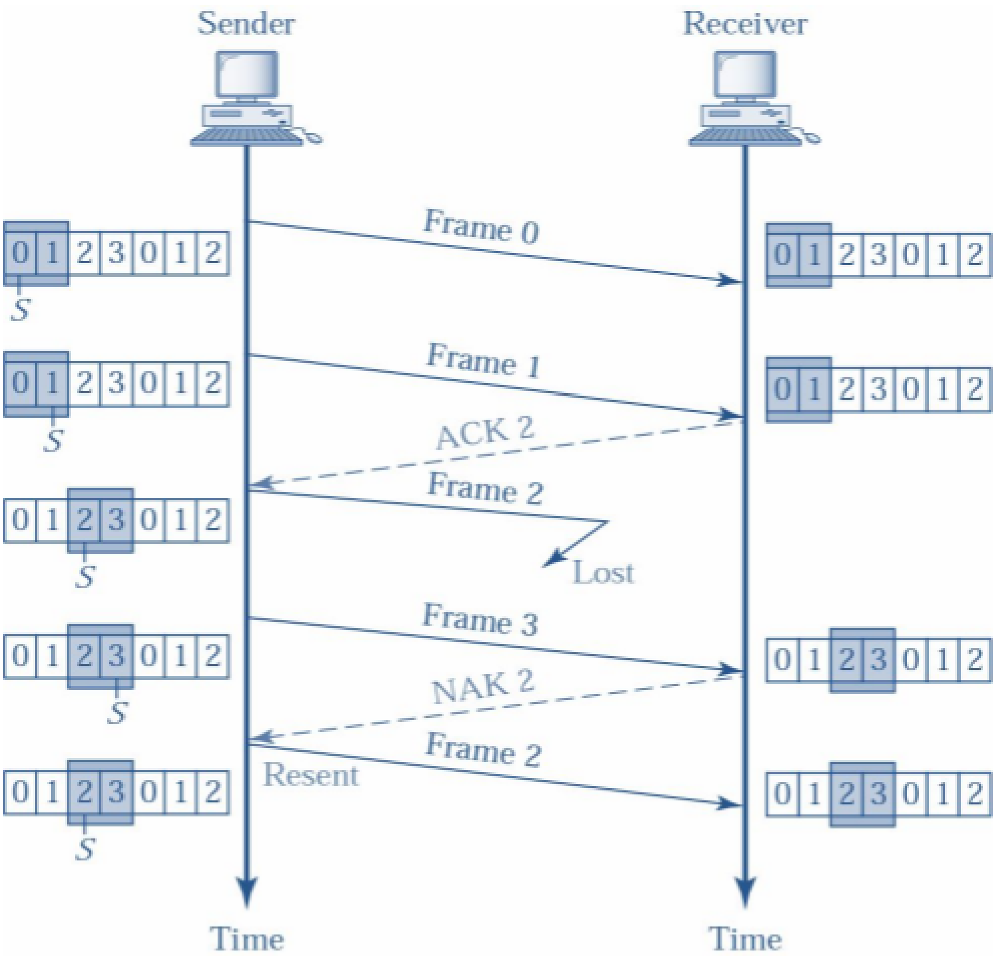
2.5.4.2. Sender and Receiver Windows



- **Note:** the use of a buffer on the receiver means that it can receive packets out of order.

2.5.4.3. Lost Segments

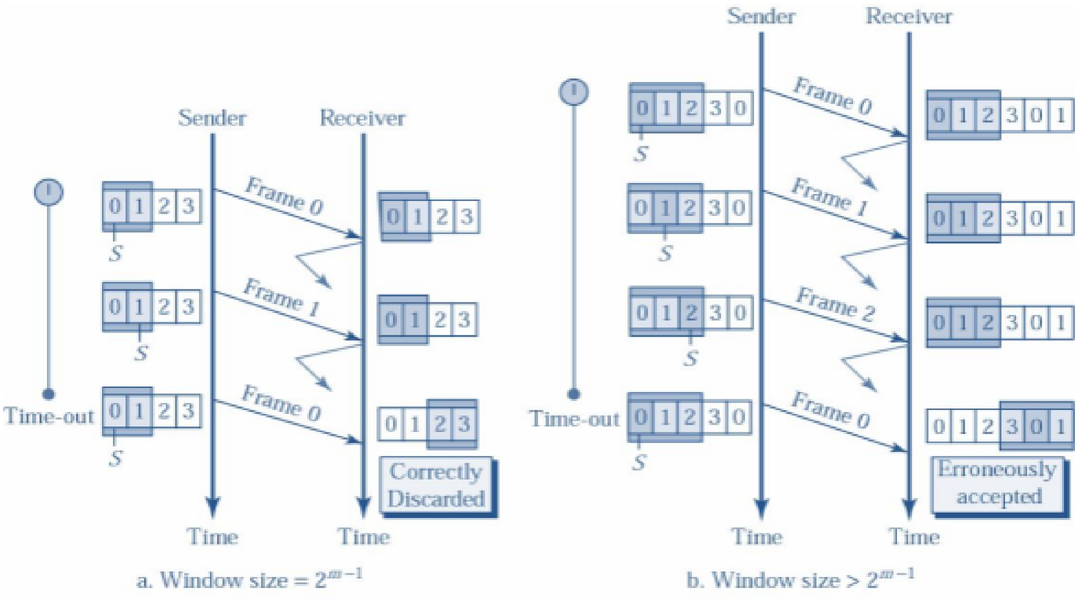
SR: Lost Segments



2.5.4.4. Lost ACKs

- If an ACK is lost then the segment timer will time out
- This causes the segment to be individually retransmitted

2.5.4.5. Window Size Restriction



- **Note:** window size must be equal to or less than half the size of our sequence numbers. If it is larger, then the receiver could erroneously accept duplicate segments.

2.6. Congestion Control

2.6.1. What is Congestion?

If we send too many packets too quickly into the network it becomes congested. Controlling congestion is the combined responsibility of the **Transport Layer** and **Network Layer**.

- Congestion occurs at routers, so the Network Layer should signal it, yet Transport Layer controls dispatching of packets.

2.6.2. Desirable Bandwidth Allocation

- What is the optimal state we are aiming for?
  - We should use all available bandwidth
  - We should avoid congestion
  - We should be fair across Transport entities
  - We should respond quickly to changing usage

$\longrightarrow$ Use all available bandwidth

- Two refinements:
  - Use a little less than all of the **bandwidth** due to 'bursty' traffic
  - We should worry about '**goodput**' not **throughput**

As we approach our **bandwidth limits**, bursts of higher traffic cause losses in network buffers. These losses cause more retransmissions initiating congestion collapse.

As the network approaches congestion, delay rises at an **increasing** rate (due to buffering). Packets are **lost** after the **maximum buffering delay** is exceeded.

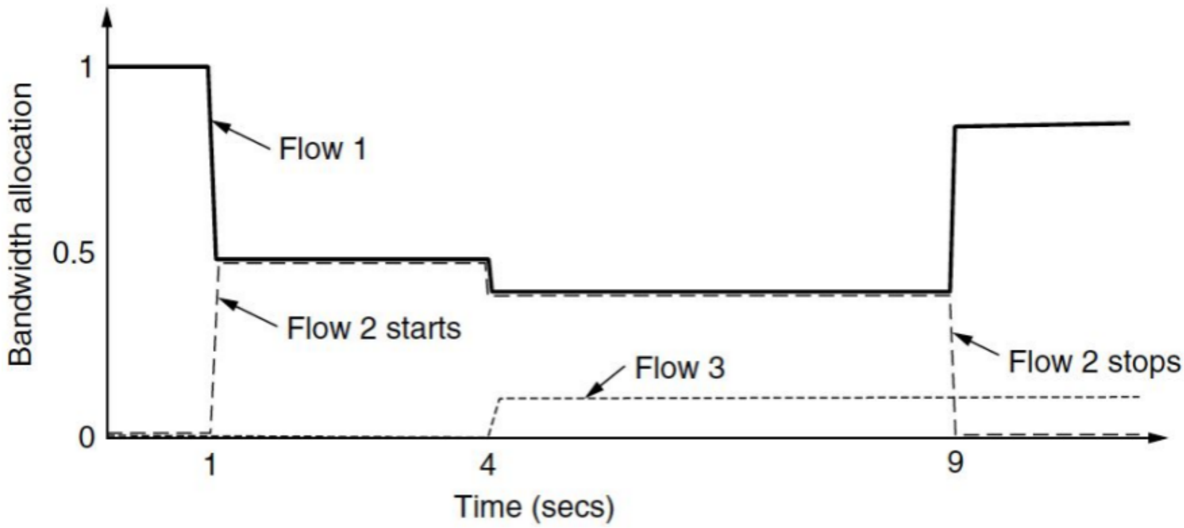
2.6.3. Min-Max Fairness

- Definition: an allocation of bandwidth is **min-max fair** if the bandwidth given to one flow cannot be increased without decreasing bandwidth for another flow
- Fairness is complicated but precise fairness is less important than preventing **starvation and congestion**

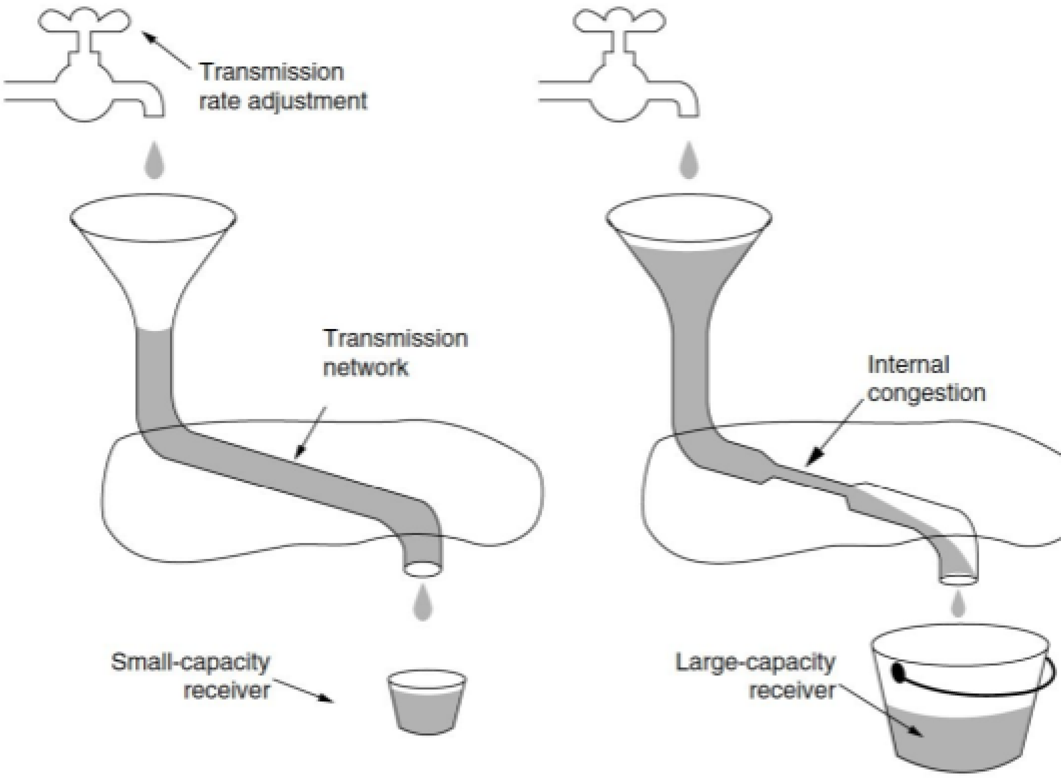
2.6.4. Convergence

Connections are not static. They come and go. Any approach to fair allocation must converge quickly to the ideal operating point.

Ideal Convergence



2.6.5. Types of Congestion



We have 2 two types of congestion:

1. We are dealing with a low capacity receiver which cannot follow the stream of data
2. We are dealing with a congested router in-between

2.6.6. Managing Send Rate

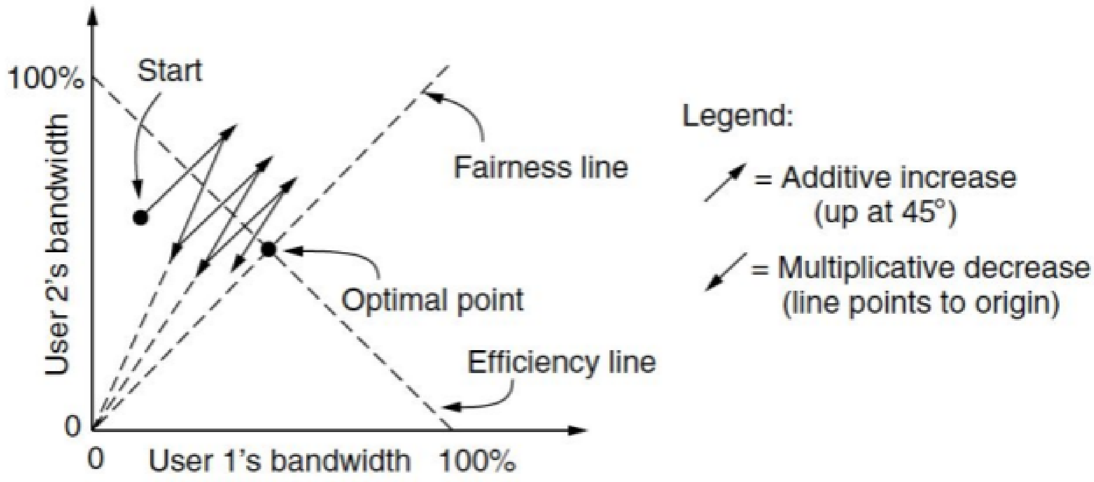
In the case of a low capacity receiver, we need to resize our flow control buffer - **Transport Layer Solution**

In the case of a network congestion, we need to lower our send rate in the Transport layer based upon a signal from the **Network Layer**

TCP does both!

2.6.7. AIMD (Additive Increase, Multiple Decrease)

By increasing in additive fashion, but decreasing by multiples, we converge in saw-tooth fashion around the optimal.



- It's easy to drive the network into congestion and difficult to recover. So our increase in rate should be gentle and our decrease should be aggressive.

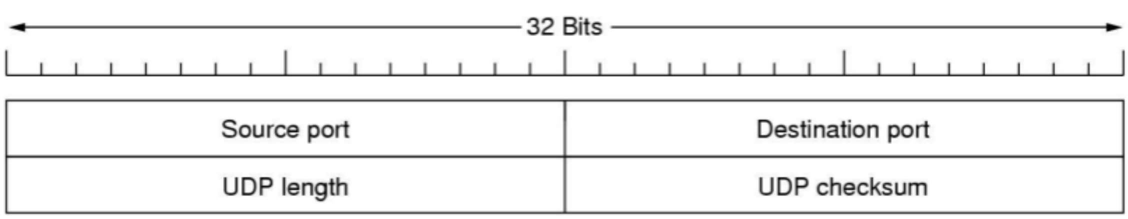
Quick recap of things you should know:

- **TCP** will start by using the slow start algorithm. This means that it will start by sending one packet, should you receive an ACK then you will **double** the amount of packets send. This will keep on doubling until you reach the **threshold**.
- If you reach the threshold, TCP will use the the **AIMD** algorithm. This means it will keep on adding an extra packet to the max of the packets it can send out instead of doubling it as long as the **ACKs** keep on being received.
- If the ACK is lost or timed out then that **Multiple decrease** will kick in and divide the amount of packets the sender can send at one time in half. Reducing the **amount of packets** on the network by a lot and probably solve the **congestion** at the router.

2.7. User Datagram Protocol (UDP)

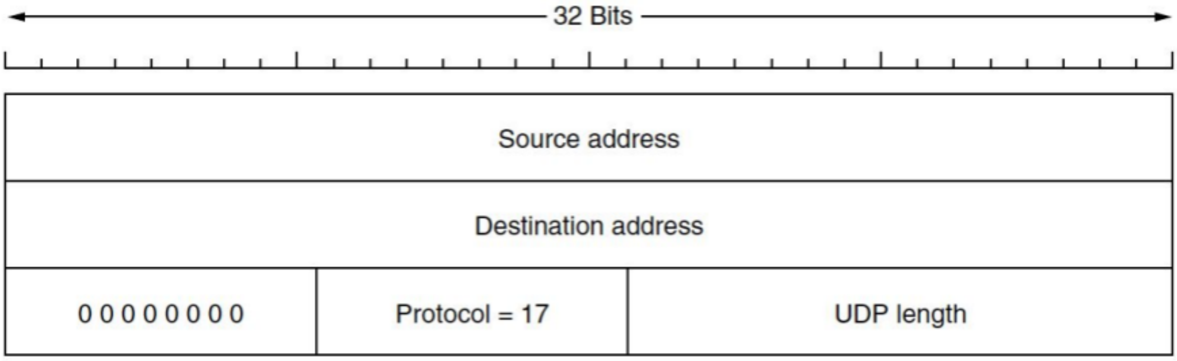
- Connectionless protocol
- Allows for the sending of segments between host with no connection overhead
- Provides no: flow control, ordering, congestion control
- Does provide: ports and checksum

2.7.1. Structure



- UDP gives us **TSAP addresses** and a simple checksum service for end-to-end error checking.

2.7.2. Checksum



- The UDP checksum is calculated based upon a simple hash of the UDP header and the IP pseudo-header.

2.7.3. Benefits

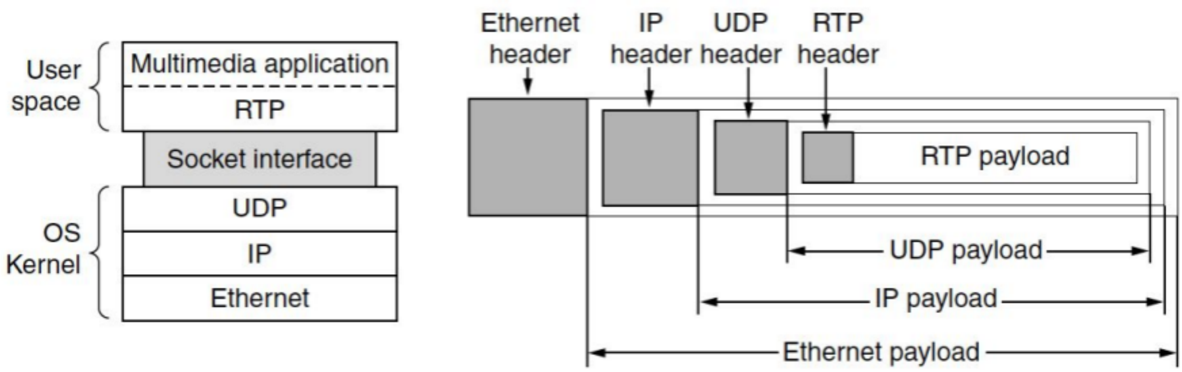
1. Connectionless, so works well with **anycast (e.g. DNS)**, **broadcast(e.g. DHCP, Wake on LAN)** and multicast.
2. Low overhead in comparison to TCP, so it can be implemented on **tiny devices**

2.8. Real Time Protocol (RTP)

- Multimedia applications tend to need the same kinds of services.
- Real-time Transport Protocol (RTP) is common approach to addressing these concerns

Real time data consists of several related data stream: meta-data, video, audio, These must be sychronized.

2.8.1. RTP in the stack



- **Question:** consider the figure. Where should RTP be placed in the stack? Provide arguments.

2.8.2. How does it work?

- The sending RTP library takes in multimedia stream (audio, text, video) and multiplexes them.
- Data is encoded in **RTP** packets and transmitted over **UDP**.
- Receiving RTP library receives, decodes and plays multimedia

2.8.3. Why use UDP for RTP

We don't need every bit of data of a stream, our eyes most of the time can't see the missing frames in a video. We can also interpolate (i.e guess at the contents based upon the previous and following frames.)

Multicast is a critical tool for efficient media distribution.

2.8.4. Features

- We still need to know if packets went missing, but the action we take could be different: **interpolate**, **ignore**, etc.
- RTP packets carry linearly increasing sequence number 1 higher than its predecessor.
- RTP also supports extension headers for more advanced functionality.

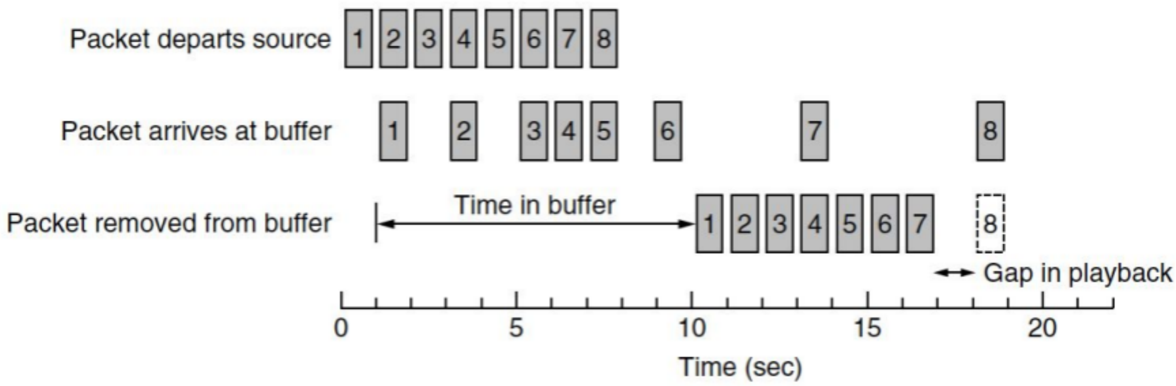
2.9. Real Time Control Protocol (RTCP)

- Real Time Control Protocol (RTCP) controls RTP streams.
- Providing feedback and control on delay, variation in delay, jitter, bandwidth and congestion
- Provides synchronization of streams.

2.9.1. Jitter

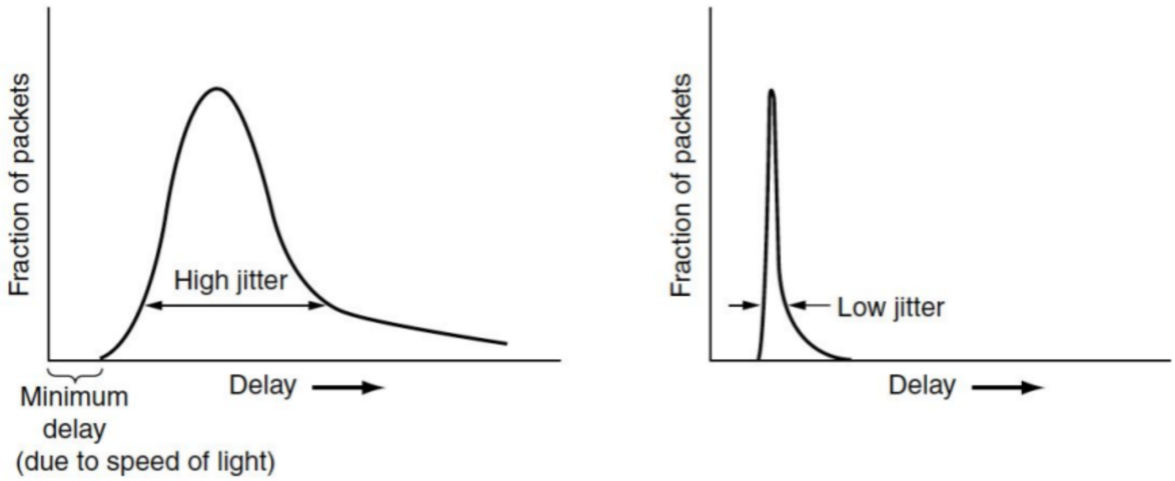
Packets take a variable amount of time to travel between hosts. This causes out of order arrival and thus results in variable delay between sender and receiver. This variability in delay is knows as **jitter**.

Jitter (2/2)



- We can minimize the observable effect of jitter using a buffer. **Note:** the delay on segment 8 was too great. We must extend our buffer or drop it.

High Jitter and Low Jitter Networks



- **Note:** the average delay is not very different, but the buffer required to capture 99% of packets differs significantly.

2.9.2. Modifying Playback Point

- The playback point determines how long to wait at the receiver before playing incoming RTP packets
- We must select it correctly, or end up with either (a.) unnecessary delay or (b.) high loss.

2.10. Transmission Control Protocol (TCP)

- TCP provides a reliable end-to-end byte-stream over an unreliable internetwork
- More than 90% of Internet traffic is TCP

The TCP Transport Entity

- The Transport Entity is the software process that **implements** the **TCP** protocol.
- The transport entity is responsible for:
    - Splitting byte streams into segments
    - Reliable transmission
    - Reconstructing byte streams
    - Efficiently using bandwidth and avoiding congestion

Addressing by Ports

- TCP implements TSAP using ports
- Ports below 1024 are reserved for well known applications

TCP Connections

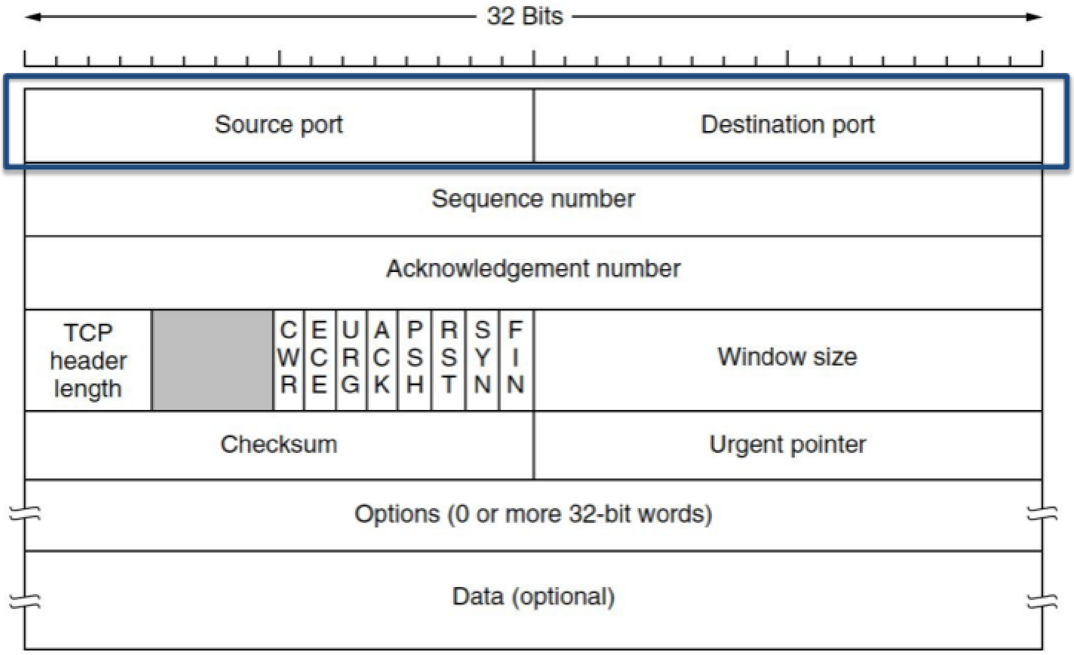
- TCP connections are full duplex, support simultaneous bi-directional communication
- TCP connections map to exactly two sockets
- The writing of data to TCP element is decoupled from reading of that data
    - Imagine you write four 512KB blocks. They may be divided into eight 256KB blocks or two 1024KB blocks. The transport entity is responsible.

Sequence Numbers and Segmentation

- Every byet has it own 32-bit sequence number.
- TCP packets have limited size:
    - Header limits packet size to **64K**

2.10.1. The TCP Header

The TCP Header



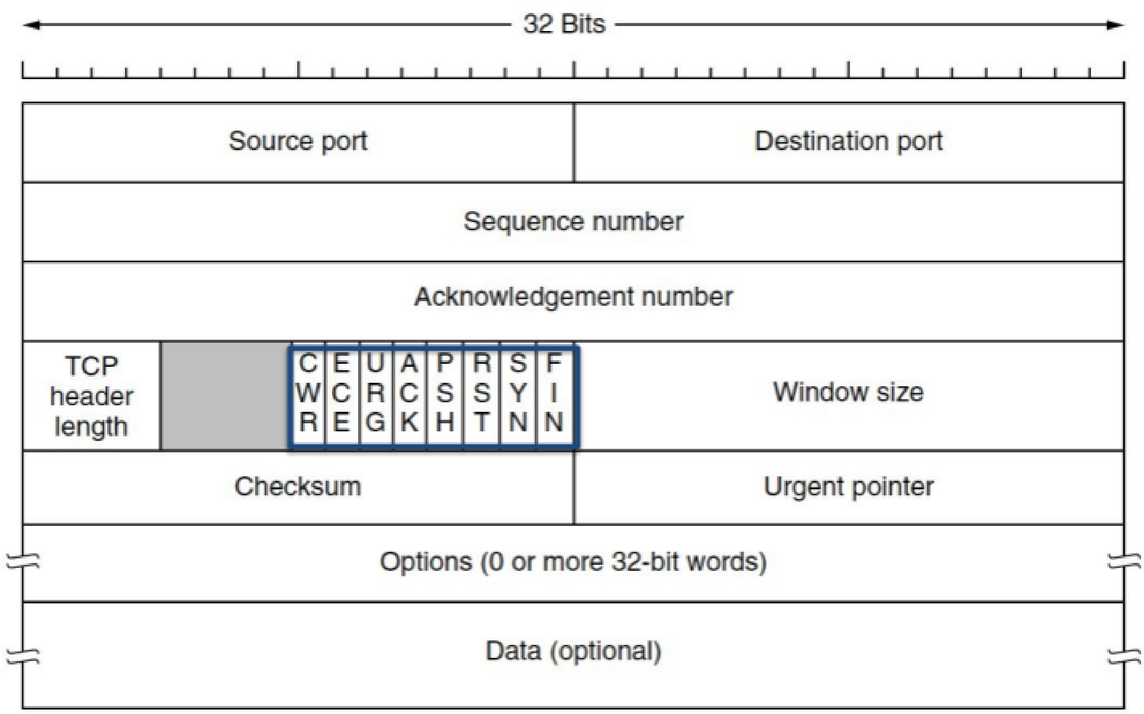
2.10.1.1. Source and Destination Ports

- Source port = **TSAP** address from which segment was sent
- Destination port = **TSAP** address to which the segment is sent
- **Note**: a connection is implicitly identified by source TSAP + source NASP + dest. TSAP + dest NSAP + procotol

2.10.1.2. TCP Header Length

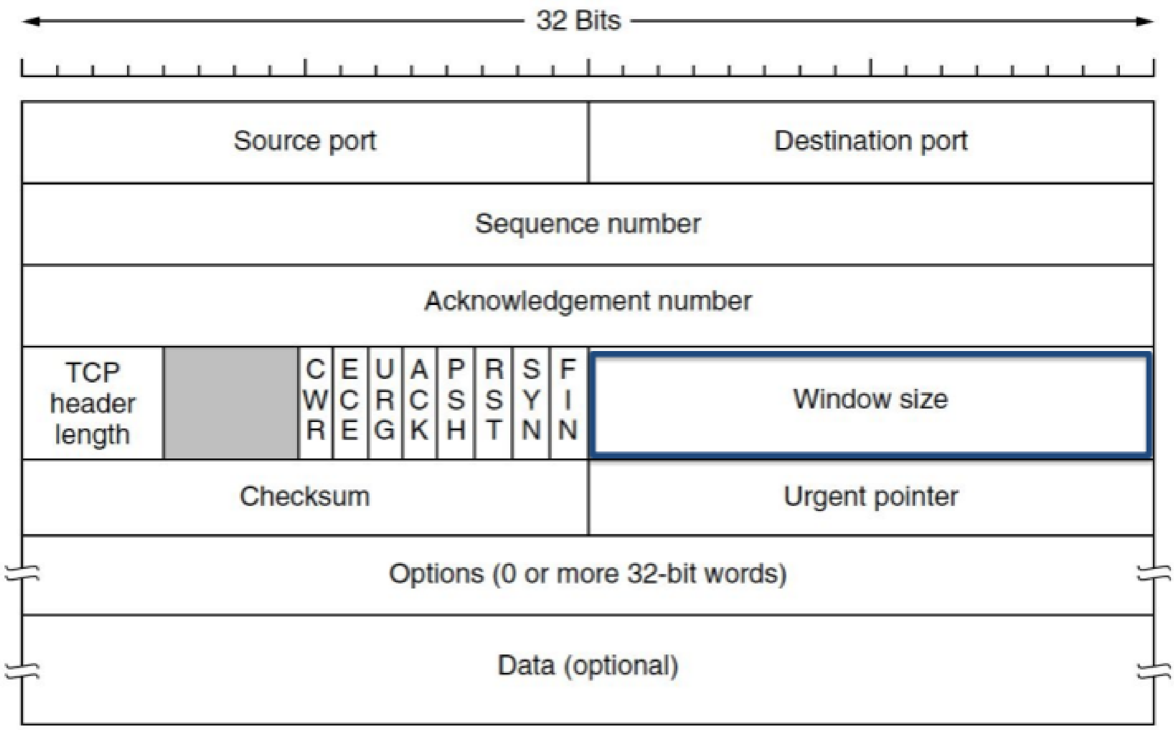
- TCP headers are variable length due to the use of **option fields**

2.10.1.3. One-Bit Flags



- **CWR** : Congestion notify: slowed down.
- **ECE** : Congestion signal: slow down.
- **URG** : Urgent pointer is set.
- **ACK** : Indicates a valid ACK number.
- **PSH** : Pushed data: do not buffer.
- **RST** : Connection reset signal.
- **SYN** : Indicates connection establishment.
- **FIN:** Used to release a connection.

2.10.1.4. Window size



- TCP implements a variable-sized sliding window
- Specifies how many bytes may be sent starting at the byte acknowledged
- Window size of 0 indicates that bytes up to ACK number have been received, but the receiver requires time to process them.

2.10.1.5. Checksum

- Identical to UDP, calculated from the TCP header and the IP pseudo-header. **Compulsory**

Urgent pointer

- if urgent flag is set, this pointer gives the byte offset where urgent data is located.

2.10.1.6. Options

- Just a place for optional headers.

2.10.1.7. Data

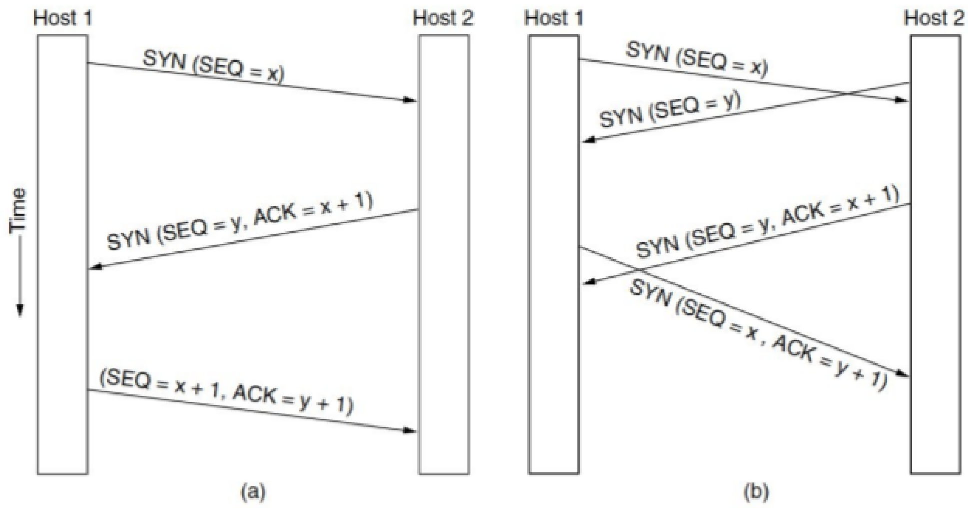
- The remainder of our packet is used for data.
  - Max payload = 65535 – IP Header – TCP Header.
  - Max payload = 65535 – 20 – 20 = 65495.

THIS IS OPTIONAL

2.10.2. 3-Way Handshake

- Connections are established and released using the 3-way handshake
- One or both sides passively listen for incoming connections **SYN(SEQ=X).**
- If no application handler, they respond with **RST**
- If they are willing to accept, they respond with **SYN(SEQ=Y,ACK=X+1)**.
- If not willing to accept, no ACK is sent.

3-Way Handshake (2/3)

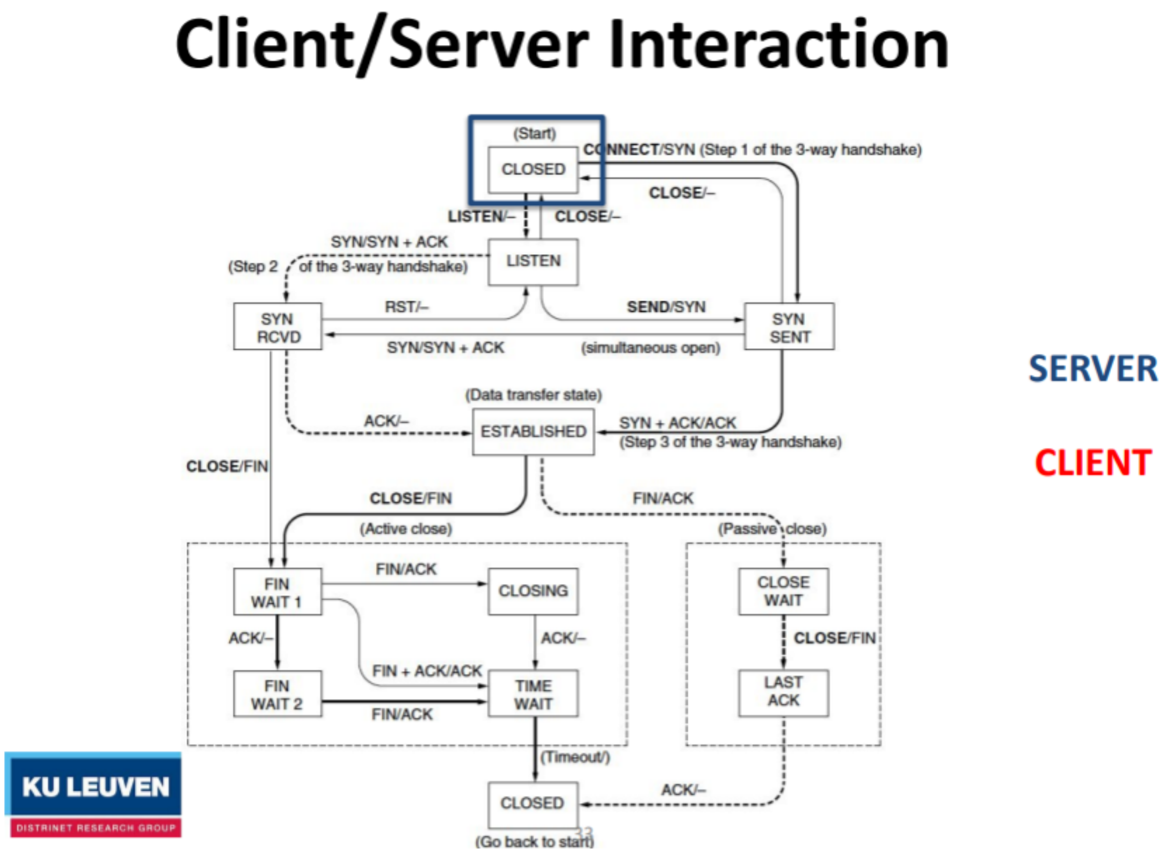


Note: connections may be established simultaneously, but this also results in 1 connection entry.

- Disconnecting is best thought of as operating on two simplex connections.
  - Either party sends a FIN segment.
  - When FIN is ACK'd, connection is shut down for new data. It may still flow in other direction.
- To avoid the **two-generals problem**, timers start on FIN transmission (twice packet lifetime) and connections are closed after this time.

2.10.3. TCP as a State Machine

- TCP can be represented as a state machine with **11 states**
- In each state, legal events cause a transition to a new state.



2.10.4. Receiver Control of Transmission

When receiver window is 0, no transmission can occur with two exceptions

1. **Urgent traffic**
2. **Window probe**

2.10.5. Tinygram Syndrome

This is basically, TCP can be such a waste of resources because it's possible we send several segments with headers and such for just one byte of data.

- Transmission = 20 + 20 + 1 bytes.
- Acknowledgement = 20 + 20 bytes.
- Window = 20 + 20 bytes **(120 bytes!)**

→ Solution: Delayed acknowledgements are a simple approach to addressing Tinygram Syndrome

- Until timeout, wait for data to be transmitted.
  - If data is transmitted before time-out, piggyback acknowledgement in data.
  - If timeout occurs, send acknowledgement

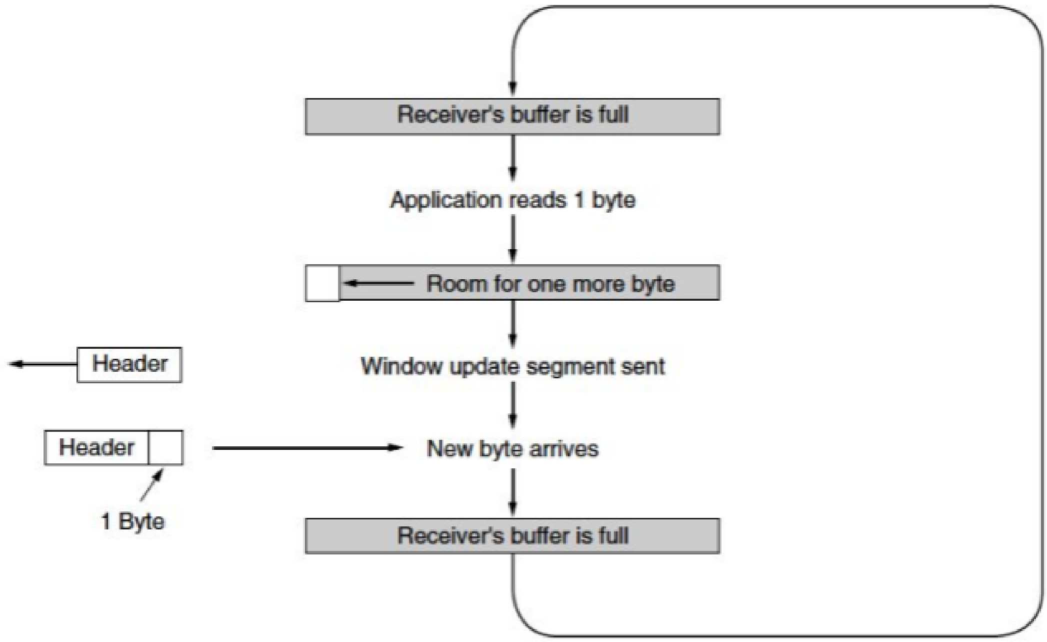
2.10.6. Nagle's Algorithm

Nagle's algorithm, while there is a sent segment with no ACK, buffer output until we have a full segment, then send at once.

This would not work well for real time systems, computer games,...

2.10.7. Silly Window Syndrome

Silly Window Syndrome occurs when the sending entity receives data in large blocks however the receiving application only reads data 1-byte at a time.



2.10.8. Clarke's Algorithm

Delay window updates until the window can receive the maximum segment size, or until the buffer is half empty.

2.10.9. Retransmission Time Out

- Retransmission Time Out (RTO) in TCP determines how long to wait for an ACK before retransmitting a segment.
  - If an ACK is not received, we retransmit and restart the timer.
- How to select an RTO period:
  - What happens if it is too long?
  - What happens if it is too short?
- TCP **dynamically adapts** the RTO period

2.10.10. Estimating Round Trip Time

For each connection maintain Smoothed Round Trip Time (SRTT), the best estimate for current round-trip time.

2.10.11. Estimating Variance in Delay

The first approaches to setting RTO used a fixed multiplier of RTT, but this has a problem:

- Recall that when load approaches capacity, delay becomes large and highly variable.

2.10.12. Putting it All Together

We combine our measurement of RTT and RTTVAR in a simple weighted sum to calculate RTO

$$RTO = SRTT + 4 \times RTTVAR$$

2.10.13. The 'Persistence' Timer

- This timer is used to determine when to send window probes
  - The timer starts when a window of size 0 is reported
  - If a non-zero window size update is received, the persistence time is cancelled.
  - If the timer expires, a window probe is issued.

2.10.14. The 'Keep Alive' Timer

- The Keep Alive Timer is restarted whenever a message is received from a remote host.
- If the timer expires, the remote host is probed to ensure that it has not crashed.
- If no response is received, the connection is terminated
- This is optional

2.10.15. Congestion Control

Congestion Control is a key feature of TCP

The network layer detects congestion and takes action (e.g. dropping packets).

- Congestion notification is explicit or implicit
- Notifications may be precise or imprecise.

TCP packet loss is not explicit or precise

2.10.16. TCP and AIMD

- The same as was previously explained.

2.10.17. The Congestion Window

Congestion Window defines the bytes a sender may have in transmission at any time.

Note: the congestion window is maintained separately to the flow control window.

We now have **two windows** that limit how fast we can transmit packets into the network.

1. The flow control window matches **host capacity**.
2. The congestion window matches network capacity

TCP uses **whichever window size is smaller** at any given time. This ensures that neither hosts nor routers are overwhelmed.

2.10.18. Packet loss as Congestion Signal

Is packet loss is a good signal of congestion?

- Wired routers will always drop packets when congested and seldom otherwise.
- Wireless devices may drop packets unpredictably, so here it is a poor signal (more later).

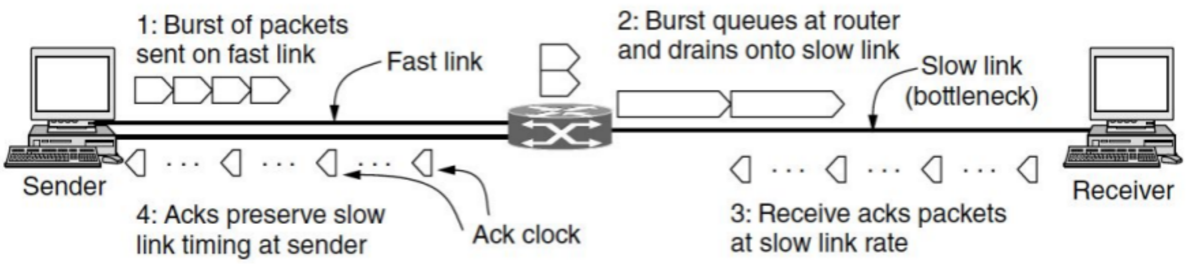
If we base our congestion control on loss, we need a good retransmission timer.

$\rightarrow$ Even if we can infer congestion from loss, we cannot send packets into the network in large bursts.

Consider a host (H) communicating via a 1Gbps Ethernet link and a 1Mbps ADSL link.

– H uses a 64KB window.
– Transmitting this over the 1GPS link takes 0.5ms.
– Transmitting this over the 1Mbps link takes 0.5s.
– This is long enough to disrupt low-latency applications like Voice over IP (VoIP).

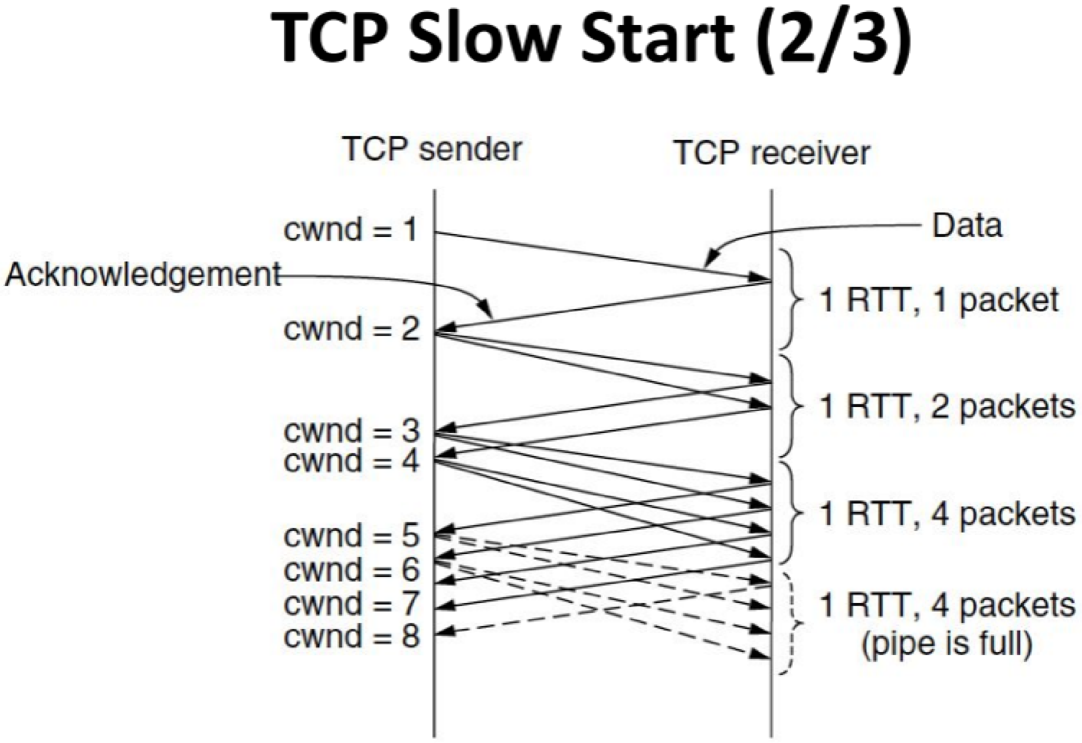
We must match the timing of segment transmission to match the speed at which they are transmitted across the slowest link. Otherwise our bursty traffic periodically blocks these low-bandwidth links. We can discover this rate by sending a small bursts of traffic.



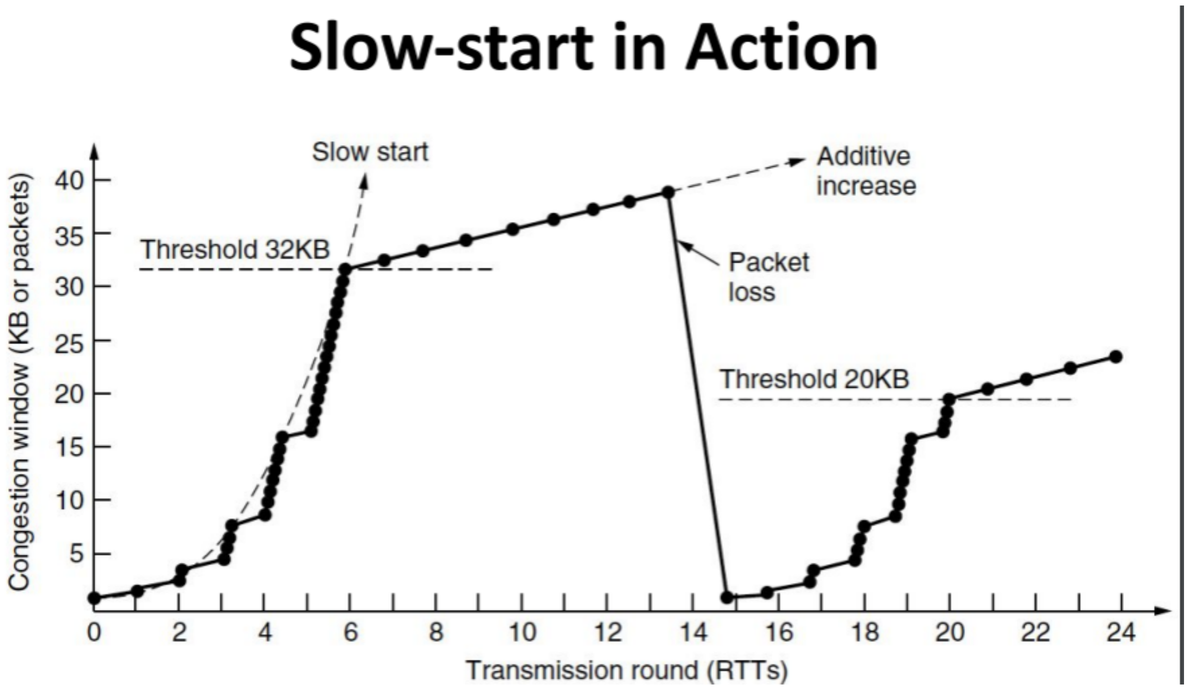
We inject new traffic into the network only as fast as we can receive ACKs. This smoothes out our traffic flow and eliminates bursts that could cause congestion. Thus we avoid overwhelming slow links with bursts of traffic.

2.10.19. A Short coming of AIMD

The additive increase of AIMD can be slow for high capacity connections, so first we use the TCP 'slow start' algorithm as mentioned earlier.

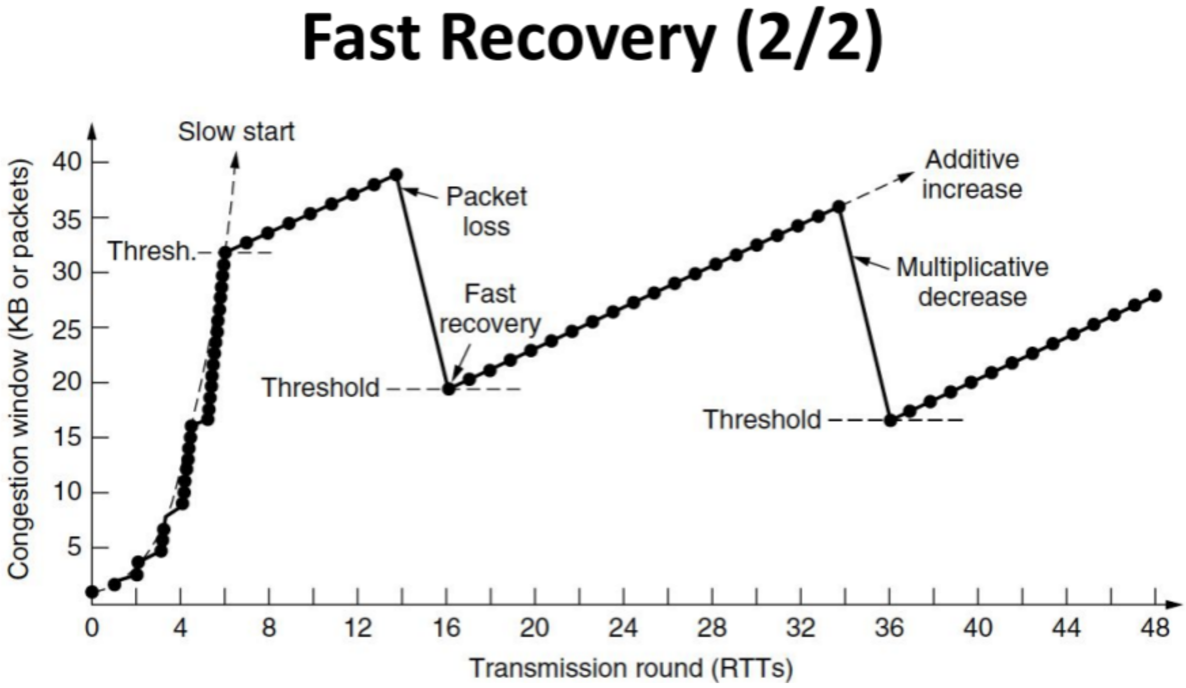


If we keep up Slow Start without stopping, congestion will occur soon so we need a threshold to keep slow-start in check. What would be a good initial value for the threshold? Flow Control Window Size. When the threshold is breached, it is re-set to half of the congestion window. TCP then switches to additive increase until packet loss occurs, when slow start re-starts.



2.10.20. Fast recovery

So instead of completely restarting the amount of packets we're sending, We wait until the number of packets in the network falls to the new threshold (about 1/2 RTT). Once we have fallen below the thresh, we begin AIMD again.



Note: We can only recover from one packet loss.

2.10.21. Selective ACK

Selective Acknowledgements (**SACK**) allows fast recovery after multiple loss, this is achieved by providing extra information within acknowledgements. SACKs allow a **receiver** to acknowledge **non-consecutive** data, so that the sender can **retransmit** only what is **missing** at the receivers end. This is particularly helpful on paths with a large bandwidth-delay product.

Note: SACK is now widely deployed.

2.10.22. Explicit Congestion Notification (ECN)

ECN allows end-to-end notification of [network congestion]() without dropping packets. Both the sending and receiving host must support ECN.

- A **flag** in the IP header specifies if the encapsulated TCP segment uses **ECN**
- When congestion is approaching, routers will set the ECN congestion flag in transit.

Unfortunately not very widely deployed.

3. Chapter 3 The Network Layer

3.0.23. Terminology

- Autonomous Systems (AS): operated by different organizations – a collection of prefixes identified by a 32-bit ASN.
- Intra-domain routing: within an AS, using an interior gateway protocol
- Inter-domain: between AS, using an exterior gateway protocol.
- Internet eXchange Points (IXPs) connect autonomous systems.

3.1. Role of the Network Layer

The network layer is the lowest layer concerned with end-to-end delivery.

End-to-end transmission requires many individual hops:

- Network offers alternative routes
- select to minimize overloading and idle routers.

The Transport Layer should be shielded from this complexity

3.1.1. Store and Forward Packet Switching

Very basic

Routers store a packet until it is fully transmitted and verified by checksum. It is then relayed to the next router on the path to the destination.

3.1.2. Services to Transport Layer

Common goals:

- Service should be independent of router technolgoies
- Transport Layer should be shielded from number, type and topology of routers
- A uniform addressing scheme should be exposed to the Transport Layer

Connectionless Protocols:

- Argues that the network is inherently unreliable
- Transport Layer handles reliability, no ordering, error-detection or correction.

Connection-oriented Protocols:

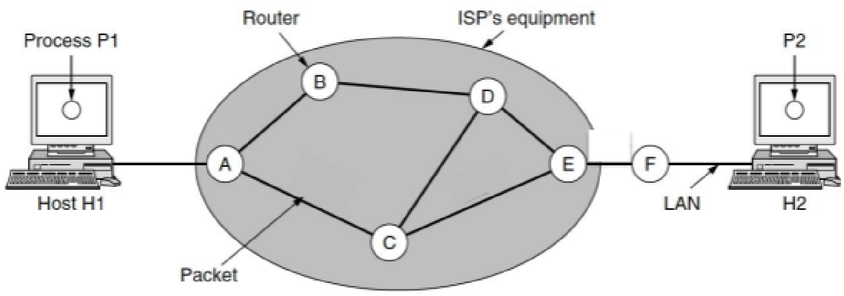
- Argues that end-to-end **QoS (Quality of service)** is needed at Layer 3
- Network Layer provides specific quality of service support for transport-layer flows

3.2. Connectionless

3.2.1. Principles

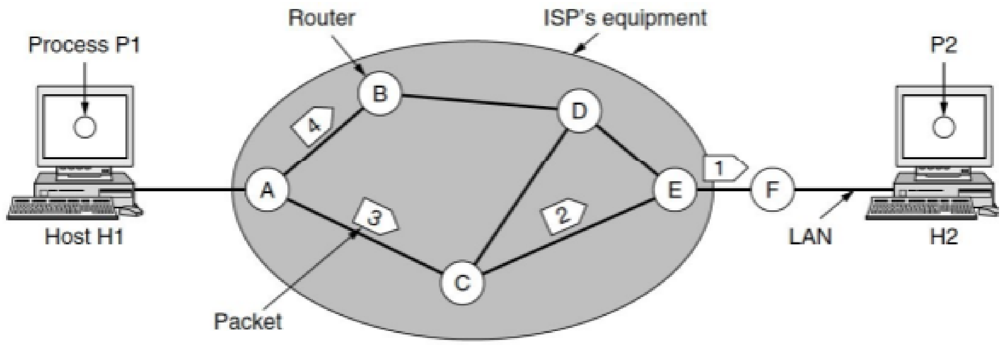
- Packets are injected into the network individually:
  - Each packet is routed independently
  - No connection setup is necessary
  - Each packet carries the end address
- We refer to units of transmission at the Network layer as 'Datagrams'.

3.2.2. Example



A's table (initially)		A's table (later)		C's table		E's table	
A	–	A	–	A	A	A	C
B	B	B	B	B	A	B	D
C	C	C	C	C	–	C	C
D	B	D	B	D	E	D	D
E	C	E	B	E	E	E	–
F	C	F	B	F	E	F	F

- The best link to use may **change over time** and thus packets follow different routes.



A's table (initially)		A's table (later)		C's table		E's table	
A	–	A	–	A	A	A	C
B	B	B	B	B	A	B	D
C	C	C	C	C	–	C	C
D	B	D	B	D	E	D	D
E	C	E	B	E	E	E	–
F	C	F	B	F	E	F	F

Packets 1,2 and 3 are routed from H1 to H2 via {A,C,E,F}. Packet 4 is routed via {A,B,D,E,F}

IP is the best example of a connectionless Network Layer protocol

3.3. Connection-oriented

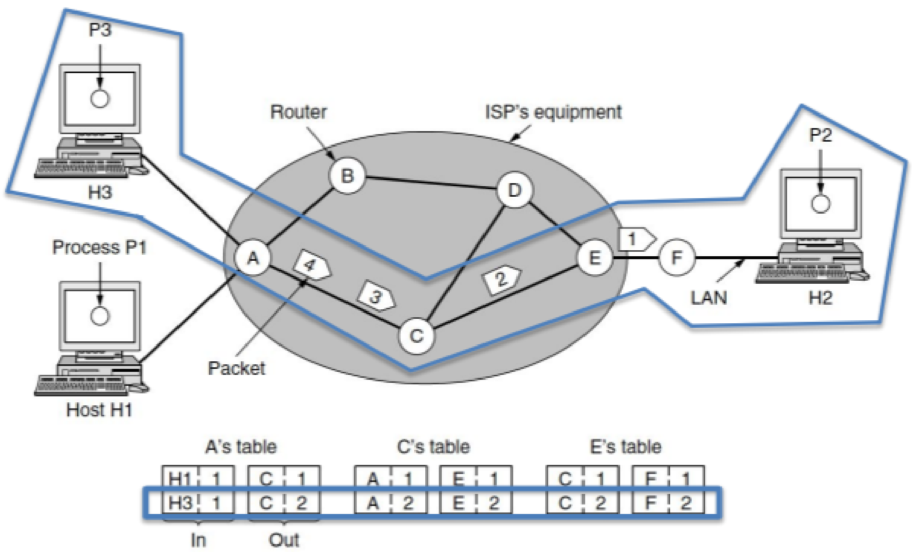
3.3.1. Principles

Based upon "Virtual Circuits":

- Circuits are calculated at connection setup.
- Packets follow the same circuit
- Circuits terminate along with connections

Each packet carries its circuit identifier rather than its end address

3.3.2. Example



The second line of routing tables defines the circuit from H3 to H2. Note the outgoing label on A is different than the incoming label. **Why is this?**

→ **Label Switching**

MPLS is a good example of a connection-oriented protocol used on the Internet.

3.4. Connection-oriented vs Connectionless

Issue	Datagram network	Virtual-circuit network
Circuit setup	Not needed	Required
Addressing	Each packet contains the full source and destination address	Each packet contains a short VC number
State information	Routers do not hold state information about connections	Each VC requires router table space per connection
Routing	Each packet is routed independently	Route chosen when VC is set up; all packets follow it
Effect of router failures	None, except for packets lost during the crash	All VCs that passed through the failed router are terminated
Quality of service	Difficult	Easy if enough resources can be allocated in advance for each VC
Congestion control	Difficult	Easy if enough resources can be allocated in advance for each VC

3.5. Routing Algorithms

Routing Algorithms choose routes through the network, determining how a packet should be forwarded. In Virtual Circuit approaches this occurs once, is an ongoing process in Datagram networks.

3.5.1. Desirable Properties

- **Correctness**: algorithm finds routes to all destinations.
- **Simplicity**: algorithm executes quickly enough for Internet-scale routing.
- **Robustness**: must be resilient to lower-level hardware and software faults.
- **Stability**: should converge towards a common set of optimal paths
- **Fairness**: all parties using the network for communication should receive a 'fair' bandwidth allocation
- **Efficiency**: The routing algorithm should optimally exploit network resources.

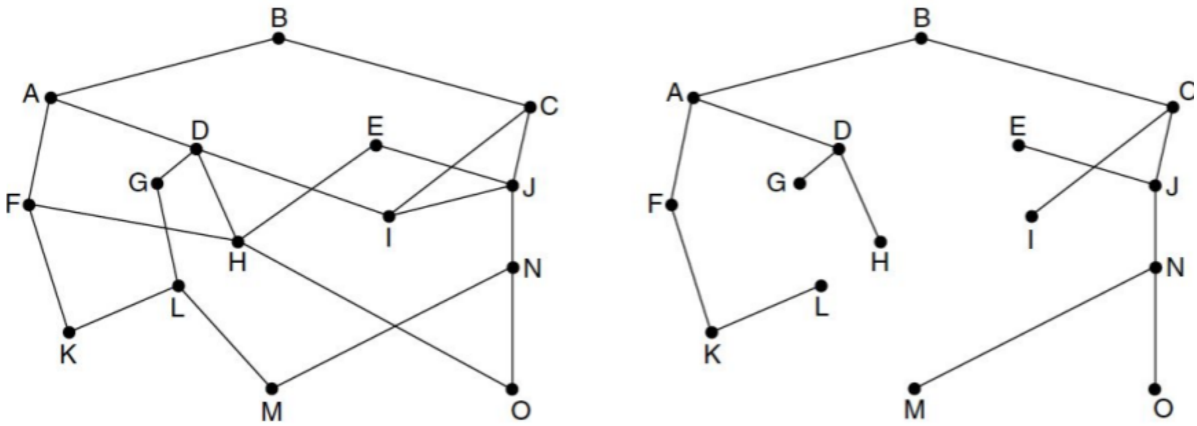
3.5.2. 2 Kinds of Algorithms

- **Non-adaptive Algorithms**: Do not consider current network topology or traffic
- **Adaptive Algorithms**: Change their routing decision to reflect changing topology and current traffic conditions.

3.5.3. The Sink Tree

The set of optimal routes from all sources to destination (D) form a sink tree routed at (D). Routing algorithms aim to discover and use the sink tree for all routers.

The Sink Tree (2/2)



Note: we will return to the issue of how to evaluate the cost of paths.

3.5.4. Shortest Path Definitions

Two common definitions:

- **Fewest Hop:** minimize the number of hops traveled between all pairs.
- **Lowest Cost:** minimize other cost factors on each path (distance, delay)

3.5.5. Dijkstra's Algorithm

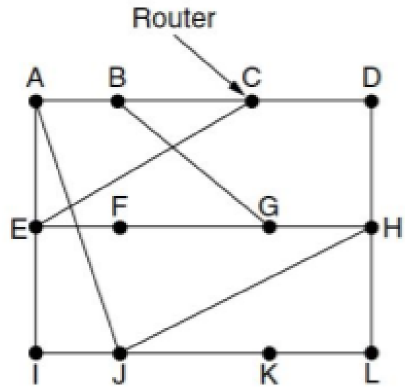
Not gonna explain this one, should know it by now

3.5.6. Bellman-Ford Algorithm

- Used in ARPANET and in the Internet as RIP
- Each router measures distance to each neighbor which is measured using special ECHO datagrams.
- Each router periodically exchanges its routing tables with all neighbors
- Upon receiving routing tables, current tables are discarded and new entries are created for each router based upon:
  - The lowest distance estimate received
  - The distance to the router providing that estimate

3.5.6.1. Example

Bellman-Ford in Action



To	A	I	H	K	New estimated delay from J	Line
A	0	24	20	21	8	A
B	12	36	31	28	20	A
C	25	18	19	36	28	I
D	40	27	8	24	20	H
E	14	7	30	22	17	I
F	23	20	19	40	30	I
G	18	31	6	31	18	H
H	17	20	0	19	12	H
I	21	0	14	22	10	I
J	9	11	7	10	0	–
K	24	22	22	0	6	K
L	29	33	9	9	15	K
	JA delay is 8	JI delay is 10	JH delay is 12	JK delay is 6	New routing table for J	

Vectors received from J's four neighbors

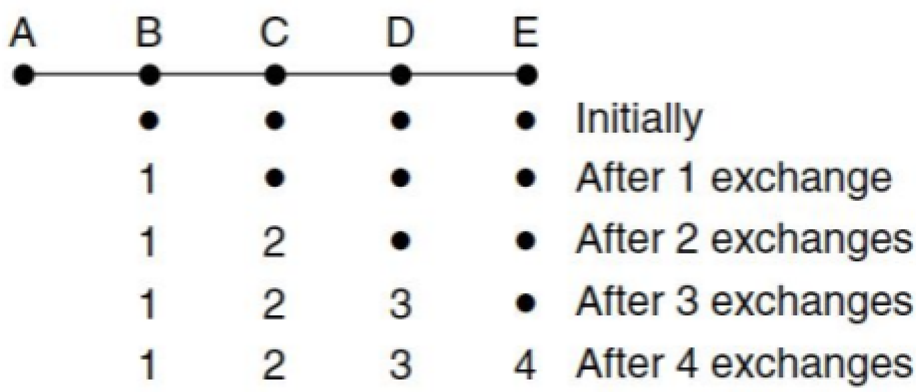
J's four neighbors

- **Question:** Assume that the delay from J for A=8,I=10,H=12 and K=6.
  - **Can you complete each table entry?**

The yellow table is made by using the previous table and the information given in the question.

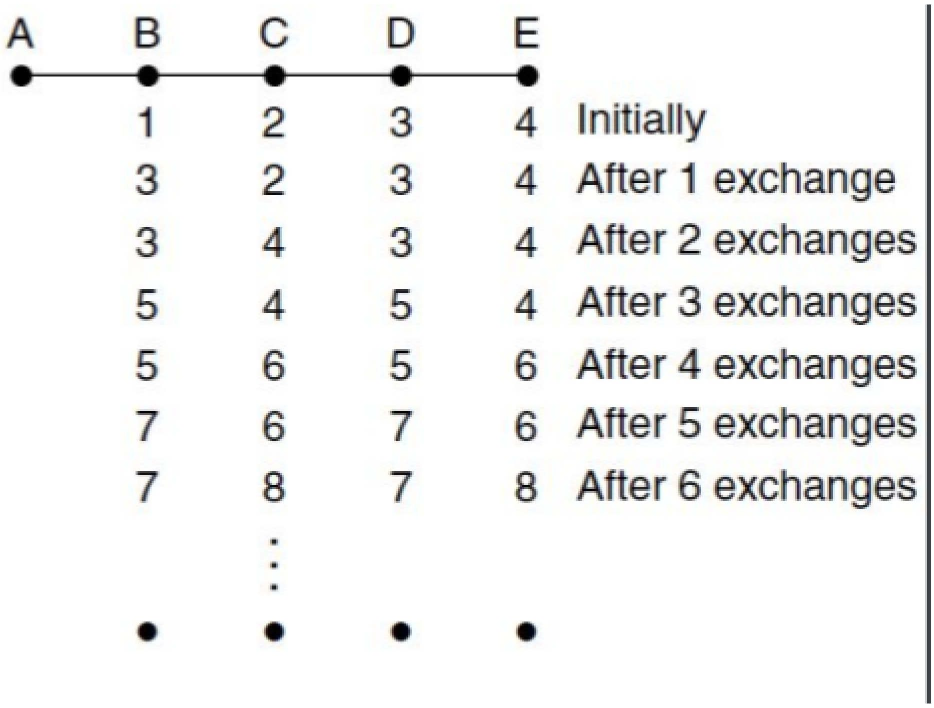
3.5.6.2. Count to Infinity problem

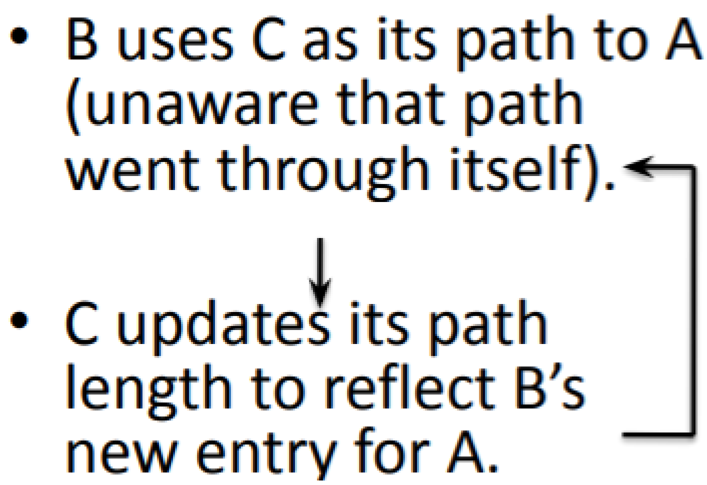
- Bellman-Ford uses only local knowledge. **Good news travel fast:**



- Imagine A is down. B-E have distances for A of infinity. A then comes online. After 4 exchanges, everyone has the new path to A.

- **Bad new travels slowly**. What if A goes down?





- **Note:** path length can only increase by 1 each round
- **Core problem** no router is aware if **itself** is included in a path

3.5.7. Link State Routing

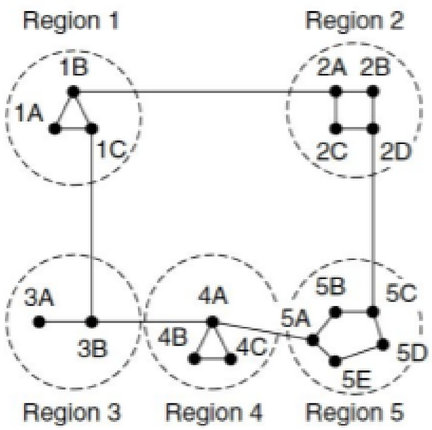
TODO

3.5.8. Hierarchical Routing

3.5.8.1. Principles

- Larger routing tables consume more resources
- At Internet scales it is not feasible for every router to have an entry for every other.
- The Hierarchy Solution:
    - Each router is assigned a region, it knows the internal structure of its own region
    - but only the entry point for remote regions

3.5.8.2. Example



Full table for 1A

Dest.	Line	Hops
1A	–	–
1B	1B	1
1C	1C	1
2A	1B	2
2B	1B	3
2C	1B	3
2D	1B	4
3A	1C	3
3B	1C	2
4A	1C	3
4B	1C	4
4C	1C	4
5A	1C	4
5B	1C	5
5C	1B	5
5D	1C	6
5E	1C	5

Hierarchical table for 1A

Dest.	Line	Hops
1A	–	–
1B	1B	1
1C	1C	1
2	1B	2
3	1C	2
4	1C	3
5	1C	4

TODO

3.6. Delivery Models

- **Unicast**: from a single address to a single address (common case considered previously)
- **Multicast**: one device on the network to many, but not all, devices on the network
- **Broadcast**: from one device on the network to all devices on the network
- **Unicast**: all messages from the sender to every receiver.
  - Very wasteful of bandwidth as routers often see the same packet twice.
  - Requires the sender to know the addresses of all end-points

3.6.1. Multi-destination Broadcast Routing

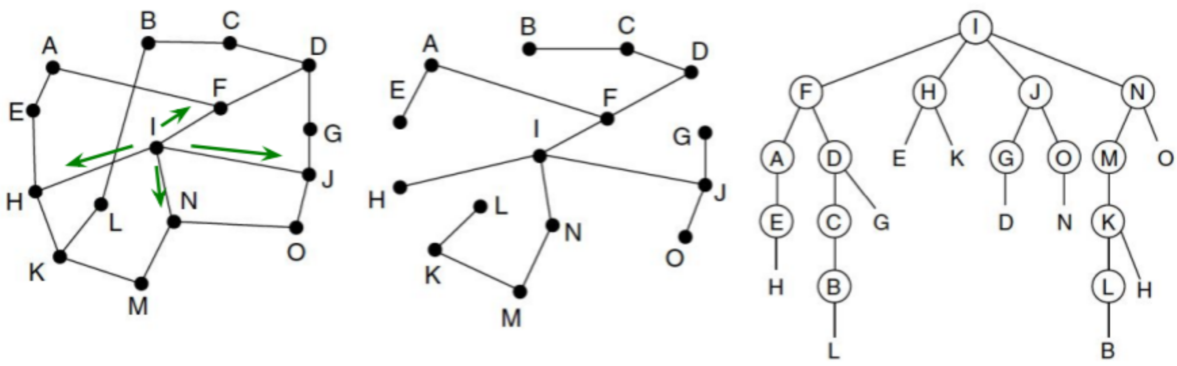
- Send a single message that contains a list of all destinations
- Each router checks the list of destinations and determines the output lines needed, generating a new copy on each line based on its routing tables.
- Lost of works for the routers
- Requires the sender to know the addresses of end-points

3.6.2. Flooding Broadcast

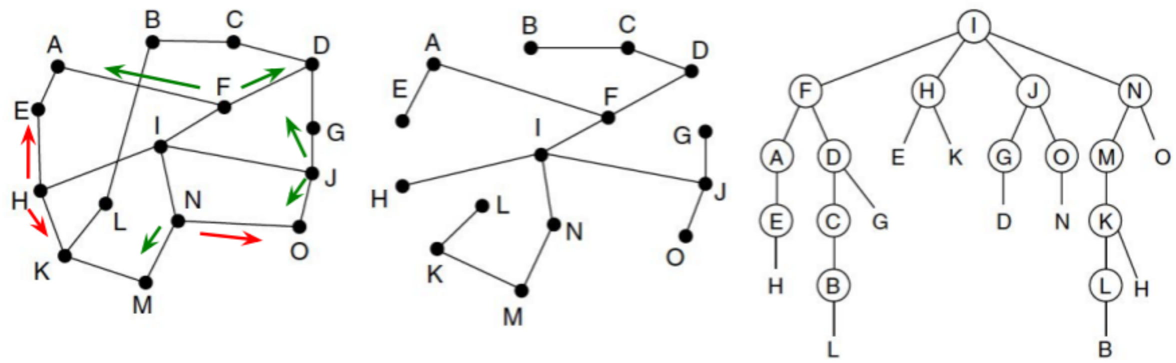
- Send a single message that should be re-transmitted by all routers
- Message contains a **Time-to-Live(TTL)** which is decremented at each host and a **sequence number ID**.
- Hosts forward all incoming messages on all links except the one they arrived one, discarding where TTL=0 or sequence number is in the cache.
- Still not the most efficient bandwidth use

3.6.3. Reverse Path Broadcast

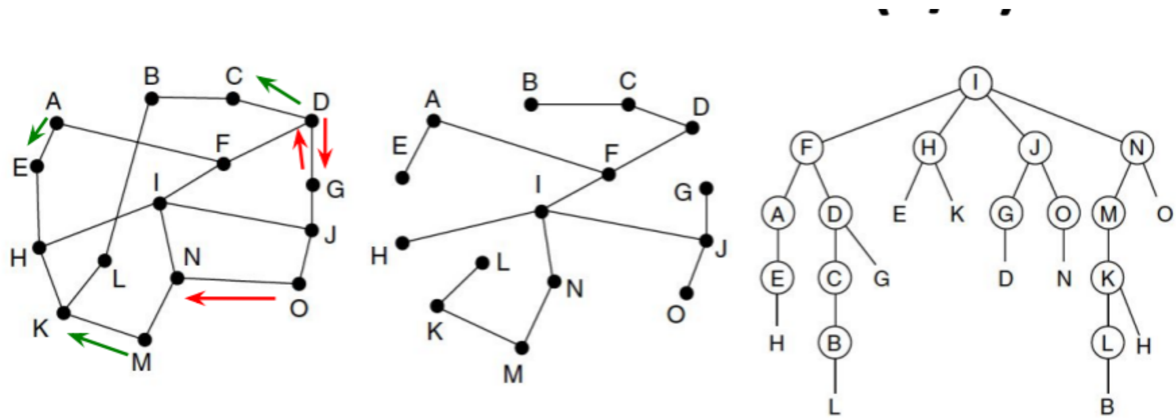
- Exploits the sink tree seen previously
- If a router receives a datagram on the link used to route to the send, it probably followed the shortest path and is first to arrive.
- • If the datagram arrives on any other link, discard it as a duplicate



- **Round 1:** I sends to {F,H,J,N}. All packets arrive along preferred path.



- **Round 2:** {F,H,J,N} generate 8 packets. 8 arrive at unvisited routers, 5 along the preferred line.



- **Round 2:** {A,D,G,O,M} generate 6 packets. 3 arrive along the preferred path and are forwarded on.

3.6.4. Spanning Tree Broadcast

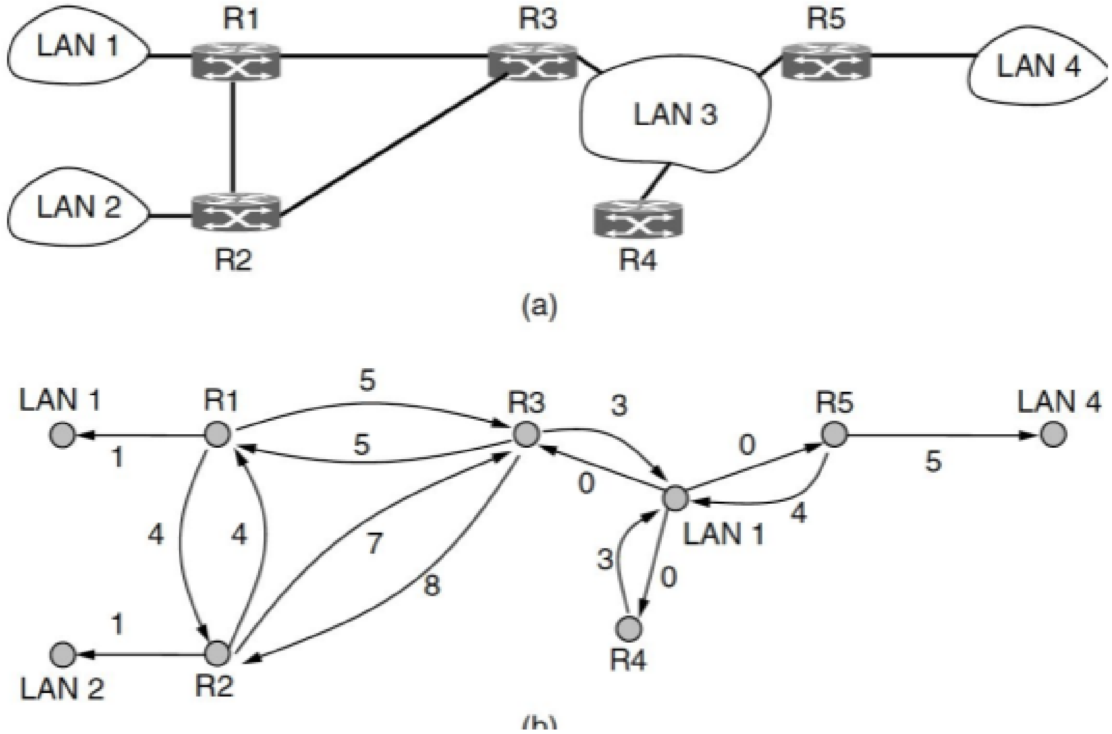
- A spanning tree is a subset of the network that includes all routers and no loops, as with the sink tree.
- Each router knows if it is in the spanning tree and will forward the packet on all spanning tree lines except the one it arrived on.
- Generates the minimum possible number of packets but is only possible where router have global knowledge, e.g. not possible with Distance Vector Routing approaches.

3.7. Open Shortest Path First (OSPF)

extension to Link State Routing

1. Discover its neighbours network addresses.
2. Set the distance to all neighbors.
3. Construct a packet with all learned.
4. Send this packet to all other routers.
5. Compute shortest path to all other routers using Dijkstra's Algorithm

OSPF Network Abstraction



Note: every link is represented twice, once for each direction as costs may be asymmetric

3.7.1. Load Balancing

- **Equal Cost MultiPath (ECMP)** routing:
  - Remembers sets of all equal length paths
  - Splits packets across them
- This allows for a simple kind of load balancing

3.7.2. OSPF Packets

Message type	Description
Hello	Used to discover who the neighbors are
Link state update	Provides the sender's costs to its neighbors
Link state ack	Acknowledges link state update
Database description	Announces which updates the sender has
Link state request	Requests information from the partner

- All OSPF messages are sent as standard IP packets.

3.8. Border Gateway Protocol (BGP)

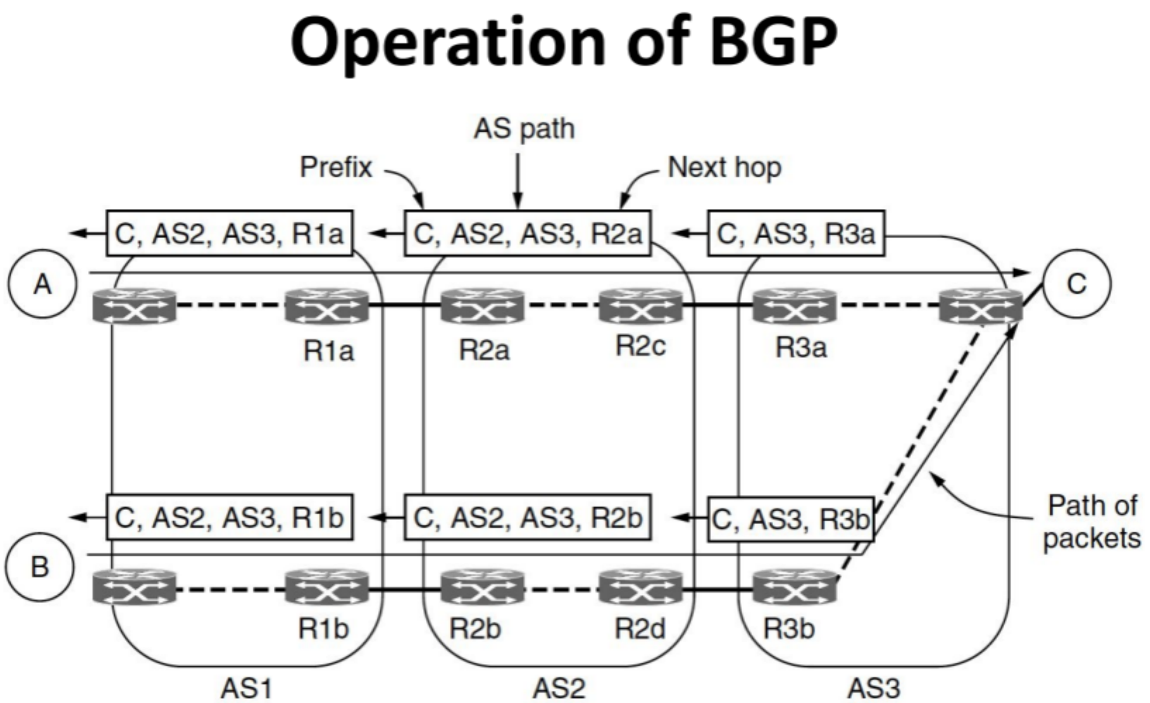
- Border Gateway Protocol focuses on supporting flexible routing policies
- AS care about more than shortest path
    - Do not carry commercial traffic on EDU network.
    - Never send Pentagon traffic through China.
    - Do not use AT&T due to bad performance.

3.8.1. Operation of BGP

- BGP is a form of distance vector protocol. However, it operates on paths not routers
- A path consists of the destination, the first hop router and then the sequence of AS
- Internal details of AS are obfuscated. They may use different routing protocols.

3.8.2. Internal BGP

- BGP routes are **not** propagated across AS using OSPF or other internal protocols
- Internal BGP (**IBGP**) requires boundary routers to learn routes of all other boundary routers
- Prefixes for external AS are stored at each boundary router and discovered internally using OSPF.



3.9. Mobile Connectivity

- The problem of keeping mobile hosts connected as they change location:
    - Truly mobile devices, e.g. cell phone
    - Nomadic devices, e.g. laptop computer
- We tackle this problem by providing a fixed home address for mobile devices
- This is better than re-computing routes, which at large scale ties too many care resources

3.10. Client Mobility

- Whenever you got to a new location, DHCP acquires you an address and DNS allows you to reach remote services

$\rightarrow$ This does not allow remote clients to remain connected to you

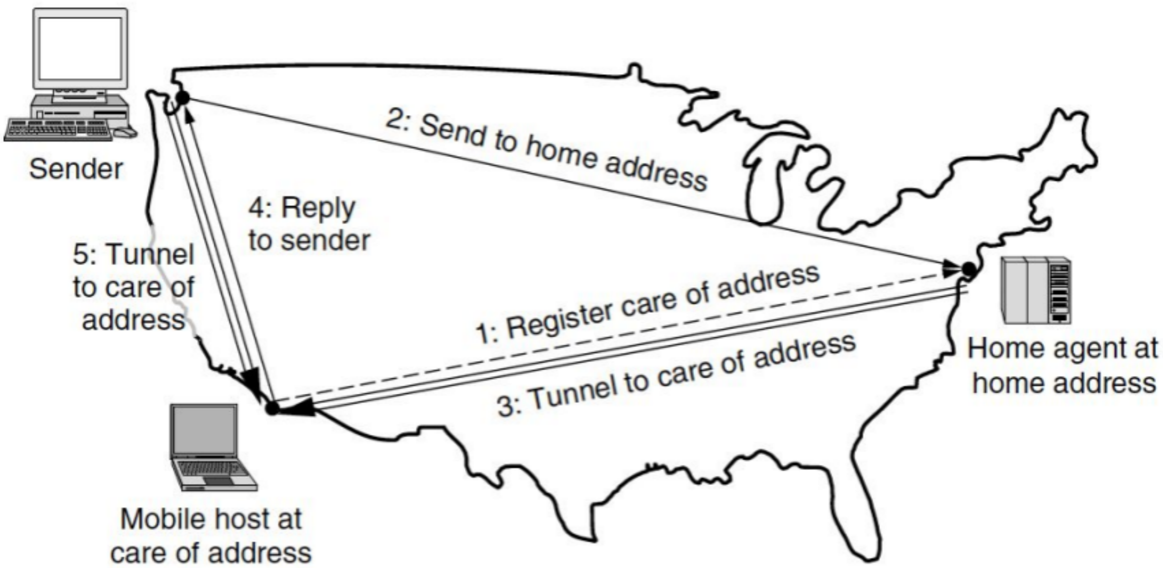
####

3.10.1. Why Network-level Mobility

- While application-level mobility is a neat solution, not everything uses domain names.
- If we want to be a full part of the Internet on a mobile host we need **network-layer mobility**

3.10.2. Network-Layer Mobility

- The home agent provides a single consistent address for mobile nodes.
- When a host acquires a new local address **(care-of-address)**, it report this to the home agent.
- When the home agent receives a datagram for the mobile host it is **encapsulated** with a new header and tunnelled to the care-of address
- Upon receiving a message from it home agent, the mobile host de-encapsulates it and responds directly
- Subsequent datagrams may be **tunnelled directly** between the remote host and the mobile host
- We refer to this as triangle routing. **Lets take a look at why..**



- Security if a big problem for this approach:
  - If I can imitate your mobile host, all of your traffic will be directed to me.

3.11. Ad-Hoc Networks

Ad-hoc networks are infrastructure-less. All nodes are both routers and hosts. This frequently occurs in V**ehicular Ad-hoc Networks (VANETs)** and **multi-hop Internet of Things Networks (IoT)**

All elements of the network may move at any time this static reasoning over topology is meaningless

3.11.1. IoT Concerns

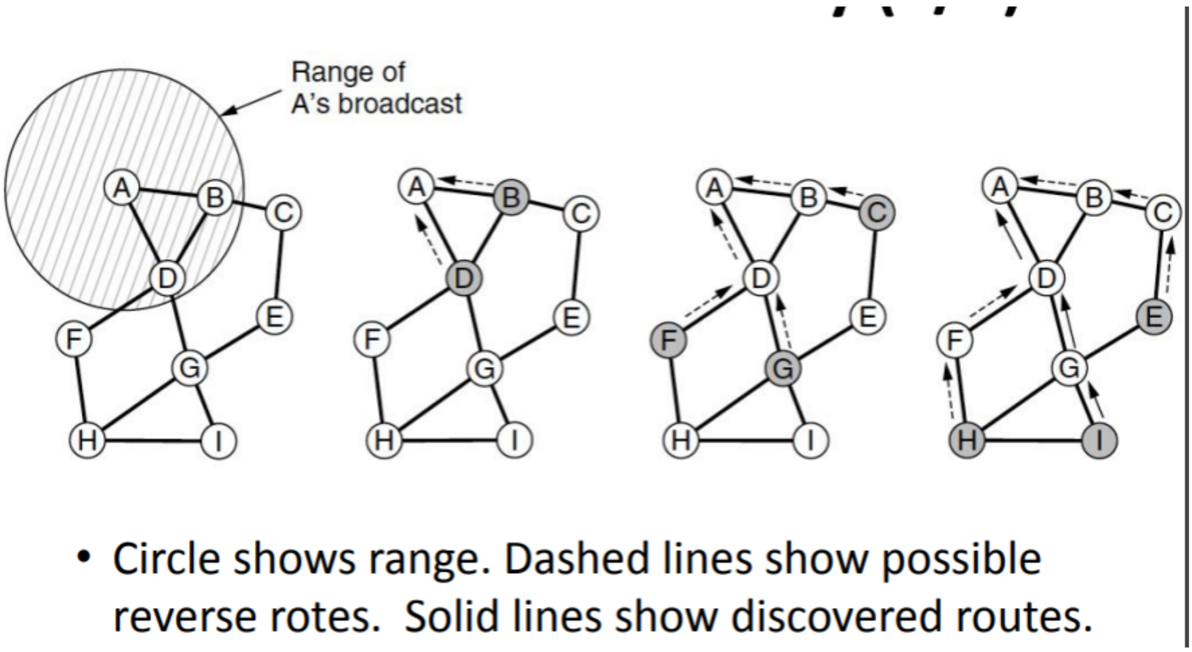
- **Resource constrained**: tiny memory, tiny storage
- **Energy conservation**: motes must run on a single charge for long periods
- **Unreliable**: mote failure due to damage, power failure, etc. is common

3.11.2. Ad-hoc On Demand Vector Routing

- **Ad-Hoc On Demand Vector Routing (AODV)** is closely related to distance vector routing.
- Routes are only discovered when someone wants to send a datagram -> **better for highly dynamic networks**

3.11.2.1. AODV: Route discovery

- Our network medium is broadcast. Each node has a range
- Neighbours determined by range + location. We assume an ideal circular range (unlikely)
- When **AODV** is asked to route a message to an unknown destination, it floods **ROUTE_REQUEST** message
- A **ROUTE_REQUEST** message contains a SEQ. Number and TTL to control flooding.
- Discovered node sends **ROUTE_REPLY** back along the path of incoming **ROUTE_REQUEST**
- A hop counter in ROUTE_REPLY is incremented at each intermediate hop and intermediate nodes add the route to their tables



As route discovery leads to high broadcast overhead. To minimize this... **ROUTE_REQUESTS** are initially sent with TTL=1 if this fails after a **time-out**, it this is incrementally increased until the route is found.

3.12. Internet Control and Message Protocol (ICMP)

3.12.1. Packets

Message type	Description
Destination unreachable	Packet could not be delivered
Time exceeded	Time to live field hit 0
Parameter problem	Invalid header field
Source quench	Choke packet
Redirect	Teach a router about geography
Echo and echo reply	Check if a machine is alive
Timestamp request/reply	Same as Echo, but with timestamp
Router advertisement/solicitation	Find a nearby router

3.12.2. Traceroute

Traceroute provides a map of how data on the internet travels from its source to is destination.

3.13. Address Resolution Protocol (ARP)

- Recall that we separate logical **IP addresses** and physical **MAC addresses.**
- Physical media routes by MAC address at each hop we need to resolve a IP to MAC addresses
- ARP maps IP addresses onto MAC addresses

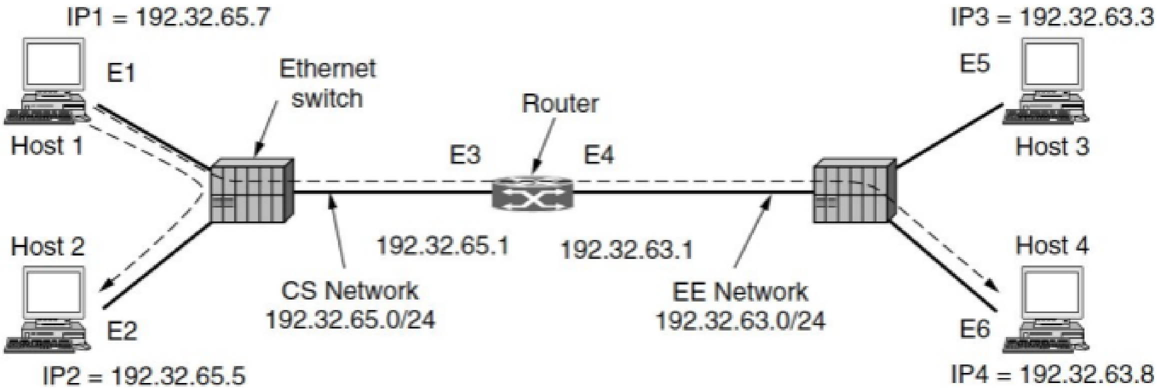
3.13.1. ARP in Action

Hosts 1 checks it ARP cache for IP address of Host 2, if it has a matching MAC address it sends to this address.

If MAC not in cache, Host 1 sends a broadcast message asking who owns the target IP address

Hosts2 reponds by sending it MAC address to Host 1. Host 1 may now route to Host 2.

Note: this is within the same subnet. What entry would you expect for hosts on remote links?



Frame	Source IP	Source Eth.	Destination IP	Destination Eth.
Host 1 to 2, on CS net	IP1	E1	IP2	E2
Host 1 to 4, on CS net	IP1	E1	IP4	E3
Host 1 to 4, on EE net	IP1	E4	IP4	E6

3.14. Throttling

- Recall: throttling occurs at the Transport Layer.
    - Implemented using a congestion window in TCP
- But it must be initiated by the Network Layer via implicit or explicit signals.
- We use three mechanisms: **Choke packets, Explicit Congestion Notification (ECN)** flags and **Random Early Detection (RED)**

3.14.1. Estimating Load

Load can be estimated based upon three metrics:

- **Link utilization**: rapid feedback, but hard to account for burst traffic.
- **Packet loss:** accounts for all traffic but usually comes too late
- **Queuing delay**: rapid feedback including burstiness

3.15. Choke Packets

Choke packets are explicit notifications provided by congested routers.

- A datagram is sent to the sender
- The packet is marked so that no further choke packets will be generated later on

Choke packet causes sending entity to throttle the rate it sends segments into the network

3.16. Explicit Congestion Notification (ECN)

ECN is used to tag packets as they travel through routers. ECN-tagged packets reach **destination host**, which messages the sender to throttle send rate.

-> In comparison to Choke Packet, ECN has **lower overhead**, but takes **longer** to enact **change**.

3.17. Random Early Detection

We randomly discard packets as delay is detected.

This is counter-intuitive, but has advantages:

- It makes packet loss a more reliable congestion signal.
- It reduces the delay between congestion starting and transport-layer throttling.
- Random Early Detected (RED) will drop random packets as congestion approaches.

Question:

- Why is randomly dropping packets a good way to signal fast senders?

  A: Because fast senders tend to have many more packets in the queue… – …so randomly dropping packets tends to affect them more. – This also reduces book-keeping overhead.

4. Chapter 4: Data Link Layer

4.1. MAC layer principles (Medium Access Control)

4.1.1. The Channel Allocation Problem

- Medium Access Control (MAC) is about sharing a single broadcast channel
- This may be part of the wireless spectrum, a cable or optical fiber
- There are two basic options: static allocation and dynamic allocation

4.1.2. Static Channel Allocation

- Traditional approach to sharing network media between multiple nodes.
- For example, FM radio channels are statically allocate to radio stations.

4.1.3. Dividing Network Media

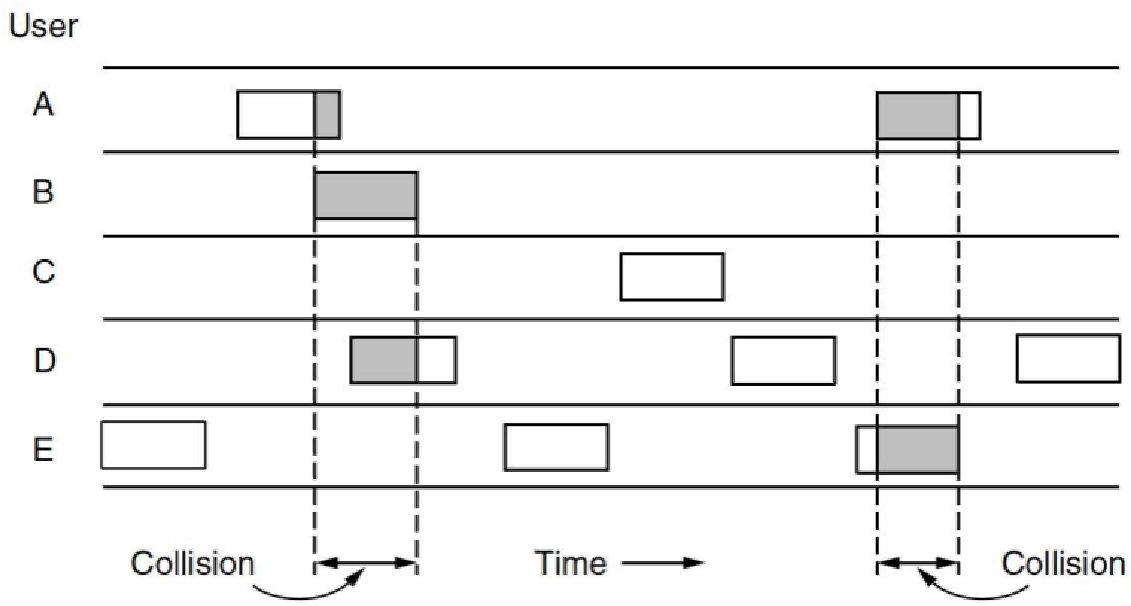
- **Different media**: n wires from N in a cable
- **Time slicing**: n time-slots from N in a given time period.
- **Frequency division**: n channels from N in a given time period

4.1.4. Dynamic Assignment Assumptions

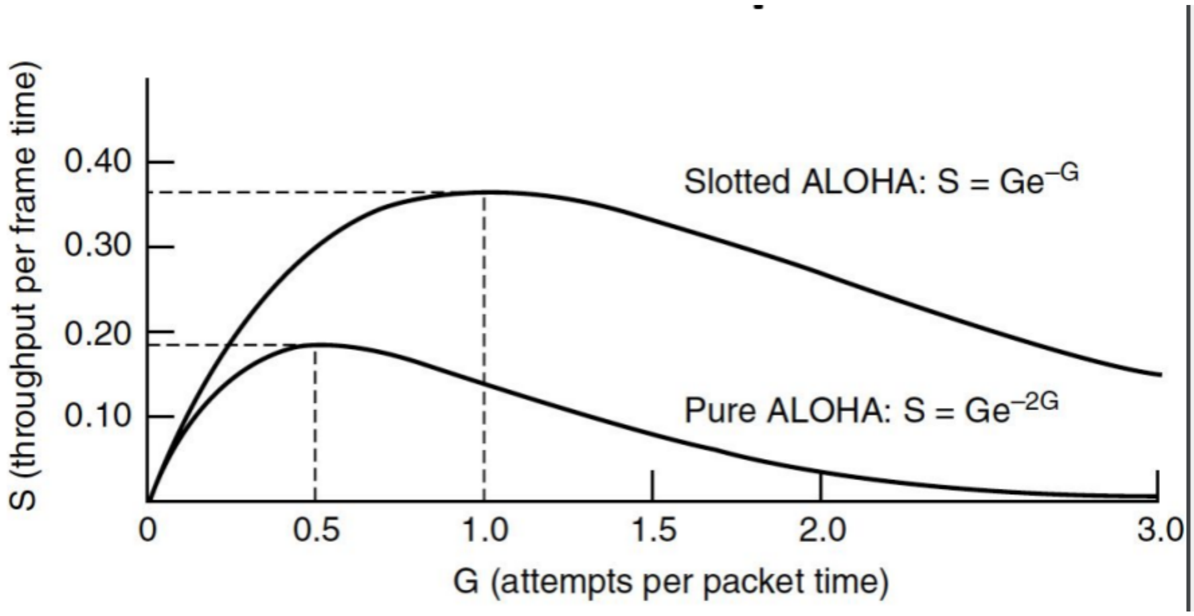
1. **Independent traffic** - multiple nodes generating constant traffic levels.
2. **Single channel or not** - do all nodes send and receive on the same channel?
3. **Observable collisions** - when two nodes transmit at once, the result is garbage
4. **Continuous or slotted** - are transmissions constrained to periods or free?
5. **Carrier sense or not** - can we tell if the channel is in use before sending?

4.2. Case studies Important TakeAways

4.2.1. Slotted



As you can see on the image, a lot of times collision happen because hosts are arbitrarily sending frames. If we now use timeframes in which only certain hosts can transmit, we can reduce the amount of collisions and thus performance and can be seen below.



4.2.2. Carrier Sense Multiple Access (CSMA)

In essence, just listen on the network and check if anyone is transmitting. If yes, do nothing. Listening all of the time is known as **persistent** CSMA.

4.2.3. non-Persistent Carrier Sense Multiple Access (CSMA)

This type of CSMA does not sample continuously. When a node has data ready, it samples the channel. If the channel is idle it transmits directly. If it is busy, it waits a random period and tries again. If a collision occurs, the node waits a random time and retransmits.

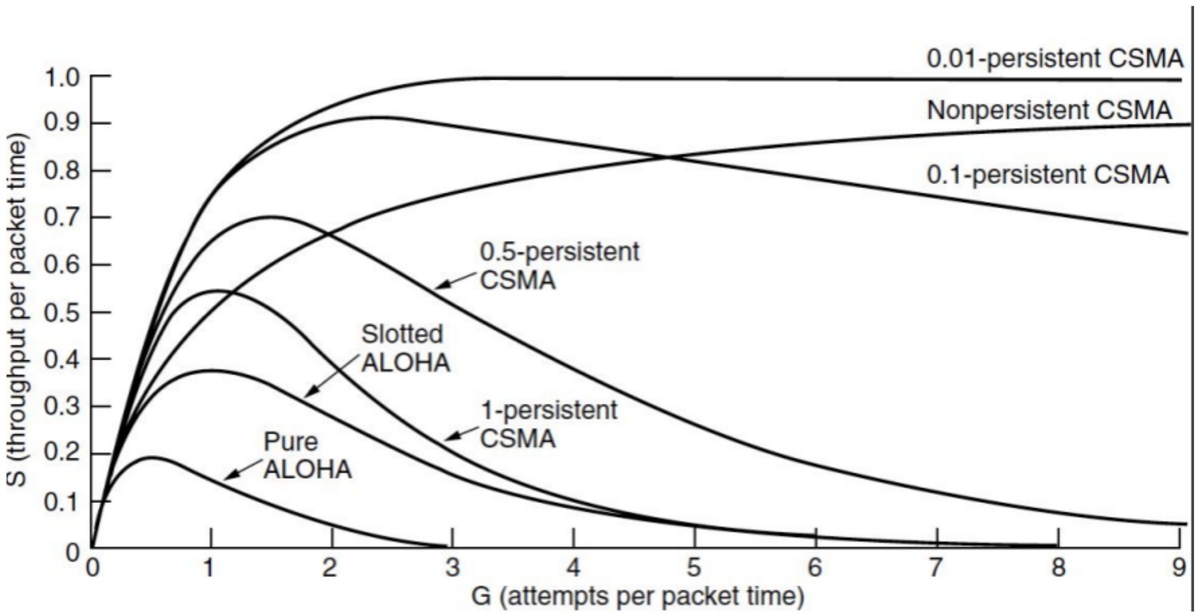
4.2.4. 1-Persistent Carrier Sense Multiple Access (CSMA)

Same thing as before but now we transmit with a probability of 1 when channel is idle.

4.2.5. p-Persistent Carrier Sense Multiple Access (CSMA)

Same thing as before but now we transmit with a probability of p when the channel is idle. If no acknowledgement it retransmits with probability Q = 1-p, and so on until transmit

Performance Comparison



4.2.6. Hidden and Exposed Terminals

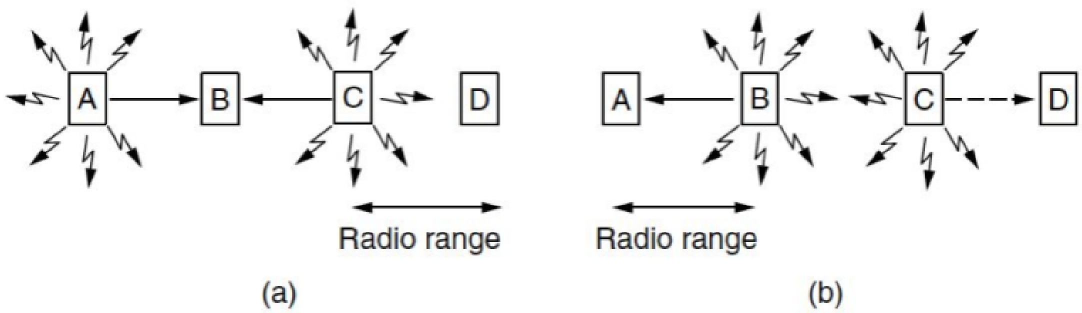


Figure 4-11. A wireless LAN. (a) *A* and *C* are hidden terminals when transmitting to *B*. (b) *B* and *C* are exposed terminals when transmitting to *A* and *D*.

4.2.7. IoT Challenges

RF Interference and occlusion:

- Only small chinks of the spectrum may be used without a license and are therefore highly congested
- Sporadic interference must be anticipated

Power constraints:

- The radio uses more power than any other component, so we must minimize its use (while providing ad-hoc routing)
- Remember it uses power while transmitting and listening

Node Loss:

- We may lose nodes at any time, monitoring, flood prediction, volcano monitoring.
- Or just simple dead batteries

→ Solution: Berkeley MAC (BMAC)

4.2.7.1. Berkeley MAC

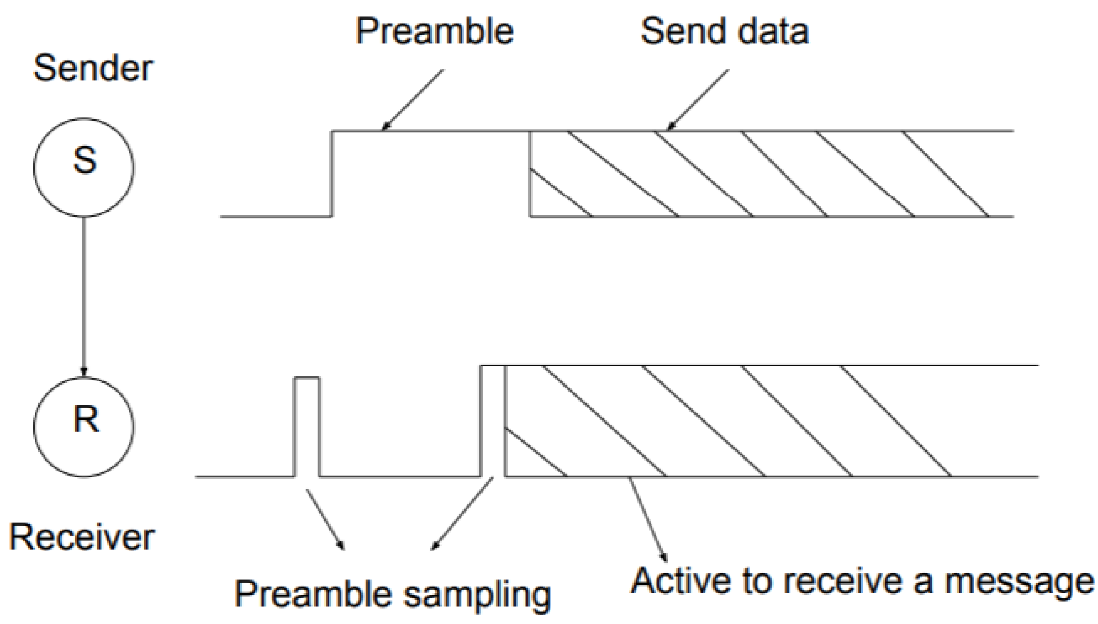
- Target requirements:
  - Low-power, simple implementation, configurable, scalable, decentralized.
  - Suitable for diverse applications

Based upon a simple Carrier Sense Medium Access scheme.

As we do not know when our neighbours may transmit, we must listen continuously. How can we reduce the power cost of this?

As transmissions are infrequent, it is better to spend a little more energy when transmitting than while listening. This is the idea behind *Low Power Listening*

Nodes only turn on their radios for short sampling periods. If traffic is detected, the radio remains active and listening to receive a packet. As transmissions may occur at any time, we add a long preamble to messages to ensure that the radio detects the incoming message and is active by the time the data arrives.



$$|\text{Preamble}| \geq \text{Sampling period}$$

4.2.8. Time Synchronized Mesh Protocol

TSMP is a MAC and Network layer protocol for WSAN.

Target requirements:

- Reliable, low power and secure, self-organizing, self-healing, minimal management.
- It should 'just work'.

TSMP (Time Synchronized Mesh Protocol) is a networking protocol that forms the foundation of reliable, ultra low-power wireless sensor networking. Wireless sensor networks (WSNs) are self-organizing, multi-hop networks of wireless sensor nodes used to monitor and control physical phenomena. Typical WSN applications include industrial process automation, commercial building climate control and security alarming.

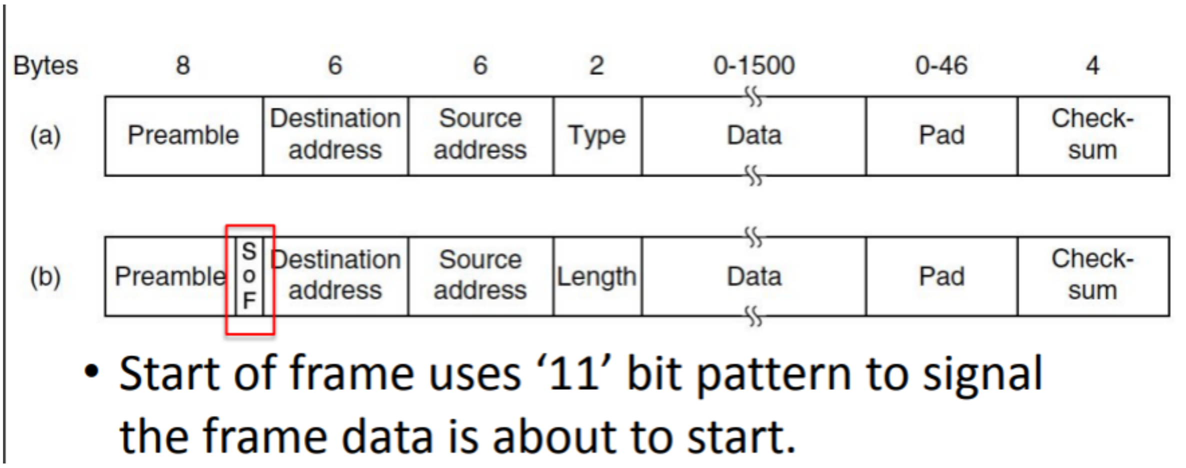
TSMP provides redundancy and fail-over in time, frequency and space to ensure very high reliability even in the most challenging radio environments. TSMP also provides the intelligence required for self-organizing, self-healing mesh routing. The result is a network that installs easily with no specialized wireless expertise, automatically adapts to unforeseen challenges, and can be extended as needed without sophisticated planning.

There are five key components of TSMP that contribute to end-to-end network reliability, simple installation and power efficiency.

- Time synchronized communication
- Frequency hopping
- Automatic node joining and network formation
- Fully-redundant mesh routing
- Secure message transfer

4.3. Ethernet

4.3.1. Frame Formats



4.3.2. DIX v IEEE 802.3

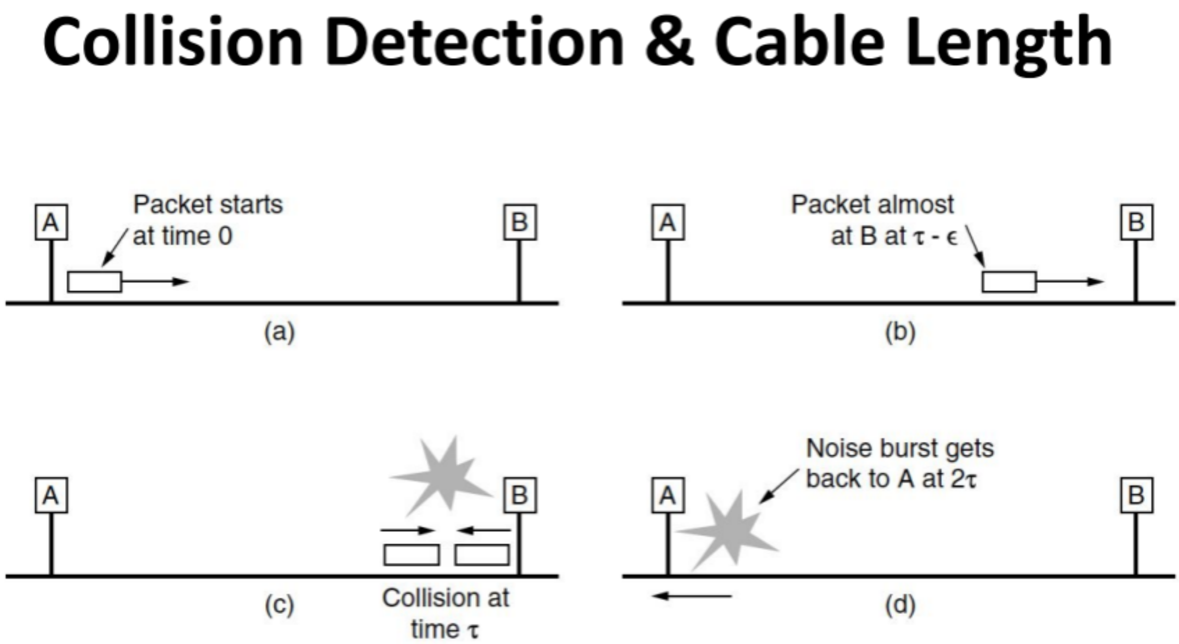
- DIX uses a length field, while 802.3 uses a type field. As can be seen above.
- Without the type field you cannot dispatch to higher layers, so an 8 byte logical link control header is added.
- Fortunately all of the type fields had values over 1500:
  - Ethernet max frame is 1500
  - <= 1500 denotes size. > 1500 denotes type

4.3.3. Min. / Max. Frame Length

Transceiver must have enough RAM for MAX frame size. RAM was expensive, so 1500 bytes was selected.

Why a minimum frame size?

- A minimum frame size of 64bytes makes it easy to distinguish valid packets from failed chunks
- On collision, Ethernet generates a 48 bit noise burst to warn other stations
- On a long cable, a small packet may be transmitted before collision is detected.



4.3.4. Ethernet applies 1-p CSMA-CD (Carrier Sense Medium Access with Collision avoidance)

- Ethernet uses 1-persistant CSMA-CD:
  - Sample channel continuously
  - If busy, wait until idle and transmit with probability 1
  - If a collision occurs wait a random time and retransmit
- Ethernet uses binary exponential back-off rather than waiting a random time.

4.3.4.1. Exponential Back-off

Back-off is slotted. After collision, each station waits a random number chosen in the range 0 to 2^i-1, where i is the number of collisions. We try: 0-1, 0-3, 0-7

Range is upper bounded to 1023 slots: after 16 collisions higher layers are signaled.

Algorithm ensures lowest delay when collisions are rare and adapt to increasing collisions.

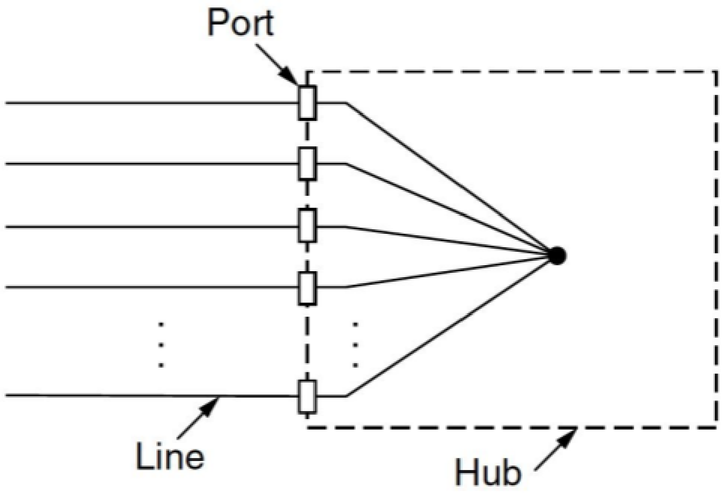
4.3.5. Security Issues

With persistent channel sampling, an Ethernet interface sees frame for all hosts on its link. Normally these frames would be thrown away. But the host is free to keep them and analyse them.

4.3.6. From a single Cable to Hubs

####

4.3.7.

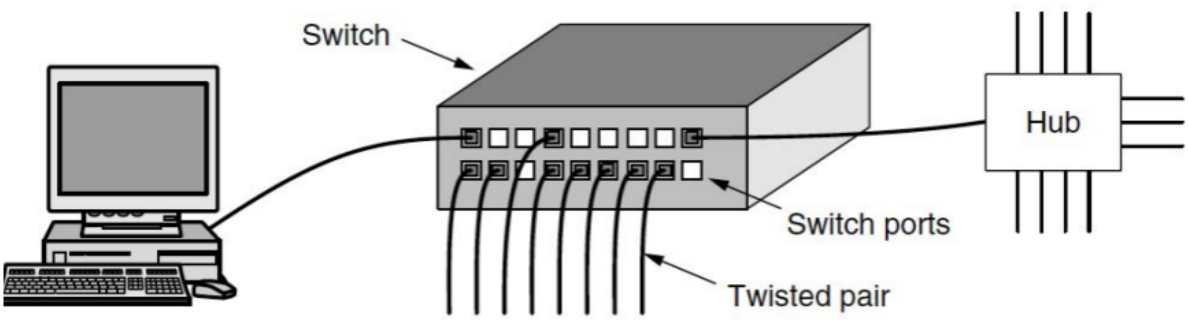


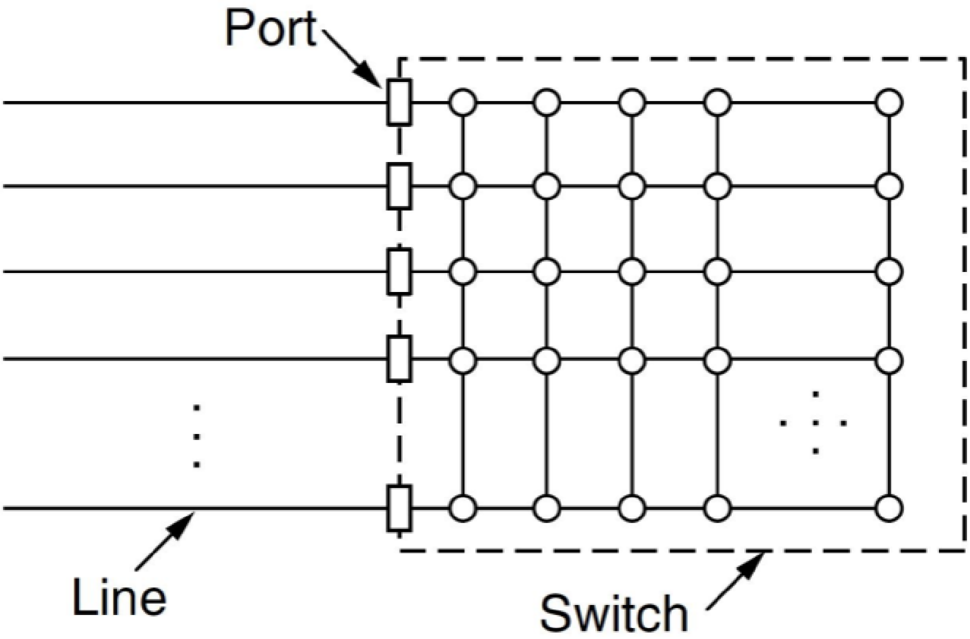
It makes it easier to debug networks. If a link is broken, it affects one host.

4.3.8. Motivation for Switch Ethernet

- Hubs do not increase capacity and eventually the LAN will saturate.
- Switched Ethernet uses a high-speed backplane to connect hosts on a single line

Switched Ethernet Setup





4.3.9. Switched Ethernet

- Switches only output frames to ports for which they are destined
- Each cable is usually full duplex, so the switch and host can talk at the same time
- Each port is thus a single collision domain that only needs to run the CSMA/CD protocol if:
    - it has multiple hosts connected via a hub
    - the cable is half-duplex.
- Switches improve performance as all hosts can send packets at the same time, **without CSMA/CD**
- Capacity is improved as frames are only deliverd to hosts that they are intended for
- The impact of promiscuous mode is limited.
- As two input ports may try to deliver frames to the same output port, the switch needs buffers, making it more expensive.

4.3.10. Fast Ethernet

- IEEE reconvened the 802.3 group in 1992 to create a faster 100mbps standard.
- Two competing proposals
    - Keep the protocol the same, but run it faster
    - Add lots of features

  -> The first proposal won
- Requires twisted pair wiring

4.3.11. 1-Gigabit Ethernet

- Gigabit Ethernet was standardized in 1999:
    - Goal 1: 10 fold increase in speed
    - Goal 2: backwards compatibility
- Gigabit Ethernet does not allow interconnection by hubs

- In the case of connection to a switch on a duplex cable CSMA is not used, thus cable lengths are only limited by signal strength
- Half duplex operation requires CSMA, limiting cable lengths to 25m - this proved problematic
- Two hardware features were added:
    - Carrier extension, which pads all packets to 512 bytes
    - Frame bursting, which concatenates small frames into a single > 512 byte transmission.

4.3.12. 10-Gigabit Ethernet

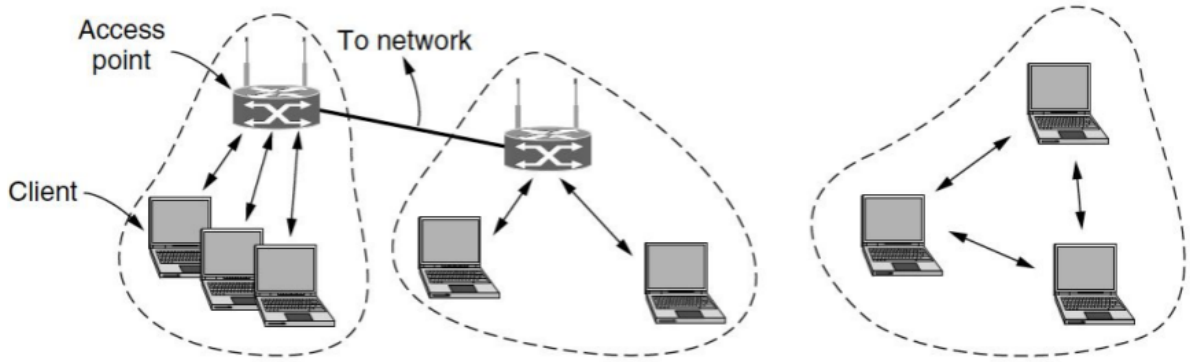
- Follows the same approach as 1 gigabit Ethernet: increase speeds using the same approach.
- To eliminate length restrictions, only full-duplex operation is allowed and hubs are not used.
- Transmission over copper is difficult (4 pairs of UTP). Transmission over optical fibre is preferred.

4.4. IEEE 802.11

Two modes of operation:

- **Infrastructure mode**: used by wired access points to provide access to wireless hosts.
- **Ad-hoc mode**: used to create a network on-the-fly between wireless hosts

802.11 Architecture



4.5. Wireless Challenges

RF Interference:

- Only small chunks of the spectrum may be used without a license and are therefore highly congested
- Sporadic interference must be anticipated

Blocked Paths:

- 2.4GHz signals are blocked by metal, absorbed by water and scattered foliage.
- Optical signals are blocked by any opaque object.

Bandwidth Limitations:

- Low-power networking typically has an order of magnitude lower bandwidth than standard WiFi
- Maximum packet sizes may be incompatible with other communication media

Mobility:

- Nodes may move; necessitating protocols that can cope with a dynamic network topology.

Power constraints:

- Yes, even on your laptop we must conserve power.

4.6. Virtual Channel Sensing

In virtual channel sensing, we use higher level information to infer if we need to check the channel again. We do this by using **Network Allocation Vectors (NAV)**. Each frame carries a **NAV field** that say how long the transmission sequence will take to complete.

→Now everyone will wait for this sequence to finish before trying to transmit

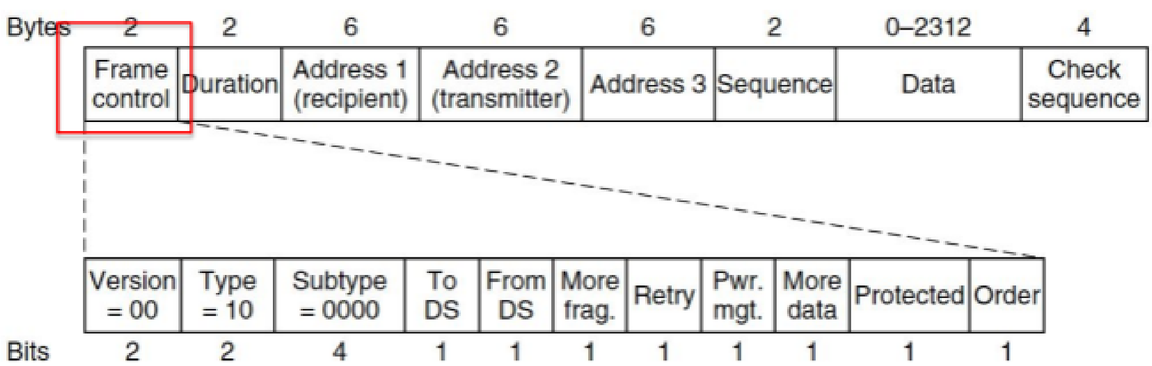
4.6.1. Power Saving

First way

- First approach is based on **periodic broadcasts** from the AP (beacon frames)
- Used to advertise to the client that the AP has data ready for them
- In-between beacon frames the client sleeps

Second way

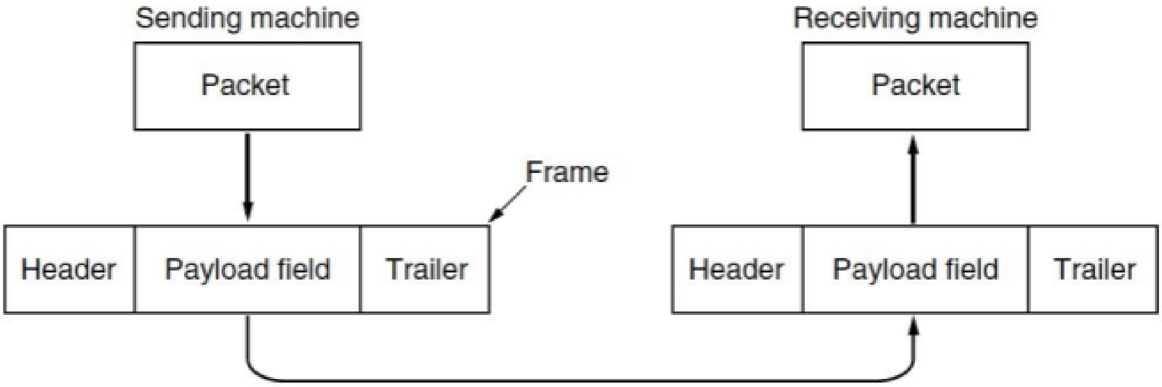
- This approach uses client transmissions for timing
- The AP buffers traffic for a client until the client transmits, then sends it immediately after the ACK
- The client can sleep until it needs to transmit



4.7. The Data Link Layer

- Provides a well-defined software service interface to the physical media
- Handles transmission error
- Regulates the flow of data so that slow receivers are not swamped by fast senders
- Contains the Medium Access Control sub-layer.

4.7.1. Packets and Frames



4.7.2. Types of Data Link Layer

- **Unacknowledged connectionless service.**
  - E.g. Ethernet, appropriate for media with inherently low error rates or when collisions are detectable.
- **Acknowledged connectionless service.**
  - E.g. 802.11, appropriate for unreliable channels and when collisions are undetectable.
- **Acknowledged connection-oriented service.**
  - E.g., Telephone or serial line, provides a reliable bit stream.

4.8. Framing

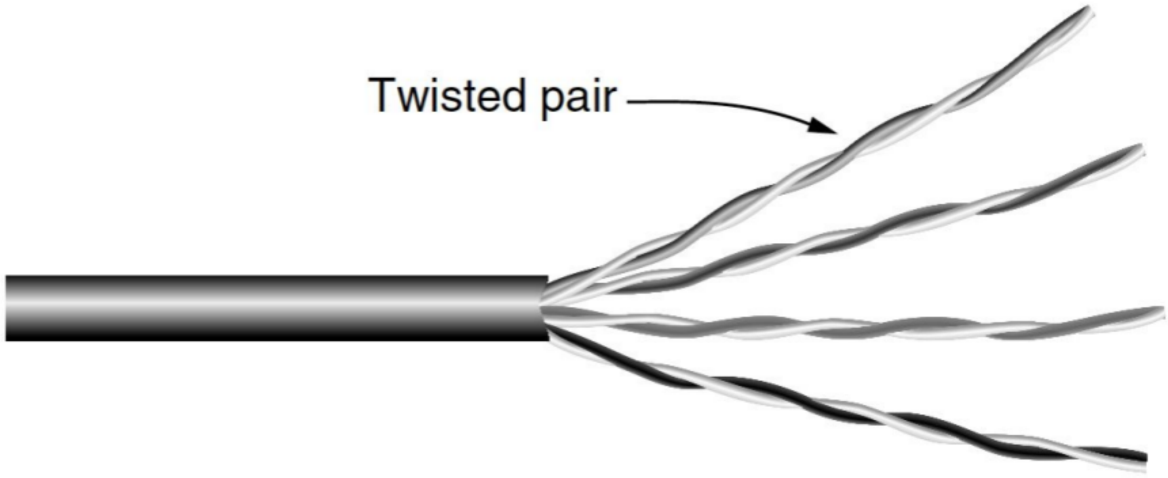
When the datalink layer receives data from the **physical** layer it may contains **errors**. Some bits may have different values and the number of bits received may be less than, equal to, or more than the number of bits transmitted. It is up to the data link layer to detect and, if necessary, **correct** errors. The data link layer usually does this by **breaking the bit stream up into discrete frame** and compute the checksum. But this is **harder** then it seems.

4 Methods:

1. **Byte count**: appends a field to the start of the frame giving its length but it doesn't help as we don't know where the next frame starts.
2. **Flag bytes with byte stuffing**: Flag bytes are special values used to mark the start and end of each frame
   - What if the flag byte occurs in the data? We just add an escape byte just before each "accidental" flag byte in the data.
3. **Flag bits with bit stuffing**: Same thing as before, just add a a special bit pattern. What if the escape bit occurs in the data?Imagine a flag byte of 01111110. Whenever the sender detects a sequence of five ones. It stuffs a zero in to make 011111010.
4. **Physical layer coding violations**

5. Select Physical Layer Topics

5.1. Twisted Pairs



Cables must be twisted, as a single cable is essentially an antenna: radiating power and absorbing interference.

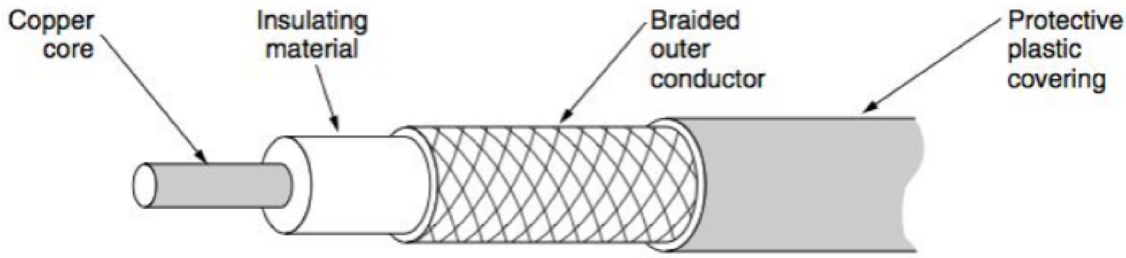
5.1.1. Category 5 Wiring (Cat5)

Category 5 wiring is an example of twisted pair. Each cable contains **fours pairs of wires gently twisted together**.

Full duplex links require a separate upstream and downstream wire, half duplex links do not

Offers limited bandwidth. Cat 6 and 7 add extra shielding and performance.

5.2. Coaxial cable



Better shielding but lower bandwidth

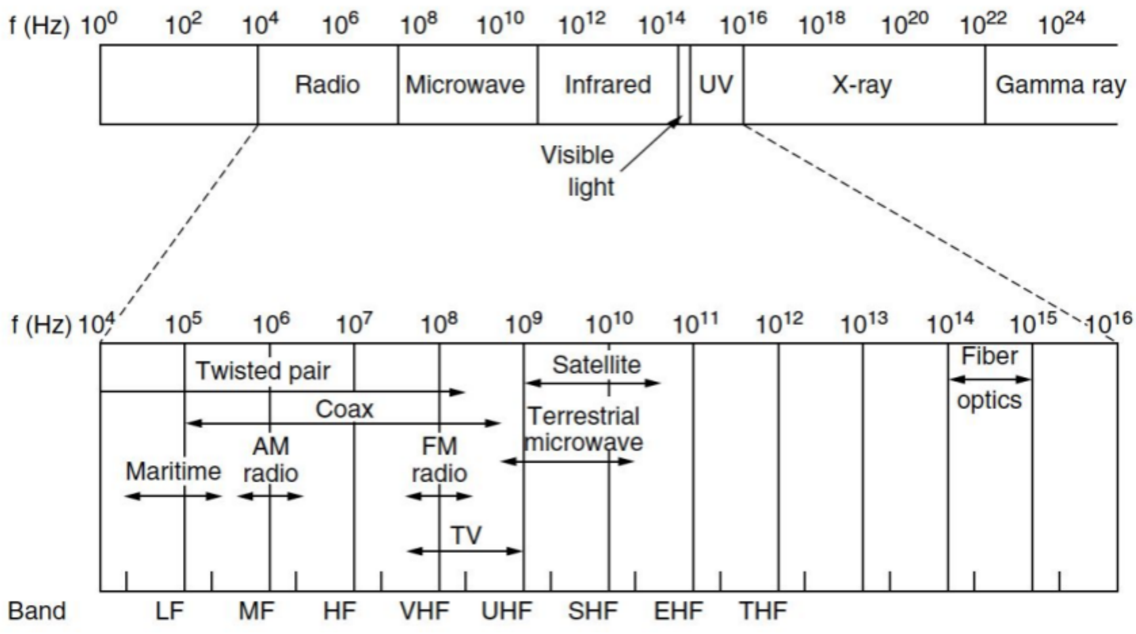
5.3. Optical Fiber

Three Part System: light source, transmission medium and detector.

Because of a media as thin as a few light wave-lengths act as a wave-guide allowing light to move in only one direction.

Can transmit at 100mbps over 100km without amplification.

5.4. Wireless Communication



5.4.1. Microwave communication

Not very important

Compared to lower frequency EM radiation, they are high bandwidth, but do not travel through solid objects well. Using an omni. antenna microwaves can evenly cover an area, or coverage can made asymmetric using a directional antenna.

Microwaves are absorbed by water molecules. So practically anything standing in between ruins it. All microwave transmitters are to some extent also serving as a microwave oven for all water containing objects in-between.