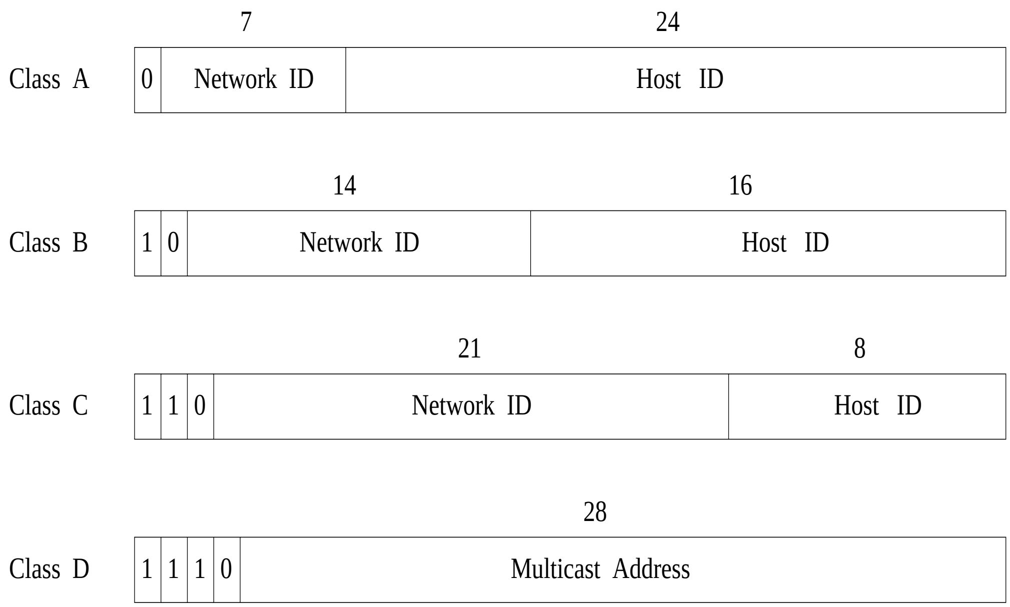
IP address format:



Dotted decimal notation: 10000000 00001011 00000011 00011111 $\leftrightarrow$ 128.11.3.31

Symbolic name to IP address translation: domain name server (DNS).

Hierarchical organization: 2-level

$\longrightarrow$ network and host

Each interface (NIU) has an IP address; single host can have multiple IP addresses.

$\longrightarrow$ single-homed vs. multi-homed

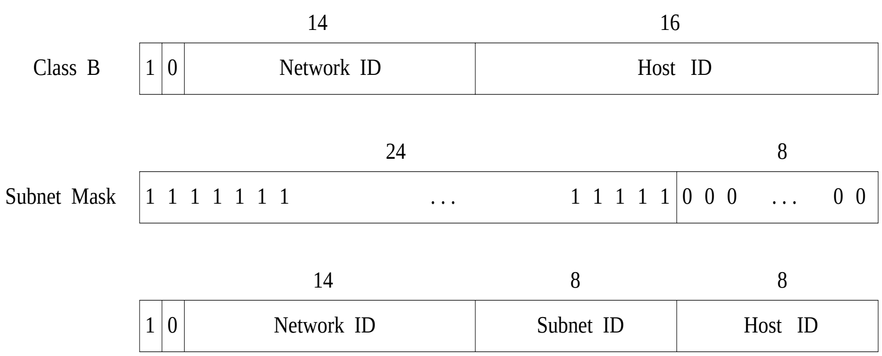
Running out of addresses…

Waste of address space:

⟶ typical organization: network of networks

⟶ not too many hosts (class B: 64K)

Solution: subnetting—subdivide host ID into subnetwork ID and host ID



To determine subnet ID:

- AND IP address and subnet mask

  → already know if class A, B, C, or D

- 3-level hierarchy

Forwarding and address resolution:

Subnet ID	Subnet Mask	Next Hop
128.10.2.0	255.255.255.0	Interface 0
128.10.3.0	255.255.255.0	Interface 1
128.10.4.0	255.255.255.0	128.10.4.250

Either destination host is connected on a shared LAN, or not (additional IP hop needed).

$\longrightarrow$ reachable by LAN address forwarding

$\longrightarrow$ if not, network address (IP) forwarding

Table look-up I ("where to"):

- For each entry, compute $SubnetID = DestAddr$ AND $SubnetMask$.
- Compare $SubnetID$ with $SubnetID$.
- Take forwarding action (LAN or IP).

Remaining task: translate destination or next hop IP address into LAN address

$\longrightarrow$ must be done in either case

$\longrightarrow$ address resolution protocol (ARP)

Table look-up II ("what's your LAN name"):

- If ARP table contains entry, using LAN address link layer can take over forwarding task.

  → ultimately everything is LAN

  → network layer: virtual

- If ARP table does not contain entry, broadcast ARP Request packet with destination IP address.

  → e.g., Ethernet broadcast address (all 1's)

- Upon receiving ARP response, update ARP table.

Dynamically maintain ARP table: use timer for each entry (15 min) to invalidate entries.

⟶ aging (old caching technique)

Other approaches to solve address depletion problem:

- IPv6

  → 128 bits (who wants it?)

- classless (vs. classful) IP addressing

  → variable length subnetting

  → $a.b.c.d/x$ (x: mask length)

  → e.g., 128.10.0.0/16, 128.210.0.0/16, 204.52.32.0/20

  → used in inter-domain routing

  → CIDR (classless inter-domain routing)

  → de facto Internet addressing standard

- dynamically assigned IP addresses

  → reusable

  → e.g., DHCP (dynamic host configuration protocol)

  → used by access ISPs, enterprises, etc.

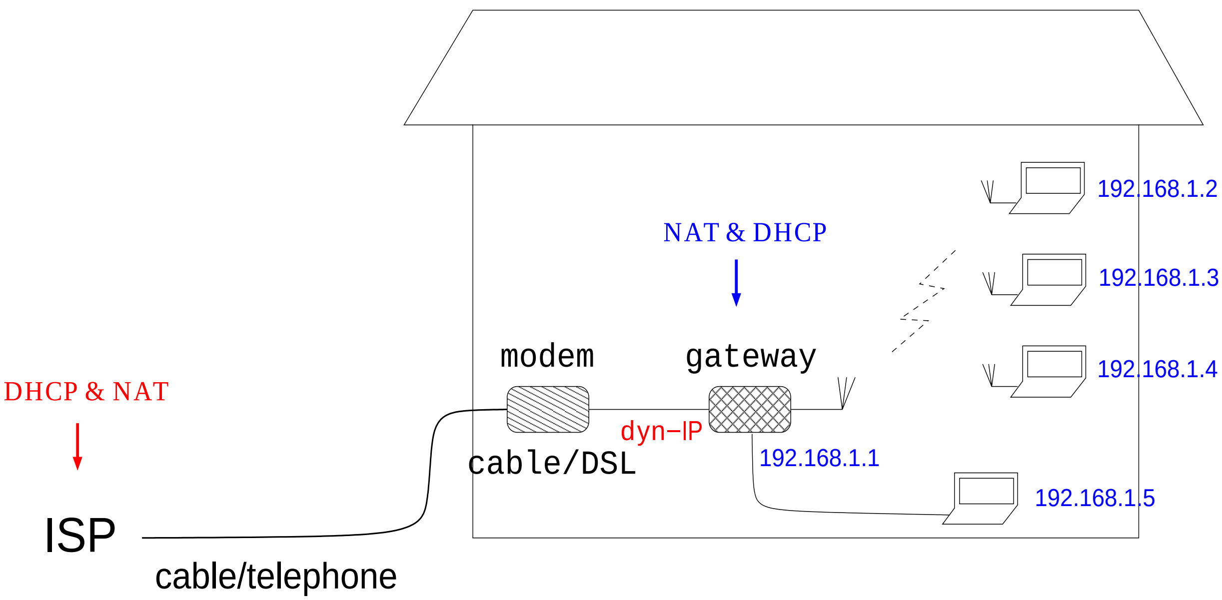
  → specifics: network address translation (NAT)

  → private/unregistered vs. public/registered IP address

  → can additionally use port numbers: NAPT

Ex.: SOHO (small office/home office)

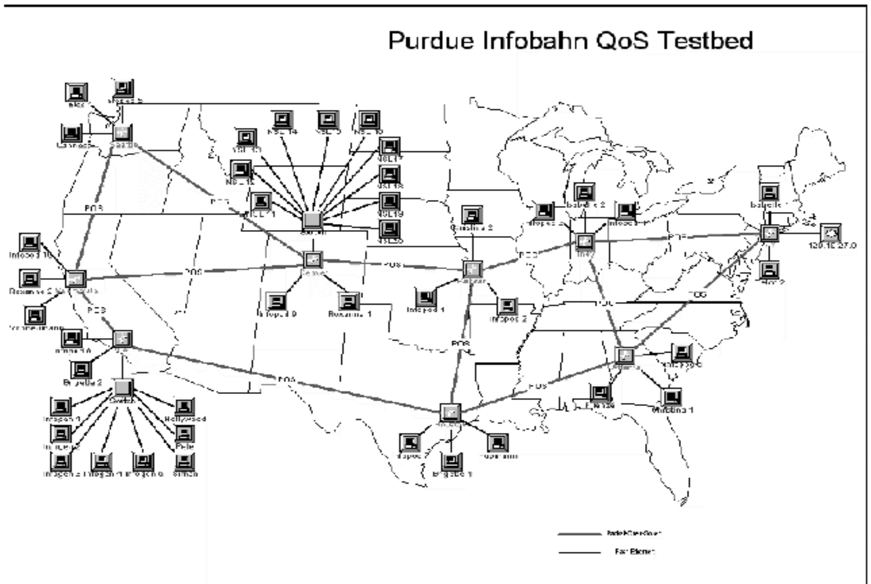
⟶ now: home networking

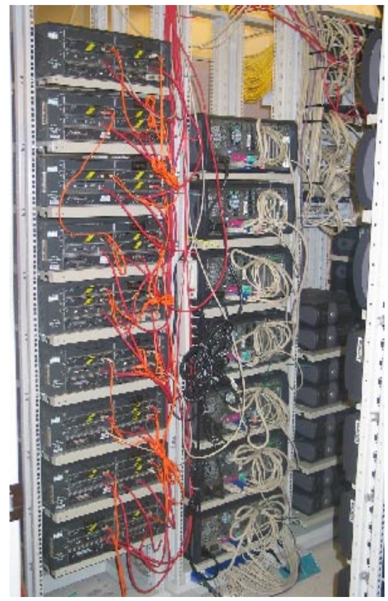


- dynamic IP address provided by ISP is shared through NAT
- IANA (Internet Assigned Numbers Authority)

  → non-routable: e.g., 192.168.0.0/16, 10.0.0.0/8

Ex.: private backbone or testbed (e.g., Q-Bahn)





- routers have 10.0.0.0/8 addresses

  → each interface: a separate subnet

- only one of the routers connected to Internet

  → 128.10.27.0/24 address

- PCs connected to routers are dual-homed

  → 10.0.0.0/8 address & 128.10.27.0/24 address

  → dual-homed forwarding

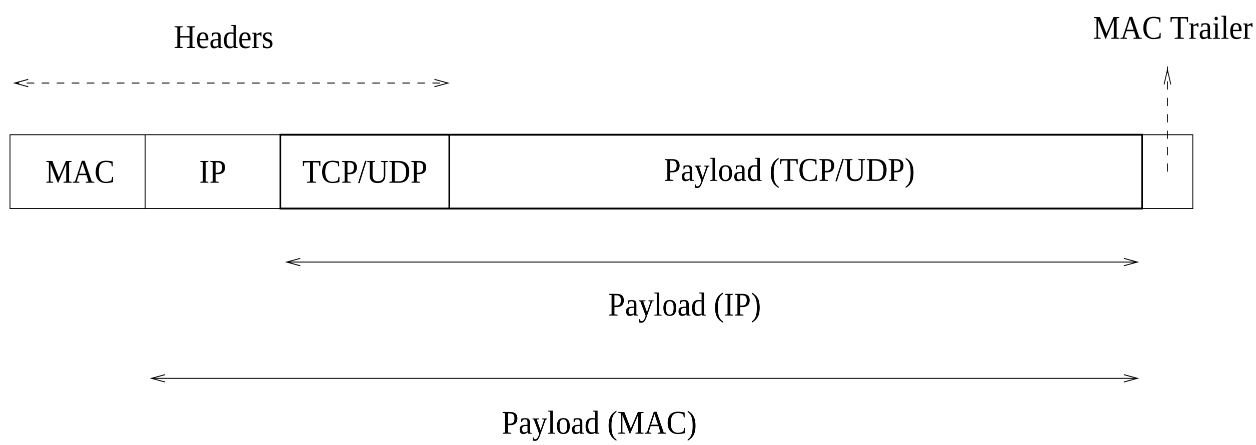
Transport Protocols: TCP and UDP

$\longrightarrow$ end-to-end protocol

$\longrightarrow$ runs on top of network layer protocols

$\longrightarrow$ treat network layer & below as black box

Three-level encapsulation:



$\longrightarrow$ common TCP payload: HTTP

Network layer (IP) assumptions:

- unreliable
- out-of-order delivery (not frequent)
- absence of QoS guarantees (delay, throughput, etc.)
- insecure (IPv4)

  → IPsec

Additional (informal) performance properties:

- Works "fine" under low load conditions
- Can break down under high load conditions

  → Atlanta Olympics

  → DoS attack

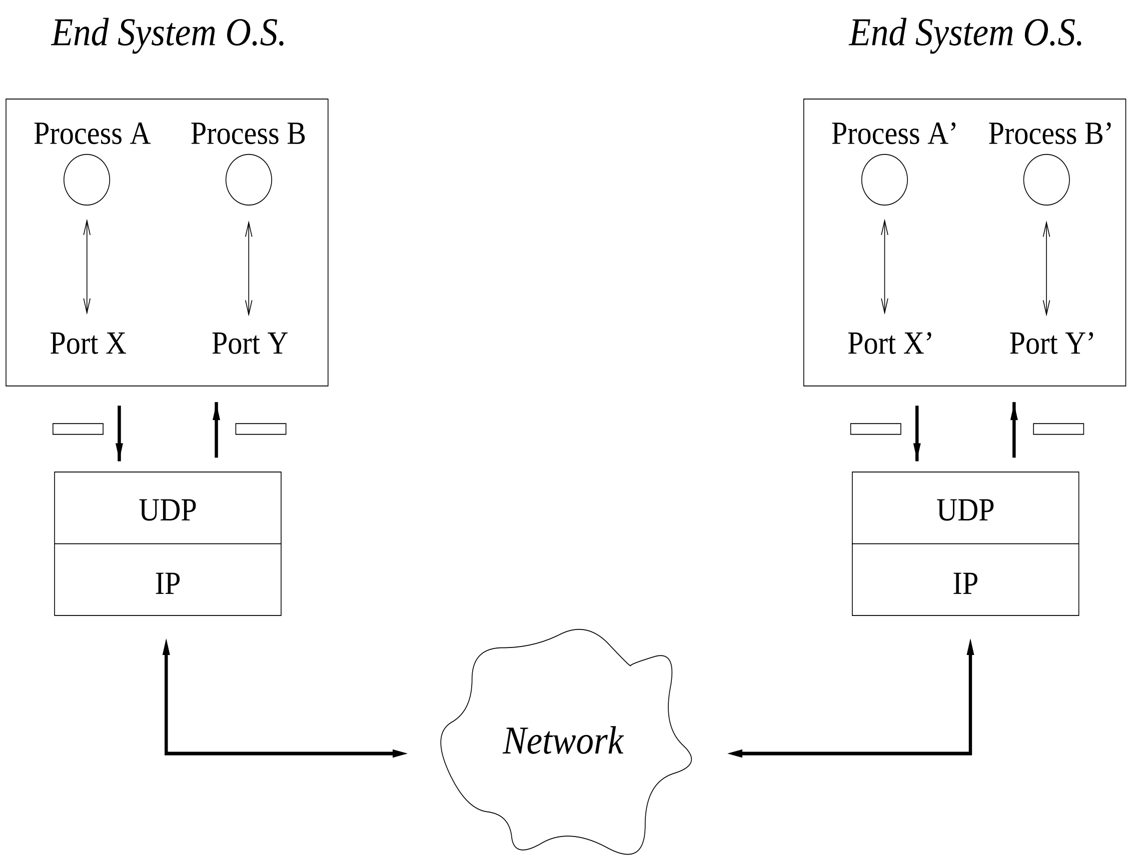
- Wide behavioral range: to some extent predictable

Goal of UDP (User Datagram Protocol):

⟶ process identification

⟶ port number as demux key

⟶ minimal support beyond IP



UDP packet format:

2	2
Source Port	Destination Port
Length	Checksum
Payload	

Checksum calculation (pseudo header):

4		
Source Address		
Destination Address		
$00 \cdots 0$	Protocol	UDP Length

$\longrightarrow$ pseudo header, UDP header and payload

UDP usage:

- Multimedia streamining

  → lean and nimble

  → at minimum requires process identification

  → congestion control carried out above UDP

- Stateless client/server applications

  → persistent state a hinderance

  → lightweight

Goals of TCP (Transmission Control Protocol):

- process identification
- reliable communication: ARQ
- speedy communication: congestion control
- segmentation

$\longrightarrow$ connection-oriented, i.e., stateful

$\longrightarrow$ complex mixture of functionalities

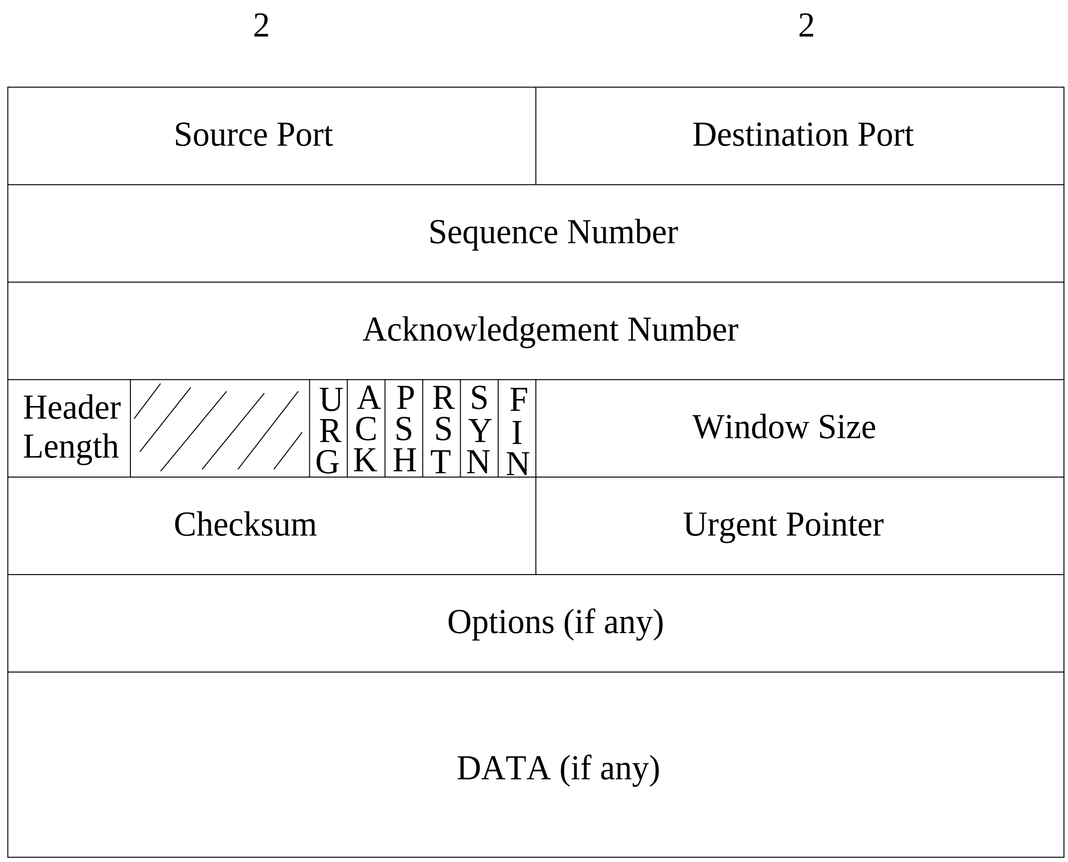
Segmentation task: provide "stream" interface to higher level protocols

$\longrightarrow$ exported semantics: contiguous byte stream

$\longrightarrow$ recall ARQ

- segment stream of bytes into blocks of fixed size
- segment size determined by TCP MTU (Maximum Transmission Unit)
- actual unit of transmission in ARQ

TCP packet format:



- Sequence Number: position of first byte of payload
- Acknowledgement: next byte of data expected (receiver)
- Header Length (4 bits): 4 B units
- URG: urgent pointer flag
- ACK: ACK packet flag
- PSH: override TCP buffering
- RST: reset connection
- SYN: establish connection
- FIN: close connection
- Window Size: receiver's advertised window size
- Checksum: prepend pseudo-header
- Urgent Pointer: byte offset in current payload where urgent data begins
- Options: MTU; take min of sender & receiver (default 556 B)

Checksum calculation (pseudo header):

4

Source Address		
Destination Address		
00 ··· 0	Protocol	TCP Segment Length

$\longrightarrow$ pseudo header, TCP header and payload