End-to-End Communication

Goal: Interconnect multiple LANs.

Why?

- Diverse LANs speak different languages

  → need to make them talk to each other

- Need management flexibility

  → global vs. local Internet

  → administrative policy barriers

Problems:

- How to choose paths (routing)?
- How to regulate flow (congestion control)?

  → not too much, not too little

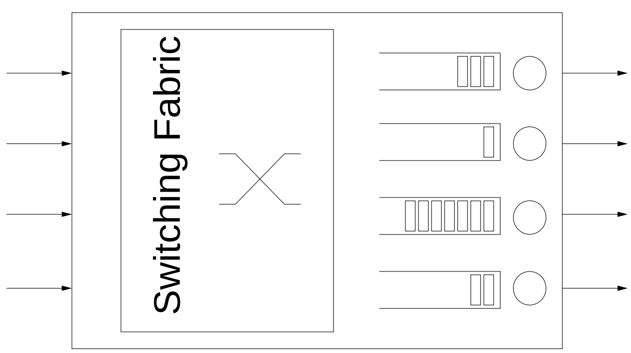
- How to provide service quality (QoS control)?

Packet Switching vs. Circuit Switching

Router/switch design:

- Hardware: ASIC
- Software: fast PC as router or gateway
- Hybrid: network processor

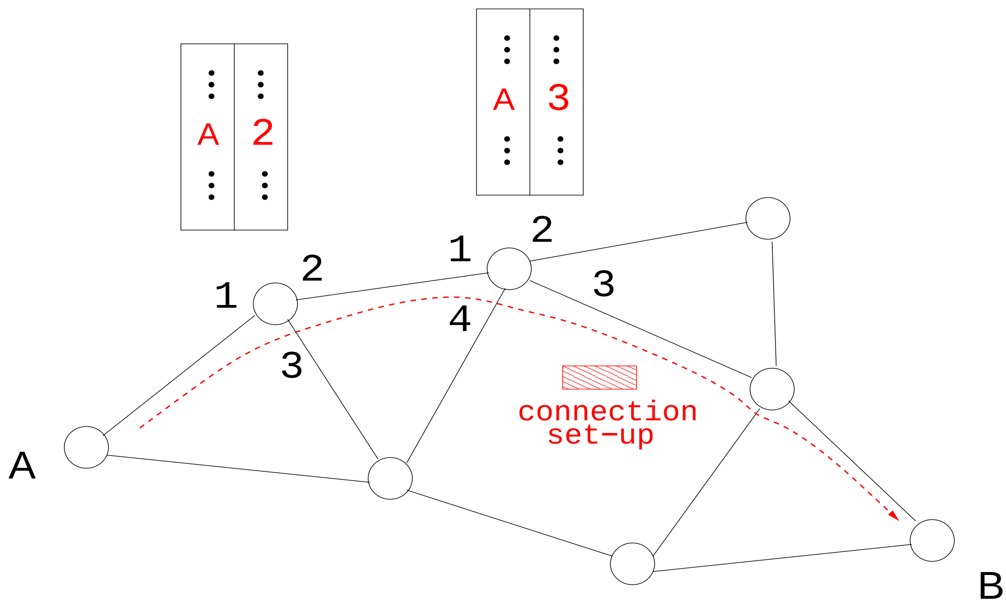
  $\rightarrow$ programmable



$\longrightarrow$ fast vs. slow forwarding path

$\longrightarrow$ interconnection network

Circuit-switched forwarding:



- connection set-up message: signaling

  $\longrightarrow$ how to: routing subsystem

  $\longrightarrow$ different from forwarding subsystem

- source tag "A" inserted into look-up table

  $\longrightarrow$ on-demand, compact look-up table

  $\longrightarrow$ deletion upon termination

  $\longrightarrow$ tag: VPI (virtual path identifier)

Packet-switched forwarding:

$\longrightarrow$ dispense with connection set-up signaling

$\longrightarrow$ each packet: autonomous entity

Source routing:

- packet contains path information

  $\rightarrow \langle A, C, \ldots, B \rangle$
- drawback: header length increases with path length

  $\rightarrow$ not good for fast packet handling

  $\rightarrow$ why?

Destination-based forwarding:

- determine output port by destination address
- source address ignored

$\longrightarrow$ same destination, same path: at any node

$\longrightarrow$ Internet packet switching

Internet Protocol (IP)

Goals:

- interconnect diverse LANs into one logical entity
- implement best-effort service

  → no assurances ("what you get is what you get")

  → simplicity

Represents:

- common language for carrying out non-LAN-specific conversations

  → technical definition of **I**nternet

- functionality and design philosophy

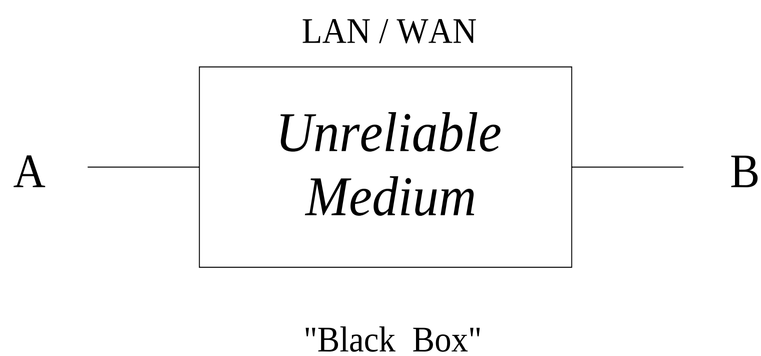
  → simple core / complex edge

  → end-to-end paradigm

Best-effort vs. reliable service:

- simplifies router design but increases complexity of end stations
- necessitates higher-up functional layer to achieve reliable transmission over unreliable medium

  $\rightarrow$ e.g., implement ARQ at sender/receiver



Best-effort vs. guaranteed service:

$\longrightarrow$ router must support leasing of bandwidth

IP packet format:

<table>
<tr><td>4</td><td>4</td><td>8</td><td colspan="2">16</td></tr>
<tr><td>version</td><td>header length</td><td>TOS</td><td colspan="2">total length</td></tr>
<tr><td colspan="3">fragmentation identifier</td><td>flags</td><td>fragment offset</td></tr>
<tr><td colspan="2">TTL</td><td>protocol</td><td colspan="2">header checksum</td></tr>
<tr><td colspan="5">source address</td></tr>
<tr><td colspan="5">destination address</td></tr>
<tr><td colspan="5">options (if any)</td></tr>
</table>

- Header length: in 4 byte (word) units.
- TOS (type-of-service): Partially used.
- 4 bytes used for fragmentation.
- TTL (time-to-live): Prevent cycling (default 64).
- Protocol: demultiplexing key (TCP 6, UDP 17).

Fragmentation and reassembly:

LAN has maximum transmission unit (MTU): maximum frame size

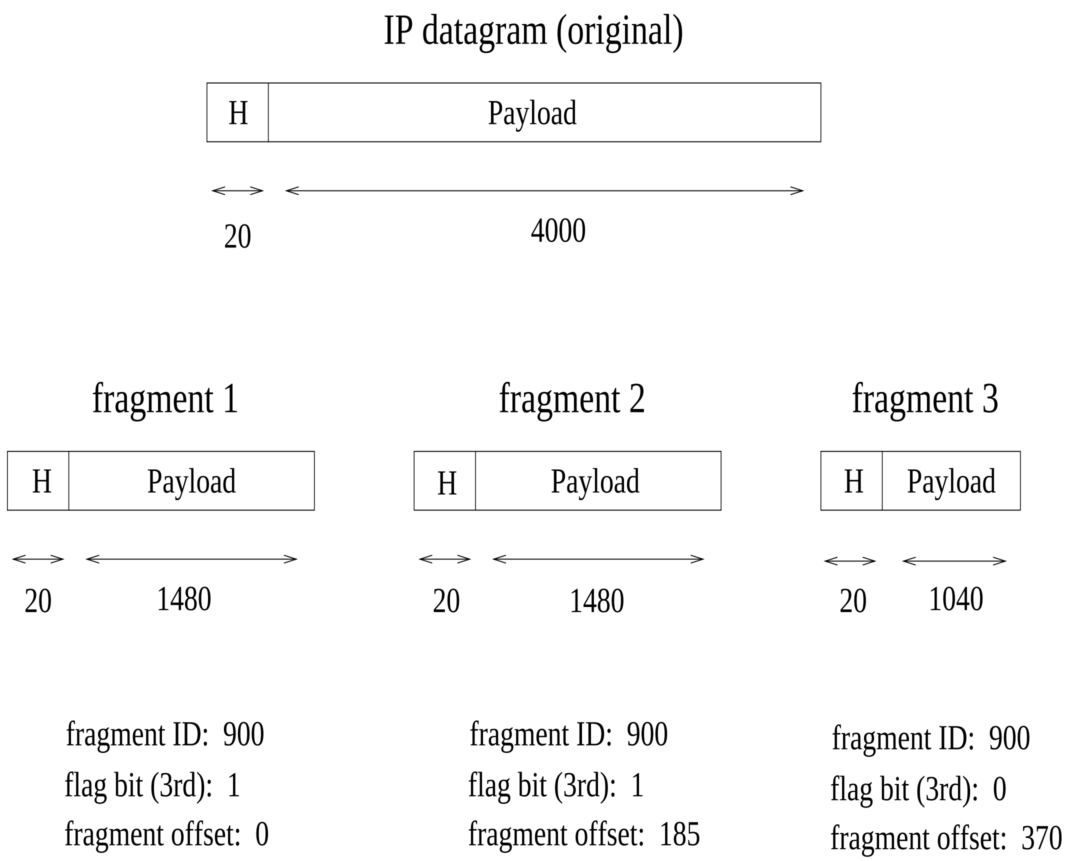
$\longrightarrow$ e.g., Ethernet 1500 B, WLAN 2313 B

- potential size mismatch problem (IP 64 kB)
- may happen multiple times hopping from LAN to LAN

Solution: fragment IP packet when needed, maintain sequencing information, then reassemble at destination.

- assign unique fragmentation ID
- set 3rd flag bit if fragmentation in progress
- sequence fragments using offset in units of 8 bytes

Example: IP fragmentation (Ethernet MTU)



Note: Each fragment is an independent IP packet.

Destination discards all fragments of an IP packet if one is lost.

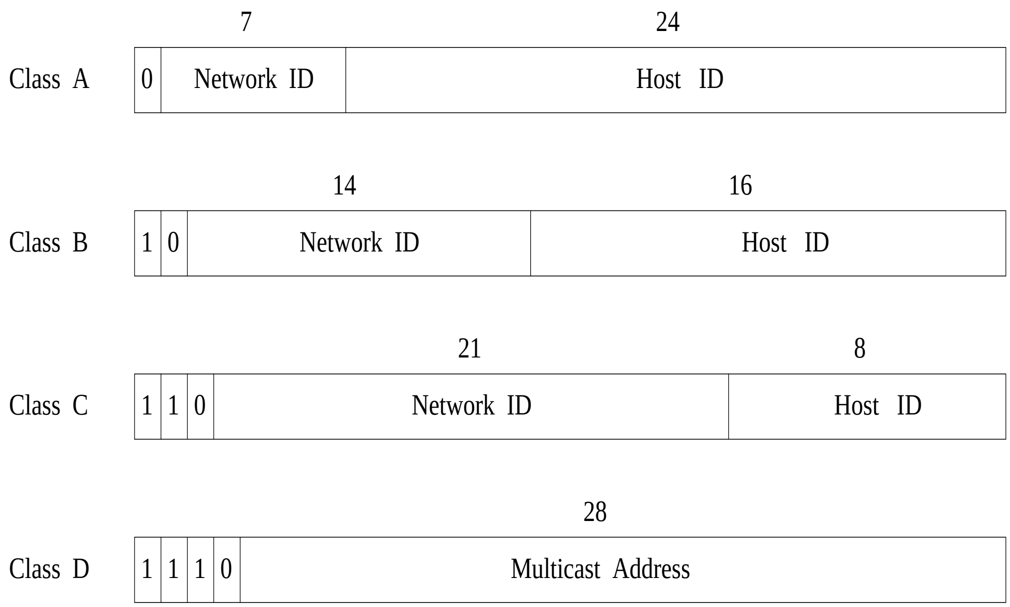
$\longrightarrow$ "all for one, one for all"

$\longrightarrow$ set 2nd flag bit to disable fragmentation

TCP: Negotiate at start-up TCP segment (packet) size based on MTU

$\longrightarrow$ tries to prevent fragmentation

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

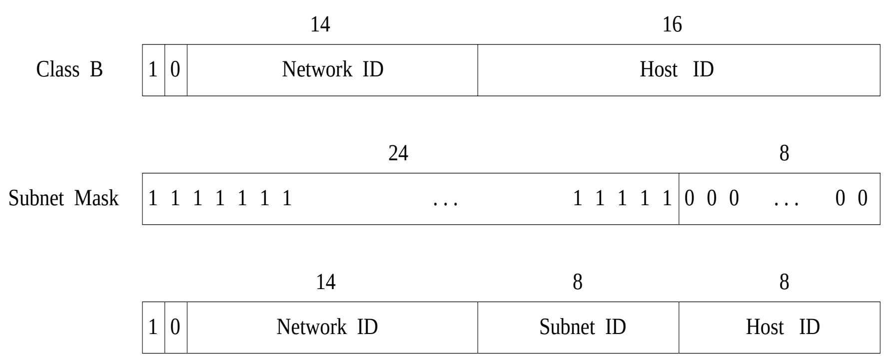
Running out of addresses. . .

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.

  $\rightarrow$ ultimately everything is LAN

  $\rightarrow$ network layer: virtual

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

  $\rightarrow$ 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.

$\longrightarrow$ aging (old caching technique)

Other approaches to solve address depletion problem:

- IPv6

  $\longrightarrow$ 128 bits (who wants it?)

- classless (vs. classful) IP addressing

  $\longrightarrow$ variable length subnetting

  $\longrightarrow$ $a.b.c.d/x$ (x: mask length)

  $\longrightarrow$ e.g., 128.10.0.0/16, 128.210.0.0/16, 204.52.32.0/20

  $\longrightarrow$ used in inter-domain routing

  $\longrightarrow$ CIDR (classless inter-domain routing)

  $\longrightarrow$ de facto Internet addressing standard

- dynamically assigned IP addresses

  → reusable

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

  → used by access ISPs, enterprises, etc.

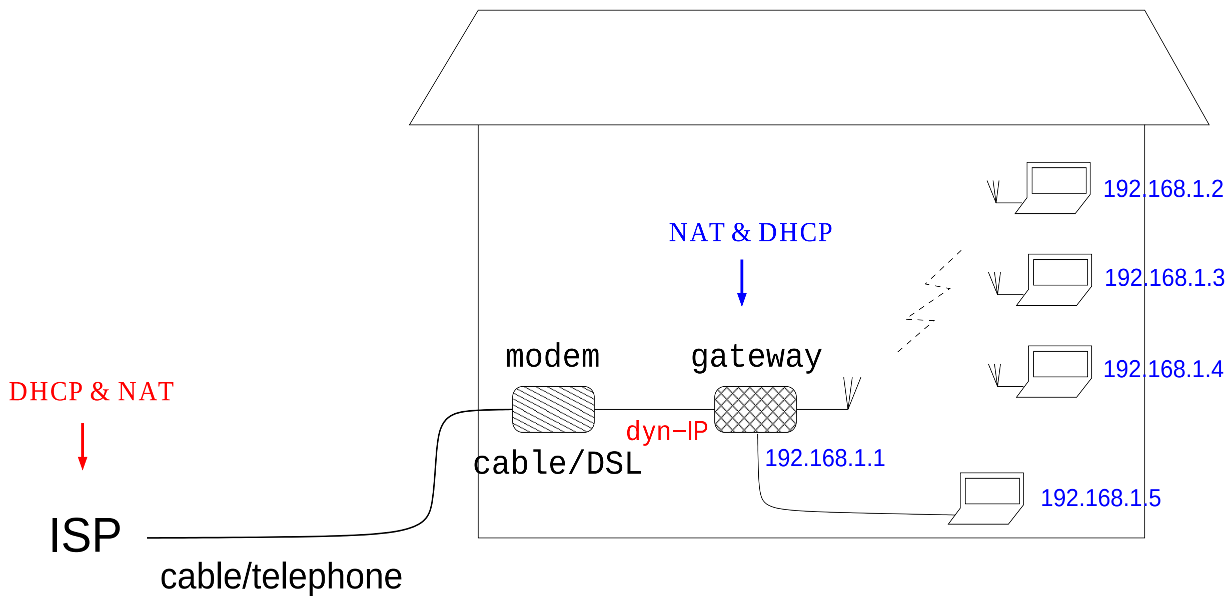
  → specifics: network address translation (NAT)

  → private/unregistered vs. public/registered IP address

  → can additionally use port numbers: NAPT

Ex.: SOHO (small office/home office)

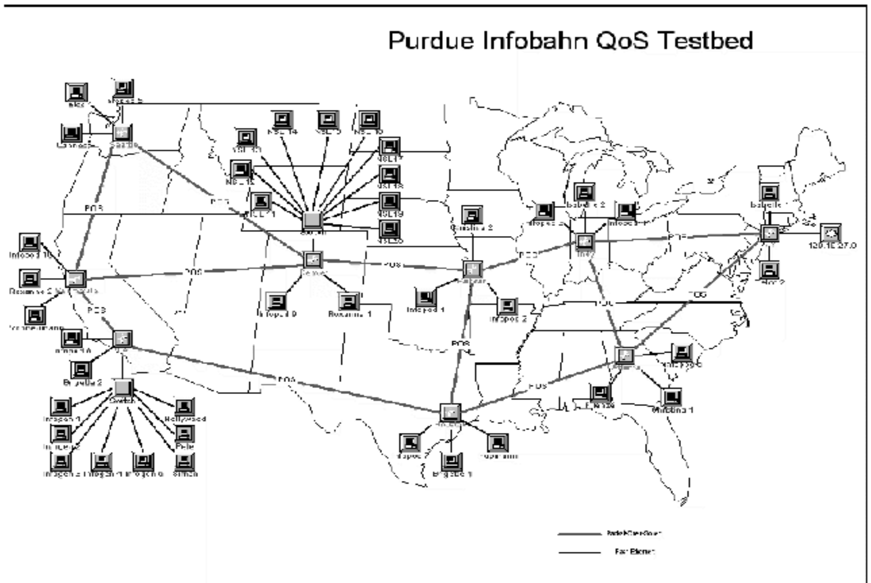
$\longrightarrow$ now: home networking



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

  $\rightarrow$ non-routable: e.g., 192.168.0.0/16, 10.0.0.0/8

Ex.: private backbone (ISP/enterprise) or testbed



- 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

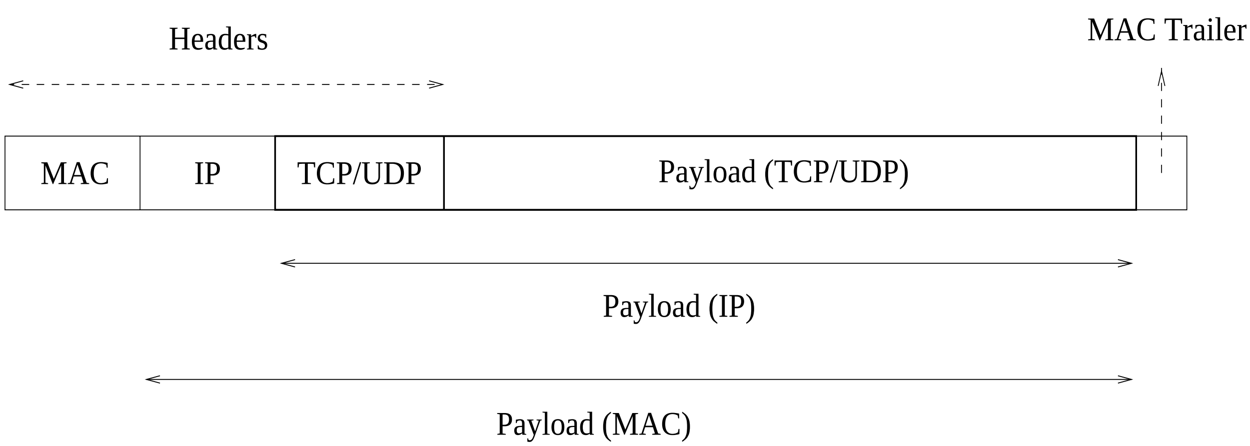
Transport Protocols: TCP and UDP

$\longrightarrow$ 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)

  $\rightarrow$ IPsec

Additional (informal) performance properties:

- Works "ok"
- Can break down under high load conditions

  $\rightarrow$ Atlanta Olympics

  $\rightarrow$ DoS attack

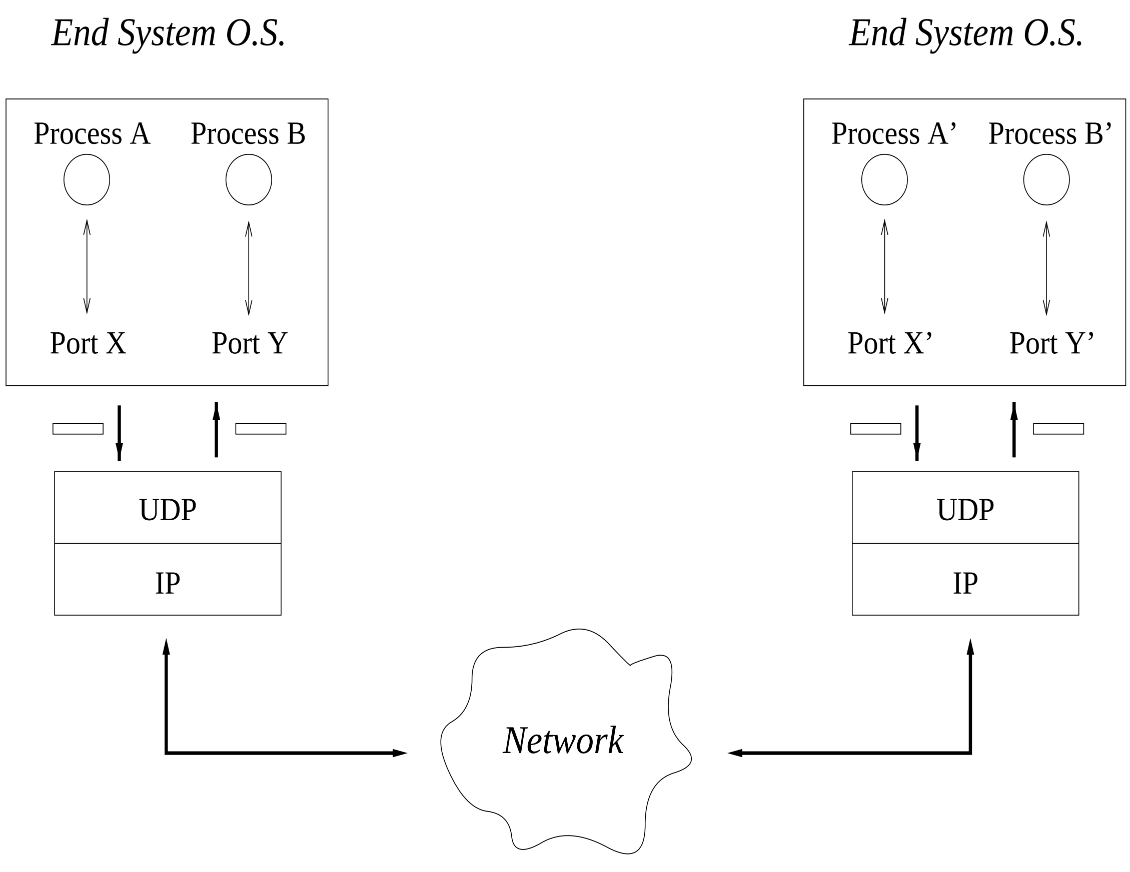
- Wide behavioral range

Goal of UDP (User Datagram Protocol):

$\longrightarrow$ process identification

$\longrightarrow$ port number as demux key

$\longrightarrow$ minimal support beyond IP



UDP packet format:

2	2
Source Port	Destination Port
Length	Checksum
Payload	

Checksum calculation (pseudo header):

<table>
<tr><th colspan="3">4</th></tr>
<tr><td colspan="3">Source Address</td></tr>
<tr><td colspan="3">Destination Address</td></tr>
<tr><td>00 $\cdots$ 0</td><td>Protocol</td><td>UDP Length</td></tr>
</table>

$\longrightarrow$ 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