IPv6 header format:

<table>
<tr><td>version
4</td><td>traffic class
8</td><td colspan="3">flow label
20</td></tr>
<tr><td colspan="3">payload length
16</td><td>next header
8</td><td>hop limit
8</td></tr>
<tr><td colspan="5">source address
128</td></tr>
<tr><td colspan="5">destination address
128</td></tr>
</table>

- traffic class: similar role as TOS field in IPv4
- flow label: flow label + source address

  $\rightarrow$ per-flow traffic management

  $\rightarrow$ significant extra bits
- next header: similar to IPv4 protocol field

  $\rightarrow$ plus double duty for option headers
- note missing fields

- Dynamically assigned IP addresses

→ share an IP address pool

→ reusable

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

→ UDP-based client/server protocol (ports 67/68)

→ note: process/daemon based service

→ used by access ISPs, enterprises, etc.

→ where are the IP address savings?

Old days of non-permanent dial-up modems . . .

- Network address translation (NAT)

$\rightarrow$ dynamically assigned + address translation

$\rightarrow$ private vs. public IP address

$\rightarrow$ private: Internet routers discard them

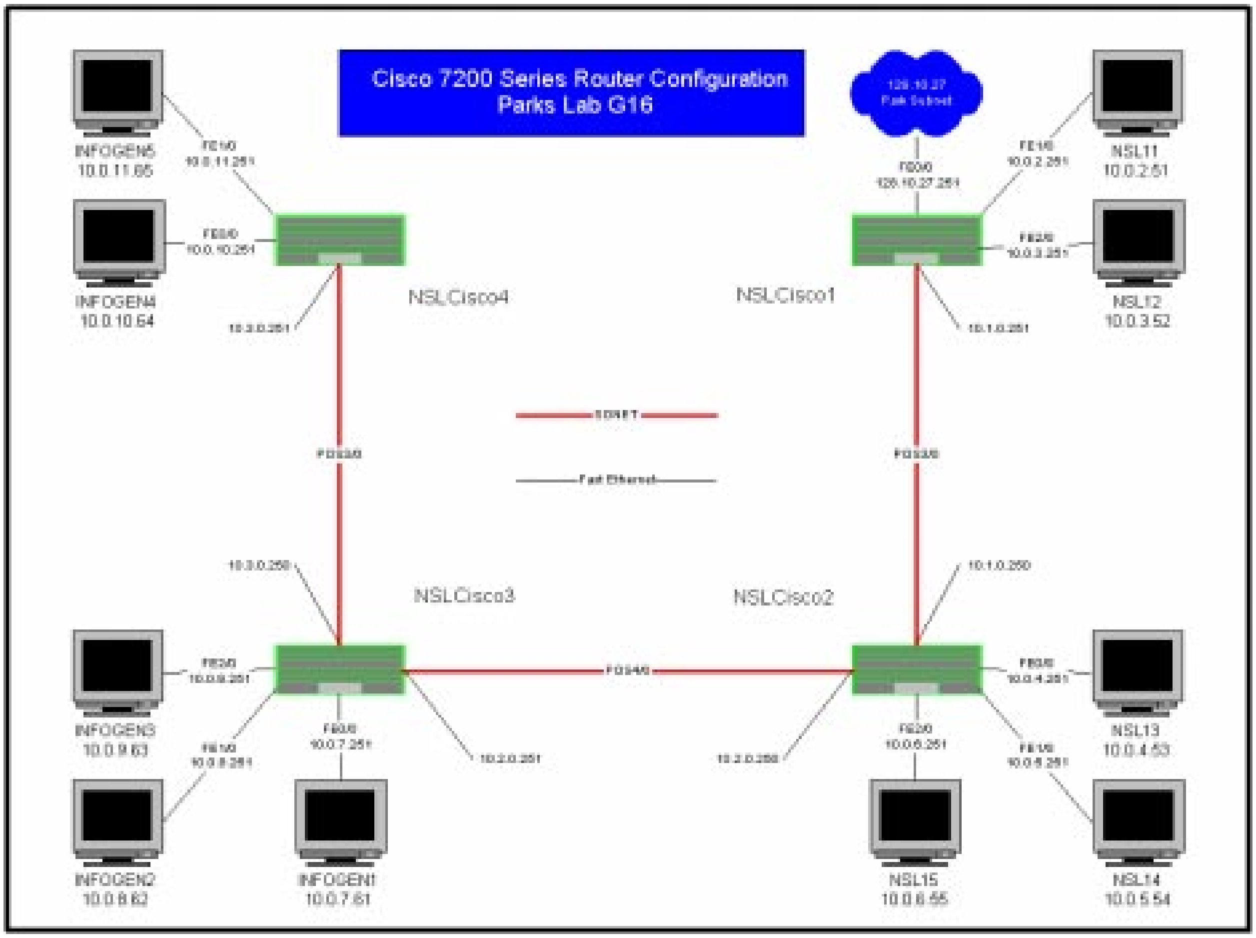
$\rightarrow$ e.g., 192.168.0.0 is private

$\rightarrow$ 10.x.x.x are also private

$\rightarrow$ useful for home networks, small businesses

$\rightarrow$ also industry and university research labs

Example: network testbed intranet (HAAS G50)



- intranet NICs have 10.0.0.0/24 addresses

  → each interface: a separate subnet

- only one of the routers connected to Internet

- NAPT (NAT + port)

→ variant of NAT: borrow src port field as address bits

Ex.: 192.168.10.10 and 192.168.10.11 both map to 128.10.27.10

but

→ 192.168.10.10 maps to 128.10.26.10:6001

→ 192.168.10.11 maps to 128.10.26.10:6002

What about port numbers of 192.168.10.10 and 192.168.10.11?

→ e.g., client process bound to 192.168.10.10:22222

→ e.g., client process bound to 192.168.10.11:33333

Doesn't matter: NAPT translation table entries

→ 192.168.10.10:22222 maps to 128.10.26.10:6001

→ 192.168.10.11:33333 maps to 128.10.26.10:6002

For example:

if 192.168.10.10:22222 is a web browser (say Firefox) downloading web page from www.purdue.edu:80

$\rightarrow$ web server knows client as 128.10.27.10:6001

$\rightarrow$ no ambiguity or confusion

$\rightarrow$ similarly for 192.168.10.11:33333

NAPT yields huge increase in effective IP address space

$\rightarrow$ IP address bits are increased to 48 (= 32 + 16)

$\rightarrow$ biggest factor preventing IP address depletion

Technical problems with NAPT?

Difficult to run servers behind DHCP intranet:

$\rightarrow$ how to discover server's dynamic IP address?

$\rightarrow$ how to discover server's dynamic port number?

$\rightarrow$ NAT traversal problem

Old solution: pay more to ISP to get fixed public IP address and port number

$\rightarrow$ not a good customer solution

$\rightarrow$ lots of P2P apps, VoIP, gaming, etc.

Two methods:

1. Proxies/relays

→ e.g., Skype: clients contact well-known server—server knows their dynamic addresses

→ server informs client its peer's dynamic IP address and port number

→ peers can talk to each directly

→ also called UDP/TCP hole punching

2. Enhanced gateway capabilities

→ e.g., IGD (Internet Gateway Device) in UPnP

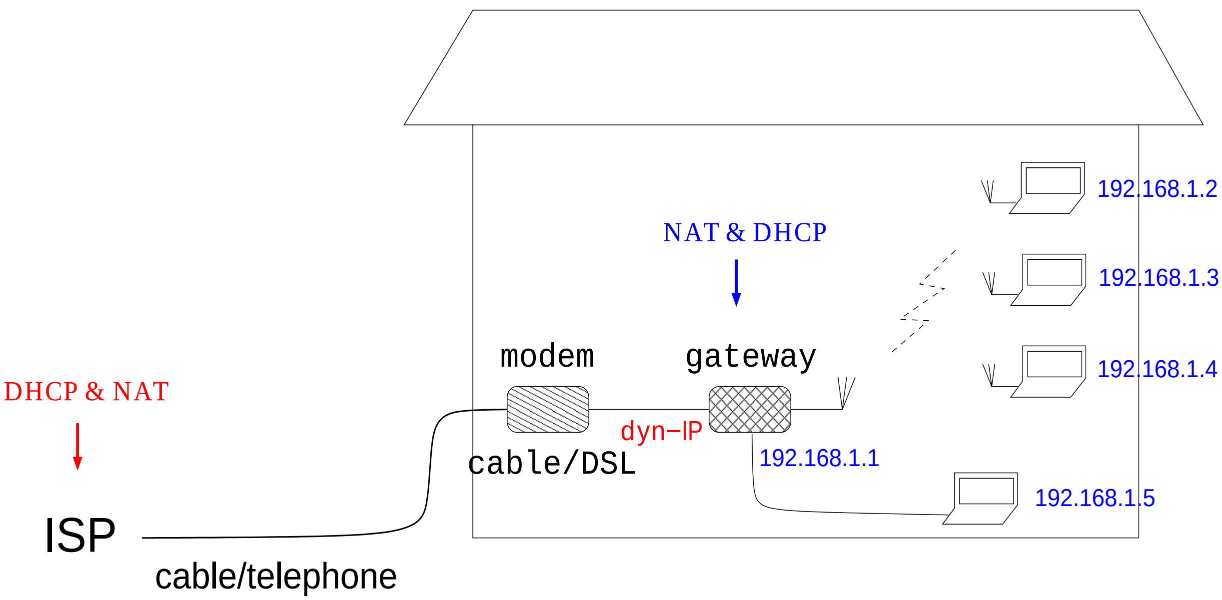
→ IGD compliant router allows user to specify desired port number

→ not much help with dynamic IP address

→ user communicates desired port number via UPnP protocol

Ex.: SOHO (small office/home office)

$\longrightarrow$ now: home networking



- dynamic IP address provided by ISP is shared through NAT
- recall: private IP addresses

  $\rightarrow$ 10.0.0.0/8, 172.16.0.0–172.31.255.255, 192.168.0.0/16

DHCP: 2-phase protocol

1. Discovery

→ client sends broadcast discovery message (UDP, client port 68, server port 67) on LAN

→ one or more DHCP servers respond with dynamic IP address

2. Allocation

→ client sends broadcast message requesting selected IP address

→ DHCP server confirms assignment

DHCP does other network configuration chores:

→ provides DNS server names

→ first-hop router/gateway

→ subnet mask

CIDR and dynamically assigned IP addresses with NAPT

$\rightarrow$ significant increase of Internet's effective address space

$\rightarrow$ saved the day

But last address block allocated by IANA (suborganization of ICANN) to regional registries early 2011

$\rightarrow$ RIRs: ARIN, RIPE, APNIC, LACNIC, AFRINIC

$\rightarrow$ back to address space crunch?

$\rightarrow$ another push for IPv6

$\rightarrow$ ISPs and companies reluctant

$\rightarrow$ a number of technical and performance issues

$\rightarrow$ 40-byte header

$\rightarrow$ not backward compatible with IPv4

$\rightarrow$ must use separate compatibility mechanisms (e.g., tunneling, hybrid sockets)

$\rightarrow$ not-so-pleasant history/memories

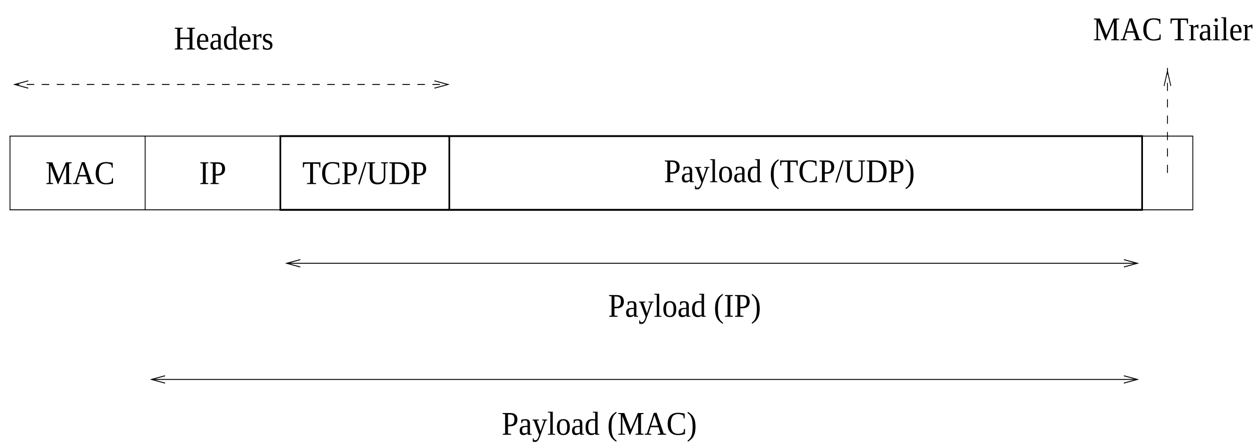
Transform Protocols: TCP and UDP

$\longrightarrow$ end-to-end protocol

$\longrightarrow$ runs on top of network layer protocols

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

Three-level encapsulation:



$\longrightarrow$ meaning of protocol "stack": push/pop headers

$\longrightarrow$ common TCP payload: HTTP

Network layer (IP) assumptions:

- unreliable
- out-of-order delivery
- absence of QoS guarantees (delay, throughput, etc.)
- insecure (IPv4)

  → IPsec (native in IPv6)

Additional performance properties:

- works "ok"
- can break down under high load conditions

  → flash crowds

  → DoS and worm attack
- wide behavioral range

  → sometimes good, so so, or bad

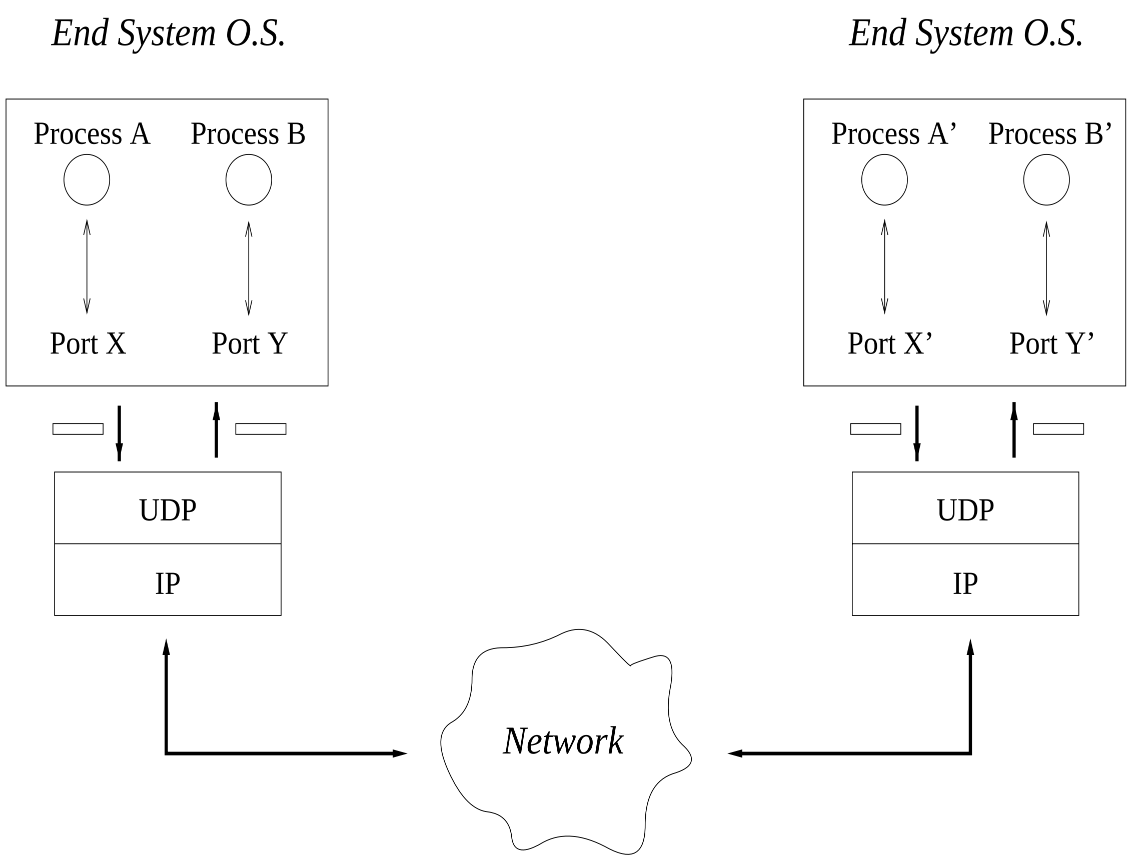
  → multitude of causes (e.g., end systems)

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):

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

$\longrightarrow$ pseudo header, UDP header and payload

UDP usage:

- multimedia streamining

  $\rightarrow$ lean and nimble

  $\rightarrow$ at minimum requires process identification

  $\rightarrow$ congestion control carried out above UDP

- stateless client/server applications

  $\rightarrow$ persistent state can be a hinderance

  $\rightarrow$ 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:

<table>
<tr><td colspan="8">2</td><td>2</td></tr>
<tr><td colspan="8">Source Port</td><td>Destination Port</td></tr>
<tr><td colspan="9">Sequence Number</td></tr>
<tr><td colspan="9">Acknowledgement Number</td></tr>
<tr><td>Header Length</td><td></td><td>URG</td><td>ACK</td><td>PSH</td><td>RST</td><td>SYN</td><td>FIN</td><td>Window Size</td></tr>
<tr><td colspan="8">Checksum</td><td>Urgent Pointer</td></tr>
<tr><td colspan="9">Options (if any)</td></tr>
<tr><td colspan="9">DATA (if any)</td></tr>
</table>

- 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 $\cdots$ 0	Protocol	TCP Segment Length

$\longrightarrow$ pseudo header, TCP header and payload