IP address format:

<table>
<thead>
<tr>
<th>Class</th>
<th>Network ID</th>
<th>Host ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class A</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Class B</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Class C</td>
<td>1 1</td>
<td></td>
</tr>
<tr>
<td>Class D</td>
<td>1 1 1</td>
<td></td>
</tr>
</tbody>
</table>

Dotted decimal notation: 10000000 00001011 00000011 00011111 ↔ 128.11.3.31

Symbolic name to IP address translation: domain name server (DNS).
Hierarchical organization: 2-level

→ network and host

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

→ 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:

<table>
<thead>
<tr>
<th>Subnet ID</th>
<th>Subnet Mask</th>
<th>Next Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>128.10.2.0</td>
<td>255.255.255.0</td>
<td>Interface 0</td>
</tr>
<tr>
<td>128.10.3.0</td>
<td>255.255.255.0</td>
<td>Interface 1</td>
</tr>
<tr>
<td>128.10.4.0</td>
<td>255.255.255.0</td>
<td>128.10.4.250</td>
</tr>
</tbody>
</table>

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

→ reachable by LAN address forwarding

→ if not, network address (IP) forwarding
Table look-up I ("where to"):

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

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

$\rightarrow$ must be done in either case

$\rightarrow$ 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
  - \textit{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

- end-to-end protocol
- runs on top of network layer protocols
- treat network layer & below as black box

Three-level encapsulation:

<table>
<thead>
<tr>
<th>Headers</th>
<th>MAC</th>
<th>IP</th>
<th>TCP/UDP</th>
<th>Payload (TCP/UDP)</th>
</tr>
</thead>
</table>

- 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:

<table>
<thead>
<tr>
<th>2</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source Port</td>
<td>Destination Port</td>
</tr>
<tr>
<td>Length</td>
<td>Checksum</td>
</tr>
<tr>
<td>Payload</td>
<td></td>
</tr>
</tbody>
</table>

Checksum calculation (pseudo header):

<table>
<thead>
<tr>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source Address</td>
</tr>
<tr>
<td>Destination Address</td>
</tr>
<tr>
<td>00 · · · 0</td>
</tr>
</tbody>
</table>

→ pseudo header, UDP header and payload
UDP usage:

• Multimedia streaming
  → 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

    \[\rightarrow\] connection-oriented, i.e., stateful

    \[\rightarrow\] complex mixture of functionalities
Segmentation task: provide “stream” interface to higher level protocols

→ exported semantics: contiguous byte stream

→ 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>
<thead>
<tr>
<th>Source Port</th>
<th>Destination Port</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sequence Number</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Acknowledgement Number</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Header Length</th>
<th>Upper</th>
<th>ACK</th>
<th>SYN</th>
<th>FIN</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Window Size</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Checksum</th>
<th>Urgent Pointer</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Options (if any)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DATA (if any)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</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):

\[
\begin{array}{cccc}
\hline
\text{Source Address} \\
\hline
\text{Destination Address} \\
00 \cdots 0 & \text{Protocol} & \text{TCP Segment Length} \\
\hline
\end{array}
\]

\[\rightarrow \text{pseudo header, TCP header and payload}\]