INTERNETWORKING

Goal: Interconnect multiple LANs

Why?

• Diverse LANs speak different languages
  → need to make them talk to each other
  → cannot use native LAN interconnection technology

• Management flexibility

Key problems:

• How to choose paths (routing)?
• How to regulate traffic flow (congestion control)?
  → not too fast, not too slow
• How to provide service quality (QoS control)?
Routing: packet- vs. circuit-switched

Global Internet: packet-switched

→ every packet is an autonomous entity

Intranet of large ISPs: both packet- and circuit-switched
Circuit-switched routing: set-up

- connection set-up message: signaling
  → route specification

- source tag “A” inserted into route look-up table
  → entry deletion upon session termination
Packet-switched routing: set-up

→ no connection set-up signaling

→ each packet: autonomous entity

Special case: source routing

• packet contains path information

→ ⟨A, C, . . . , B⟩

• drawback: header length increases with path length

→ not good for fast packet handling

→ option still available in IP: may be used for management purposes
Destination-based forwarding:

- determine output port by destination address
- source address ignored

Negative impact stemming from ignoring source address?
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 is key

IP represents:

• common language for carrying out non-LAN-specific conversations
  → technical definition of Internet
• functionality and design philosophy
  → simple core/complex edge
  → called end-to-end paradigm
Reliability over best-effort Internet:

- simplifies router design
- increases complexity of end systems (e.g., servers, laptops, handhelds)

→ implement ARQ at sender/receiver

```
LAN/WAN

A Unreliable Medium B

"Black Box"
```
IPv4 packet format:

<table>
<thead>
<tr>
<th>4</th>
<th>4</th>
<th>8</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>version</td>
<td>header length</td>
<td>TOS</td>
<td>total length</td>
</tr>
<tr>
<td>fragmentation identifier</td>
<td>flags</td>
<td>fragment offset</td>
<td></td>
</tr>
<tr>
<td>TTL</td>
<td>protocol</td>
<td>header checksum</td>
<td></td>
</tr>
<tr>
<td>source address</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>destination address</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</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 (e.g., 64).
- Protocol: demultiplexing key (TCP 6, UDP 17).
Fragmentation and reassembly:

LAN has maximum transmission unit (MTU):
→ maximum frame size
→ e.g., Ethernet 1500 B, WLAN 2304 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)

<table>
<thead>
<tr>
<th>Fragment 1</th>
<th>Fragment 2</th>
<th>Fragment 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>H</td>
<td>Payload</td>
<td>H</td>
</tr>
<tr>
<td>20</td>
<td>1480</td>
<td>20</td>
</tr>
</tbody>
</table>

fragment ID: 900
flag bit (3rd): 1
fragment offset: 0

fragment ID: 900
flag bit (3rd): 1
fragment offset: 185

fragment ID: 900
flag bit (3rd): 0
fragment offset: 370
Note: Each fragment is an independent IP packet.

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

→ “all for one, one for all”

→ set 2nd flag bit to disable fragmentation

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

→ prevent fragmentation
IP address format:

Class A

\[
\begin{array}{c|c|c}
0 & \text{Network ID} & \text{Host ID} \\
\end{array}
\]

Class B

\[
\begin{array}{c|c|c}
1 & \text{Network ID} & \text{Host ID} \\
\end{array}
\]

Class C

\[
\begin{array}{c|c|c}
1 & 1 & \text{Network ID} & \text{Host ID} \\
\end{array}
\]

Class D

\[
\begin{array}{c|c}
1 & 1 & 1 & \text{Multicast Address} \\
\end{array}
\]

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 (NIC) has an IP address; single host can have multiple IP addresses.
→ single-homed vs. multi-homed

Running out of IP addresses . . . or not?
→ note: IANA gave out last block to regional registries
→ should Purdue get a class B address?
→ how about your start-up company?
→ what about Purdue’s CS Dept.?
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

<table>
<thead>
<tr>
<th>Class B</th>
<th>14</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network ID</td>
<td>10</td>
<td>Host ID</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Subnet Mask</th>
<th>24</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1 1 1 1 1 1</td>
<td>...</td>
<td>1 1 1 1 1 0 0 0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>14</th>
<th>8</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 0</td>
<td>Network ID</td>
<td>Subnet ID</td>
</tr>
</tbody>
</table>

To determine subnet ID:

• AND IP address and subnet mask
  → already know if class A, B, C, or D
• 3-level hierarchy
Example: Purdue CS network

→ a few years back
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 $\text{SubnetID} = \text{DestAddr AND SubnetMask}$.
- Compare $\text{SubnetID}$ with $\text{SubnetID}$.
- 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 (standard caching technique)
Subnetting only goes so far.

→ depts. within Purdue share same class B address
→ what about your start-up company?
→ only $2^{21}$ class C addresses available