Last Time

- How to interconnect LANs
- Repeaters/Hubs
- Bridges
- Learning Bridges
- Spanning trees
- Switches
Today

- The Network Layer and Internetworking

- Recall problems with L2 bridging/switching
  - Homogeneous link layer (all Ethernet)
  - Broadcast traffic to all members (scaling)
  - No control over topology (and root is bottleneck)
  - Single administrative domain (who controls?)

- Goal: scalably interconnect large numbers of networks of different types
First some history...

- 1968: DARPAnet (precursor to Internet)
  - Bob Taylor, Larry Roberts create program to build first wide-area packet-switched network
  - Why?

- 1978: new networks emerge
  - SATNet, Packet Radio, Ethernet
  - All “islands” to themselves

- Big question: how to connect these networks?

Plug: “Where Wizards Stay Up Late” by Hafner and Lyon is the best account of early Internet History I’ve seen.
DARPAnet Primary Goal: Connect Stuff

- “Effective technique for multiplexed utilization of existing interconnected networks” – David Clark

  - **Minimal** assumptions about underlying networks
    - No support for broadcast, multicast, real-time, reliability
    - Extra support could actually get in the way (X.25 example)
  - Packet switched, store and forward
    - Matched application needs, nets already packet switched
    - Enables **efficient resource sharing**/high utilization
  - “Gateways” interconnect networks
    - Routers in today’s nomenclature
Why is this hard?

Heterogeneity

- **Addressing**
  - Each network media has a different addressing scheme; routing protocol
- **Bandwidth**
  - Modems to terabits
- **Latency**
  - Seconds to nanoseconds
- **Packet size**
  - Dozens to thousands of bytes
- **Loss rates**
  - Differ by many orders of magnitude
- **Service guarantees**
  - Send and pray vs reserved bandwidth
How to connect different networks?

- **Monopoly**
  - Re-engineer network to use a single set of protocols everywhere
  - Economic cost

- **Translation Gateways**
  - Translates directly between different network formats
  - $O(n^2)$ complexity ($n$ is # of protocols)
  - May not be able to translate perfectly (QoS)

- **Indirection Gateways**
  - Translates between local network format and universal “intermediate” format
  - $O(n)$ complexity
  - May not take advantage of features in underlying network

- Note impact of economics on decision. Engineering not science.
Internetworking

- Cerf & Kahn74, “A Protocol for Packet Network Intercommunication”
  - Foundation for the Internet

- All packets use a common Internet Protocol
  - Any underlying data link protocol
  - Any higher layer transport protocol
How IP works

Separate physical networks communicate to form a *single* logical network

---

**How IP works**

Separate physical networks communicate to form a *single* logical network.
**What's a router?**

- Box with a bunch of network interfaces in it

- **Forwarding**
  - Receives packet
  - Strips off data-link layer header
  - Uses network layer header to determine where to forward packet (i.e. which output port)
  - Puts on new data-link layer header
  - Transmits packet

- **Routing**
  - Keep track of network topology so you know where to forward packets

- **Queuing**
  - Buffer packets if router can’t forward it right now

- **Buffer Management**
  - Drop packet if you run out of buffer space
Internet architecture: What should IP do?

- Impose few demands on network
  - Make few assumptions about what network can do
  - No QoS, no reliability, no ordering, no large packets
  - No persistent state about communications
- Manage heterogeneity at hosts
  - Adapt to underlying network heterogeneity
  - Re-order packets, detect errors, retransmit lost messages, etc.
  - Persistent network state only kept in hosts (fate-sharing)
- Service model: send and pray
So what *does* IP do?

- Addressing
- Fragmentation
  - E.g. FDDI’s maximum packet is 4500 bytes while Ethernet is 1500 bytes, how to manage this?
- Some error detection
- Potpourri

- Routers forward packets to next hop
  - They do not
    - Detect data corruption, packet loss, packet duplication
    - Reassemble or retransmit packets
Addressing Considerations

- Fixed length or variable length?
- Issues:
  - Flexibility
  - Processing costs
  - Header size
- Engineering choice: IP uses fixed length addresses
Addressing Considerations (2)

- Hierarchical vs flat
  - How much does each router need to know?
- Original DARPAnet IP addressing (24 bits)
  - Global inter-network address (8 bits)
  - Local network-specific address (16 bits)

- Very successful, but now obsolete… what assumption do you think was problematic? (hint: modern IP uses 32 bit addresses, and IPv6 uses 128 bit addresses)
Fragmentation Issue

- Different networks may have different frame limits (MTUs)
  - Ethernet 1.5K, FDDI 4.5K
- Don’t know if packet will be too big for path beforehand
- Router breaks up single IP packet into two or more smaller IP packets
  - Each fragment is labeled so it can be correctly reassembled
  - End host reassembles them into original packet
Error detection

- **Bit errors**
  - Data-link layer (e.g. Ethernet) generates CRC for each packet
    - When packet is received by router or host, it checks packet against CRC for errors
    - Why isn’t this enough?
  - Network-layer (IP) checksum written by sender
  - Checked at each hop and by receiver
    - Why not just check at the receiving host?

- **Packet losses**
  - Not part of IP, we’ll deal with this next time
# Today’s IP Packet Header

A typical IP packet header consists of several fields, totaling 20 bytes. Here’s a breakdown:

<table>
<thead>
<tr>
<th>Field</th>
<th>Position</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version (ver)</td>
<td>0</td>
<td>4-bit version number</td>
</tr>
<tr>
<td>Header Length (HL)</td>
<td>1-2</td>
<td>16-bit length of header in 32-bit words</td>
</tr>
<tr>
<td>Type of Service (TOS)</td>
<td>3-4</td>
<td>8-bit code that reflects priorities to network</td>
</tr>
<tr>
<td>Identification</td>
<td>5-12</td>
<td>16-bit identification number for each fragment of a datagram</td>
</tr>
<tr>
<td>Flags (F)</td>
<td>13-14</td>
<td>2-bit code that controls fragmentation and defragmentation of datagrams</td>
</tr>
<tr>
<td>Offset (O)</td>
<td>15-16</td>
<td>13-bit field that specifies the offset of the first fragmenant in a datagram</td>
</tr>
<tr>
<td>Time to Live (TTL)</td>
<td>17-18</td>
<td>8-bit code that indicates the time available before it is discarded</td>
</tr>
<tr>
<td>Protocol</td>
<td>19-20</td>
<td>8-bit field that identifies the protocol used in the payload</td>
</tr>
<tr>
<td>Header Checksum</td>
<td>21-31</td>
<td>16-bit field that provides a checksum over the header</td>
</tr>
<tr>
<td>Source Address</td>
<td>32-47</td>
<td>32-bit address of the sending host</td>
</tr>
<tr>
<td>Destination Address</td>
<td>48-63</td>
<td>32-bit address of the destination host</td>
</tr>
<tr>
<td>Options (if any)</td>
<td>64-</td>
<td>Variable-length fields that provide additional information to the network</td>
</tr>
<tr>
<td>Data (if any)</td>
<td>64-</td>
<td>Variable-length data payload</td>
</tr>
</tbody>
</table>

The header is structured to accommodate various types of information, ensuring efficient and reliable communication across networks.
Version field

- Which version of IP is this?
  - Plan for change
  - Very important!

- Current versions
  - 4: most of Internet
  - 6: new protocol with larger addresses
  - What happened to 5? Standards body politics.
Header length

- How big is IP header?
  - In bytes/octetes
  - Variable length
    » Options
  - Engineering consequences of variable length...

- Most IP packets are 20 bytes long
Type-of-Service

- How should this packet be treated?
  - Care/don’t care for delay, throughput, reliability, cost
  - How to interpret, how to apply on underlying net?
  - Largely unused until 2000 (hijacked for new purposes, ECN & Diffserv)
Length

- How long is whole packet in bytes/octets?
  - Includes header
  - Limits total packet to 64K
  - Redundant?
Fragmentation

- Sender writes unique value in *identification* field.
- If router fragments packet it copies id into each fragment.
- *Offset* field indicates position of fragment in bytes (offset 0 is first).
  - *MoreFragments* flag indicates that this isn’t the last fragment.
  - *DontFragment* flag tells gateway not to fragment.
- All routers must support 576 byte packets (MTU).
IP Fragmentation and Reassembly

One large datagram becomes several smaller datagrams

<table>
<thead>
<tr>
<th>length</th>
<th>ID</th>
<th>MF</th>
<th>offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>4000</td>
<td>x</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1500</td>
<td>x</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1500</td>
<td>x</td>
<td>1</td>
<td>1480</td>
</tr>
<tr>
<td>1040</td>
<td>x</td>
<td>0</td>
<td>2960</td>
</tr>
</tbody>
</table>
Aside: costs of fragmentation

- Interplay between fragmentation and retransmission

- Packet must be completely reassembled before it can be consumed on the receiving host

- What if a fragment gets lost?
Path MTU Discovery

- Path MTU is the smallest MTU along path
  - Packets less than this size don’t get fragmented
- Fragmentation is a burden for routers
  - We already avoid reassembling at routers
  - Avoid fragmentation too by having hosts learn path MTUs
- Hosts send packets, routers return error if too large
  - Hosts discover limits, can size packets at source
  - Reassembly at destination as before
**TTL (Time-to-Live)**

- How many more routers can this packet pass through?
  - Designed to limit packet from looping forever
- Each router decrements TTL field
- If TTL is 0 then router discards packet
Protocol

- Which transport protocol is the data using?
  - i.e. how should a host interpret the data

- TCP = 6
- UDP = 17
**Header checksum**

- Detects errors in IP header
  - Calculated by sending host
  - Checked by receiving host
- Must be recalculated by router. Why?
- Only protects header, not data
Modern IP addressing

- 32-bits in an IPv4 address
  - Dotted decimal format a.b.c.d
  - Each represent 8 bits of address
- Hierarchical: Network part and host part
  - E.g. IP address 132.239.15.3
    - 132.239 refers to the UCSD campus network
    - 15.3 refers to the host gremlin.ucsd.edu
- Which part is network vs host?
Class-based addressing (<1993)

- Most significant bits determines “class” of address

<table>
<thead>
<tr>
<th>Class</th>
<th>Network</th>
<th>Host</th>
<th>Networks</th>
<th>Hosts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class A</td>
<td>0</td>
<td>1</td>
<td>127</td>
<td>16M</td>
</tr>
<tr>
<td>Class B</td>
<td>1 0</td>
<td>21</td>
<td>16K</td>
<td>64K</td>
</tr>
<tr>
<td>Class C</td>
<td>1 1 0</td>
<td>8</td>
<td>2M</td>
<td>254</td>
</tr>
</tbody>
</table>

- Special addresses
  - Class D (1110) for multicast, Class E (1111) experimental
  - 127.0.0.1: local host (a.k.a. the loopback address
  - Host bits all set to 0: network address
  - Host bits all set to 1: broadcast address
Original IP Route Lookup – Example

- Address would specify prefix for forwarding table
  - Simple, fast lookup
- www.ucsd.edu address 132.239.50.184
  - Class B address – class + network is 132.239
  - Lookup 132.239 in forwarding table
  - Prefix – part of address that really matters for routing
- Forwarding table contains
  - List of class+network entries
  - A few fixed prefix lengths (8/16/24)

- Problems looming?
IP Address Problem (1991)

- Address space depletion
  - In danger of running out of classes A and B
- Why?
  - Class C too small for most organizations (only 253 addresses)
  - Very few class A – very careful about giving them out (who has 16M hosts anyway?)
  - Class B – greatest problem
IP Address Utilization (‘98)

http://www.caida.org/outreach/resources/learn/ipv4space/
IP Addressing Problems

- Class B sparsely populated
  - But people refuse to give it back
- Large forwarding tables
  - 2 Million possible class C groups
- Solution
  - Assign multiple class C addresses
  - Assign consecutive blocks
  - RFC1338 – Classless Inter-Domain Routing (CIDR)
Classless addressing (1993>)

- Classless Inter-Domain Routing (CIDR)
  - Routes represented by tuple (network prefix/mask)
  - Allows arbitrary allocation between network and host address

<table>
<thead>
<tr>
<th>Network</th>
<th>Host</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prefix</td>
<td>Mask=# significant bits representing prefix</td>
</tr>
</tbody>
</table>

- e.g. 10.95.1.2/8: 10 is network and remainder (95.1.2) is host

- Pro: Finer grained allocation; aggregation
- Con: More expensive lookup: longest prefix match
Route Aggregation

- CIDR allows forwarding table size to be reduced by suballocating networks from larger contiguous address range

Organization 0
- 200.23.16.0/23

Organization 1
- 200.23.18.0/23

Organization 2
- 200.23.20.0/23

Organization 7
- 200.23.30.0/23

Fly-By-Night-ISP

"Send me anything with addresses beginning 200.23.16.0/20"

Internet

ISPs-R-Us

"Send me anything with addresses beginning 199.31.0.0/16"
Complexity: More specific addresses

- ISPs-R-Us has a more specific route to Organization 1… how does forwarding table work?

```
Organization 0
  200.23.16.0/23

Organization 2
  200.23.20.0/23

Organization 7
  200.23.30.0/23

Organization 1
  200.23.18.0/23

ISP's-R-Us

Fly-By-Night-ISP

“Send me anything with addresses beginning 200.23.16.0/20”

“Send me anything with addresses beginning 199.31.0.0/16 or 200.23.18.0/23”

Internet
```
Longest prefix match

- Forwarding table contains a bunch of prefix/length tuples
  - E.g. 200.23.16.0/20 and 200.23.18.0/23
  - What to do if a packet arrives for destination 200.23.18.1?
  - Need to find the longest prefix in the table which matches it (200.23.18.0/23)

- Not a simple table, requires multiple memory lookup
LPM Algorithms

- Traditional method – Patricia Tree
  - Arrange route entries into a series of bit tests
  - Worst case = 32 bit tests
  - Problem: memory speed is a bottleneck
- Lots of research has been done to improve this…

0

10

default
0/0

16

128.2/16

19

128.32/16

128.32.130/24 128.32.150/24

Bit to test – 0 = left child, 1 = right child
Forwarding example

- Packet to 10.1.1.3 arrives
- Path is R2 – R1 – H1 – H2
Forwarding example (2)

- Packet to 10.1.1.3
- Matches 10.1.0.0/23

Routing table at R2

<table>
<thead>
<tr>
<th>Destination</th>
<th>Next Hop</th>
<th>Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>127.0.0.1</td>
<td>127.0.0.1</td>
<td>lo0</td>
</tr>
<tr>
<td>Default or 0/0</td>
<td>provider</td>
<td>10.1.16.1</td>
</tr>
<tr>
<td>10.1.8.0/24</td>
<td>10.1.8.1</td>
<td>10.1.8.1</td>
</tr>
<tr>
<td>10.1.2.0/23</td>
<td>10.1.2.1</td>
<td>10.1.2.1</td>
</tr>
<tr>
<td>10.1.0.0/23</td>
<td>10.1.2.2</td>
<td>10.1.2.1</td>
</tr>
</tbody>
</table>
Forwarding example (3)

- Packet to 10.1.1.3
- Matches 10.1.1.1/31
  - Longest prefix match

Routing table at R1

<table>
<thead>
<tr>
<th>Destination</th>
<th>Next Hop</th>
<th>Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>127.0.0.1</td>
<td>127.0.0.1</td>
<td>lo0</td>
</tr>
<tr>
<td>Default or 0/0</td>
<td>10.1.2.1</td>
<td>10.1.2.2</td>
</tr>
<tr>
<td>10.1.0.0/24</td>
<td>10.1.0.1</td>
<td>10.1.0.1</td>
</tr>
<tr>
<td><strong>10.1.1.0/24</strong></td>
<td><strong>10.1.1.4</strong></td>
<td><strong>10.1.1.1</strong></td>
</tr>
<tr>
<td>10.1.2.0/23</td>
<td>10.1.2.2</td>
<td>10.1.2.2</td>
</tr>
<tr>
<td><strong>10.1.1.2/31</strong></td>
<td><strong>10.1.1.2</strong></td>
<td><strong>10.1.1.2</strong></td>
</tr>
</tbody>
</table>
### Forwarding example (4)

- Packet to 10.1.1.3
- Direct route
  - Longest prefix match

#### Routing table at H1

<table>
<thead>
<tr>
<th>Destination</th>
<th>Next Hop</th>
<th>Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>127.0.0.1</td>
<td>127.0.0.1</td>
<td>lo0</td>
</tr>
<tr>
<td>Default or 0/0</td>
<td>10.1.1.1</td>
<td>10.1.1.2</td>
</tr>
<tr>
<td>10.1.1.0/24</td>
<td>10.1.1.2</td>
<td>10.1.1.1</td>
</tr>
<tr>
<td>10.1.1.3/31</td>
<td>10.1.1.2</td>
<td>10.1.1.2</td>
</tr>
</tbody>
</table>
Misc stuff

- How do you get an IP address?
  - Providers (registries)
  - Users (DHCP)

- How to map between IP and link-layer addresses?
  - ARP

- ICMP
How to get an IP address?

- Providers (e.g. ISPs)
  - You already have a bunch from the days when you called Jon Postel and asked for them (e.g. AT&T)
  - You get them from another provider (e.g. SavageDialup buys service from Sprint and gets a /24 from one of their address blocks)
  - You get one directly from a routing registry
    - ARIN: North America, APNIC (Asia Pacific), RIPE (Europe), LACNIC (Latin America)
    - Registries get address from IANA (Internet Assigned Numbers Authority)
    - IANA, in theory, perhaps gets its authority from ICANN (Internet Corporation for Assigned Names and Numbers) and who knows where they get their authority from (used to be US Dept of Commerce)
How do *you* get an IP address?

- Well from your provider!
- But how do you know what it is?
- Manual configuration
  - They tell you and you type that number into your computer (along with the default gateway, DNS server, etc.)
- Automated configuration
  - Dynamic Host Resolution Protocol (DHCP)
DHCP

- Basic algorithm
  - Host broadcasts “DHCP discover” msg on LAN (e.g. Ethernet broadcast)
  - DHCP server responds with “DHCP offer” msg
  - Host requests IP address: “DHCP request” msg
  - DHCP server sends address: “DHCP ack” msg
    » Containing IP address you get to use
- Easy to have fewer addresses than hosts (e.g. UCSD wireless) and easy to renumber network (use new addresses)
- Issues
  - What if host goes away (how to get address back?)
    » Address is a “lease” not a “grant”, has a timeout associated with it. Host must periodically renew if it wants to keep address
  - Same host may have different IP addresses at different times?
How to map IP address to Ethernet address?

- Address Resolution Protocol (ARP)
  - Broadcast “search” for IP address
    » E.g., “who-has 128.2.184.45 tell 128.2.206.138” sent to Ethernet broadcast (all 1’s address)
  - Destination responds (only to requester using unicast) with appropriate 48-bit Ethernet address
    » E.g, “reply 128.2.184.45 is-at 0:d0:bc:f2:18:58” sent to 0:c0:4f:d:ed:c6

- Issues
  - What if mapping changes (IP address changes, host moves, etc)
    » Again, use leases/timeouts
  - What if someone lies? (arp cache poisoning)
ICMP

- What happens when things go wrong?
  - Need a way to test/debug a large, widely distributed system
- ICMP = Internet Control Message Protocol (RFC792)
  - Companion to IP – required functionality
- Used for error and information reporting:
  - Errors that occur during IP forwarding
  - Queries about the status of the network
ICMP Error Message Generation

Error during forwarding!
Common ICMP Messages

- Destination unreachable
  - “Destination” can be host, network, port or protocol
- Redirect
  - To shortcut circuitous routing
- TTL Expired
  - Used by the “traceroute” program
- Echo request/reply
  - Used by the “ping” program
- ICMP messages include portion of IP packet that triggered the error (if applicable) in their payload
ICMP Restrictions

- The generation of error messages is limited to avoid cascades … error causes error that causes error!
- Don’t generate ICMP error in response to:
  - An ICMP error
  - Broadcast/multicast messages (link or IP level)
  - IP header that is corrupt or has bogus source address
  - Fragments, except the first
- ICMP messages are often rate-limited too.
The Internet was designed
- There is no natural law that says TCP/IP, network routing, etc., had to look the way it does now
- It could well have been done differently

The Internet evolves
- The Internet today is not the same Internet as 1988, 1973
- IP (and other protocols) have changed considerably over the years (and continues to change -> IPv6)

Many of these design issues are deep
- Seemingly straightforward decisions can have very subtle correctness and performance implications
- E.g. Implications of fragmentation
Summary & For Next Time...

- Network layer provides end-to-end data delivery across an internetwork, not just a LAN
  - Issues of scale and heterogeneity
  - IP protocol: simple common intermediate protocol

- Up next: Reliable communication (the transport layer)
  - Read 2.5 and Chapter 5 (up to, but not including, 5.3)
  - I’ll give out the first project assignment (solo)