Lecture 4: IP, Forwarding, and Switch Fabrics
Overview

- **Internet Protocol (v4)**
  - What it provides and its header
  - Fragmentation and assembly

- **IP Addresses**
  - Format and assignment: class-A, class-B, CIDR
  - Mapping, translation, and DHCP

- **Packet forwarding, circuits, source routing**

- **Switch fabrics**

- **Bisection bandwidth**
Internet Protocol Goal

- Glue lower-level networks together
The Hourglass, Revisited

FTP
TCP
UDP
IP
NET1
NET2
NETn
HTTP
NV
TFTP

TCP

UDP

IP

NET1
NET2
⋯
NETn
Internet Protocol

- Connectionless (datagram-based)
- Best-effort delivery (unreliable service)
  - packets are lost
  - packets are delivered out of order
  - duplicate copies of a packet are delivered
  - packets can be delayed for a long time
# IPv4 packet format

<table>
<thead>
<tr>
<th>vers</th>
<th>hdr len</th>
<th>TOS</th>
<th>Total Length</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Identification</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TTL</td>
</tr>
<tr>
<td>Source IP address</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Destination IP address</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Options</td>
<td>Padding</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
IP header details

- Routing is based on destination address
- TTL (time to live) decremented at each hop (avoids loops)
  - TTL mostly saves from routing loops
  - But other cool uses...
- Fragmentation possible for large packets
  - Fragmented in network if crosses link w. small frame size
  - MF bit means more fragments for this IP packet
  - DF bit says “don’t fragment” (returns error to sender)
- Following IP header is “payload” data
  - Typically beginning with TCP or UDP header
Example Encapsulation

Sending

Application data
Transport header
IP header
Link layer header

Receiving
## IPv4 packet format

<table>
<thead>
<tr>
<th>0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>vers</td>
</tr>
<tr>
<td>------</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>F FF</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>TCP or UDP header</td>
</tr>
<tr>
<td>TCP or UDP payload</td>
</tr>
</tbody>
</table>
Other IP Fields

- **Version**: 4 (IPv4) for most packets, there’s also IPv6 (lecture 12)
- **Header length** (in case of options)
- **Type of Service** (difserv, we won’t go into this)
- **Protocol identifier** (UDP: 17, TCP: 6, ICMP:1, why is TCP earlier?)
- **Checksum** over the header
- Let’s look at a packet with wireshark
Fragmentation & Reassembly

- Each network has some maximum transmission unit (MTU)
- **Strategy**
  - Fragment when necessary (MTU < size of Datagram)
  - Source host tries to avoid fragmentation
    - When fragment is lost, whole packet must be retransmitted!
  - Re-fragmentation is possible
  - Fragments are self-contained datagrams
  - Delay reassembly until destination host
  - Do not recover from lost fragments
• Ethernet MTU is 1,500 bytes

• PPP MTU is 576 bytes
  - R2 Must fragment IP packets to forward them
Fragmentation example (continued)

- IP addresses plus ident field identify fragments belonging to same packet
- MF (more fragments) bit is 1 in all but last fragment
- Fragment size multiple of 8 bytes
  - Multiply offset field by 8 to get fragment position within original packet
TCP Path MTU discovery

- **Problem:** How does TCP know what MSS to use?
  - On local network, obvious, but for more distant machines?

- **Solution:** Exploit ICMP—another protocol on IP
  - ICMP for control messages, not intended for bulk data
  - IP supports **DF** (don’t fragment) bit in IP header
  - Set DF to get ICMP can’t fragment when segment too big

- **Can do binary search on packet sizes**
  - But better: Base algorithm on most common MTUs
  - Common algorithm may underestimate slightly (better than overestimating and losing packet)
  - See **RFC1191** for details

- **Is TCP a layer on top of IP?**
IP Address Format, Translation, and DHCP
Format of IP addresses

• Globally unique (or made to seem that way)

• Hierarchical: network + host
  - Aggregating addresses saves memory in routers, simplifies routing (as we will see next lecture)

• Originally, routing prefix embedded in address:

  (Still hear “class A,” “class B,” “class C”)

• Now, routing info on “CIDR” blocks, addr+prefix-len
  - E.g., 171.67.0.0/16
Translating IP to lower-level addresses

- Map IP addresses into physical addresses
  - E.g., Ethernet address of destination host
  - Or Ethernet address of next hop router

- Techniques
  - Encode link layer address in host part of IP address
    (option is available, but only in IPv6)
  - Each network node maintains a lookup table (link→IP)

- ARP – *address resolution protocol*
  - Table of IP to link layer address bindings
  - Broadcast request if IP address not in table
  - Everybody learns physical address of requesting node (broadcast)
  - Target machine responds with its link layer address
  - Table entries are discarded if not refreshed
Need for Address Translation

- **Layer 2 (link) address** names a hardware interface
  - E.g., my wireless ethernet 00:26:b0:f9:25:cf

- **Layer 3 (network) address** names a host
  - E.g., www06.stanford.edu is 171.67.216.19
  - (lecture 8 will explain mapping from name to IP)

- **Details:**
  - A single host can have multiple hardware interfaces, so multiple link layer addresses for a single network address
  - A node is asked to forward a packet to another IP address: out which hardware interface does it send the packet?
## Arp Ethernet packet format

<table>
<thead>
<tr>
<th>0</th>
<th>8</th>
<th>16</th>
<th>31</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardware type = 1</td>
<td>ProtocolType = 0x0800</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HLen = 48</td>
<td>PLen = 32</td>
<td>Operation</td>
<td></td>
</tr>
<tr>
<td>SourceHardwareAddr (bytes 0–3)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SourceHardwareAddr (bytes 4–5)</td>
<td>SourceProtocolAddr (bytes 0–1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SourceProtocolAddr (bytes 2–3)</td>
<td>TargetHardwareAddr (bytes 0–1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TargetHardwareAddr (bytes 2–5)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TargetProtocolAddr (bytes 0–3)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Internet Control Message Protocol (ICMP)

- Echo (ping)
- Redirect (from router to source host)
- Destination unreachable (protocol, port, or host)
- TTL exceeded (so datagrams don’t cycle forever)
- Checksum failed
- Reassembly failed
- Cannot fragment

- Many ICMP messages include part of packet that triggered them
  - Example: Traceroute
### ICMP message format

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 |
| 20-byte IP header  |
| (protocol = 1—ICMP)  |

<table>
<thead>
<tr>
<th>Type</th>
<th>Code</th>
<th>Checksum</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Types include:
  - echo, echo reply, destination unreachable, time exceeded, …
  - See [http://www.iana.org/assignments/icmp-parameters](http://www.iana.org/assignments/icmp-parameters)
### Example: Time exceeded

```
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
```

<table>
<thead>
<tr>
<th>Type</th>
<th>Code</th>
<th>Checksum</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>unused</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

20-byte IP header
(protocol = 1—ICMP)

- IP header + first 8 payload bytes
  of packet that caused ICMP to be generated

- Code usually 0 (TTL exceeded in transit)

- Discussion: How does traceroute work?
Recall: UDP packet format

- First 8 bytes of UDP packet is UDP header
  - Which is conveniently included in ICMP packets
DHCP

- Hosts need IP addrs for their network interfaces
- Sometimes assign manually (but this is a pain)
- Or use Dynamic Host Configuration Protocol
  - Client broadcasts *DHCP discover* message
  - One or more DHCP servers send back *DHCP offer*
    - Sent to offered IP address (client hasn’t accepted yet)
    - But sent to client’s Ethernet address (not broadcast)
  - Client picks one offer, broadcasts *DHCP request*
  - Server replies with *DHCP ack*

- Discussion: why also a gateway and netmask?
IP Forwarding
Forwarding

- IP routers have multiple input/output ports

- **Note distinction between forwarding and routing**
  - Forwarding is passing packets from input to output port
  - Routing is figuring out the rules for mapping packets to output ports (topic of next two lectures)

- IP forwarding maps packet to output port based on destination address
  - Operates at network layer, not link layer
  - May forward between different kinds of networks (E.g., Ethernet on one side, cable TV wire on the other)
  - Does certain required processing on network-layer header (TTL, etc.)
Big Picture

Communication Network

Switched Communication Network

Circuit-switched    Packet-switched

Datagram    Virtual Circuit

Broadcast Communication Network
Physical Circuit Diversion: Old PSTN

- A telephone number is a program
- Number sets up a physical wire connection to another phone
- Old phones used to click...
Virtual Circuit Switching

- **Explicit connection setup (and tear-down) phase**
  - Establishes virtual-circuit ID (VCI) on each link

- **Each switch maintains VC table**
  - Switch maps \( \langle \text{in-link, in-VCI} \rangle \rightarrow \langle \text{out-link, out-VCI} \rangle \)
  - Subsequent packets follow established circuit

- **Sometimes called** connection-oriented model
Datagram switching

- No connection setup phase
  - Switches have routing table based on node addresses
- Each packet forwarded independently
- Sometimes called connectionless model
Source routing

- Simple way to do datagram switching (punt forwarding decisions to the sender)
Virtual Circuit Model

− Typically wait full RTT for connection setup before sending first data packet

+ Each data packet contains only a small identifier, making the per-packet header overhead small

− If a switch or a link in a connection fails, the connection is broken and a new one needs to be established

+ Connection setup provides an opportunity to reserve resources

+ Packets to the same destination can use different circuits
Datagram Model

+ There is no round trip time delay waiting for connection setup; a host can send data as soon as it is ready
  - Source host has no way of knowing if the network is capable of delivering a packet or if the destination host is even up
+ It is possible to route around failures
  - Overhead per packet is higher than for the connection-oriented model
  - All packets to the same destination must use the same path
2-minute stretch
Switch Fabrics
Cut through vs. store and forward

- Two approaches to forwarding a packet
  - Receive a full packet, then send it on output port
  - Start retransmitting as soon as you know output port, before you have even received the full packet (cut-through)

- Cut-through routing can greatly decrease latency

- Disadvantage: Can’t always send useful packet
  - If packet corrupted, won’t check CRC till after you started transmitting
  - Or if Ethernet collision, may have to send runt packet on output link, wasting bandwidth
Generic hardware switching architecture

- Goal: deliver packets from input to output ports
- Three potential performance concerns:
  - Throughput in terms of bytes/time
  - Throughput in terms of packets/time
  - Latency
• **Shared bus – like your PC**
  - NIC DMAs packet to memory over I/O bus
  - CPU examines pkt header, sends to dest NIC over bus
  - I/O bus is serious bottleneck
  - For small packets, CPU may be limited, too

• **Shared memory – similar, has memory bottleneck**
Crossbar switch

- One [vertical] bus per input interface
- One [horizontal] bus per output interface
- Can connect any input to any output
  - Trivially allows any input→output permutation
  - But, expensive for large number of inputs/outputs
Self-routing switches

- Idea: Build up switch out of $2 \times 2$ elements
- Each packet contains a “self-routing header”
  - For each switch along the way, specifies the output
- Must somehow compute a path when introducing packet
  - Is there more than one path to chose from?
  - Will path collide with another packet?
- Easy to implement stages once path computed
Banyan networks

• A Banyan network has exactly one path from any input port to a given output port
  - Example: Each stage can flip one bit of the port number

• Easy to compute paths

• Problem: Not all permutations can be routed
  - Might want 1 → 0 and 7 → 1, but both paths use same link

• But: Can always route packets if sorted
  - Leads to batcher banyan networks
  - Batcher phase sorts packets before banyan
Example: Banyan network

Switch on middle bit
Switch on high bit
Switch on low bit
Where to buffer?

- At some point more than one input port will have packets for the same output port
- Where do you buffer the packet?
  - Input port
  - Output port
Emerging technology: optical switches

• Already analog optical repeaters deployed
  - Will amplify any signal
  - Can change your low-level transmission protocol w/o replacing repeaters

• Could possibly do the same thing for switching
  - Microscopic mirrors can redirect light to different ports
  - (The ultimate cut-through routing)

• Technology exists, but not widely deployed
  - Optical switch will not see packet headers
  - Instructions on where to send packet need to be out-of-band
Bisection Bandwidth
Bisection bandwidth

- Can speak of the bandwidth between sets of ports
  - Bandwidth is maximum achievable aggregate bandwidth between the two sets

- **Bisection bandwidth** is important property of network
  - Lowest possible bandwidth between equal-sized sets of ports
  - Or almost equal-sized if odd number of ports

- A network with bad bisection bandwidth may offer poor behavior
  - Even if no conflict between input and output link utilization, may have internal bottlenecks reducing throughput
Example: Poor bisection bandwidth

- Connect two Ethernet switches with Ethernet
  - Suppose all clients on left, and all servers on right. . .
  - Aggregate bandwidth between all clients and servers only 100Mbit/s
Example: Poor bisection bandwidth 2

- Remember it’s worst case cut
  - Even with one fat link, don’t have to slice down middle
  - Put fat link in one partition, and bisection b/w very small

- Bisection bandwidth is a big concern in data center networks: more in lecture 16
Overview

- **Internet Protocol (v4)**
  - What it provides and its header
  - Fragmentation and assembly

- **IP Addresses**
  - Format and assignment: class-A, class-B, CIDR
  - Mapping, translation, and DHCP

- **Packet forwarding, circuits, source routing**

- **Switch fabrics**

- **Bisection bandwidth**

- **Next lecture: how TCP works**