Topics for This Week

• Address Resolution
• Network Service Models
  – Datagram
  – Virtual Circuit

• Readings
  – Sections 5.1, 5.6.1-5.6.3
Address Resolution

• IP address is **virtual**
  – Not understood by underlying physical networks

• IP packets need to be transmitted by the underlying physical network

• Address resolution
  – Translating IP address to physical address
  – Address Resolution Protocol (ARP)
ARP Cache

• Each computer maintains a cache table
  – IP address → hardware address address mapping
  – Only about computers on the same network
  – Try out “/usr/sbin/arp –a” command

• Exchanges ARP messages
  – To resolve IP addresses with unknown hardware addresses
ARP Protocol

• When a node sends an IP packet
  – To another node on the same physical network
• Look up destination address in the ARP table
• If not found
  – Broadcast a request to the local network
  – Whose IP address is this?
• What info should the request message contain?
## ARP Message

<table>
<thead>
<tr>
<th>HARDWARE ADDRESS TYPE</th>
<th>PROTOCOL ADDRESS TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>HADDR LEN</td>
<td>PADDR LEN</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>OPERATION</td>
<td></td>
</tr>
<tr>
<td>SENDER HADDR (first 4 octets)</td>
<td></td>
</tr>
<tr>
<td>SENDER HADDR (last 2 octets)</td>
<td>SENDER PADDR (first 2 octets)</td>
</tr>
<tr>
<td>SENDER PADDR (last 2 octets)</td>
<td>TARGET HADDR (first 2 octets)</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>TARGET HADDR (last 4 octets)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>TARGET PADDR (all 4 octets)</td>
</tr>
</tbody>
</table>
ARP Response

• The target node responds to sender (unicast?)
  – With its physical address
  – Adds the requester into its ARP table (why?)

• On receiving the response
  – Requester updates its table

• Other nodes upon receiving the request
  – Refresh the requester entry if already there
  – No action otherwise (why?)

• Table entries deleted if not refreshed for a while
ARP Example
Network Service Models

• Datagram
  – Packets forwarded independently
  – Connectionless

• Virtual Circuit (VC)
  – Packets of the same VC follow the same path
  – Need VC setup before packets can be sent
Network Layer Service: Datagram

- No notion of connection in network layer
  - No path or connection setup
  - Packets routed independently
- No guarantee of reliable or in-order delivery
  - Packet loss recovery at end-systems
- Advantages
  - No connection state in routers
  - Robust with respect to link failures
Datagram networks: the Internet model

• no call setup at network layer
• routers: no state about end-to-end connections
  – no network-level concept of “connection”
• packets typically routed using destination host ID
  – packets between same source-dest pair may take different paths
Case Study: IP

• IP datagram delivery model
• Each packet carries source and destination
• IP tries its best to deliver every packet
  – Best effort service
  – No guarantees
# IP Packet Format

<table>
<thead>
<tr>
<th>Field</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version (VERS)</td>
<td>0-3</td>
</tr>
<tr>
<td>Header Length (H. LEN)</td>
<td>4-7</td>
</tr>
<tr>
<td>Service Type (SERVICE TYPE)</td>
<td>8-11</td>
</tr>
<tr>
<td>Total Length (TOTAL LENGTH)</td>
<td>12-15</td>
</tr>
<tr>
<td>Identification (IDENTIFICATION)</td>
<td>16-19</td>
</tr>
<tr>
<td>Flags (FLAGS)</td>
<td>20-23</td>
</tr>
<tr>
<td>Fragment Offset (FRAGMENT OFFSET)</td>
<td>24-27</td>
</tr>
<tr>
<td>Time to Live (TIME TO LIVE)</td>
<td>28-31</td>
</tr>
<tr>
<td>Type (TYPE)</td>
<td>32-35</td>
</tr>
<tr>
<td>Header Checksum (HEADER CHECKSUM)</td>
<td>36-39</td>
</tr>
</tbody>
</table>

- **Source IP Address**: 32-47
- **Destination IP Address**: 48-63
- **IP Options (may be omitted)**: 64-79
- **Padding**: 80-87
- **Beginning of Data**

...
Forwarding/Routing IP Datagrams

• Routing and IP address
  – Routing based on network id
    • Only delivers packet to the appropriate network
    • Once on destination network, direct delivery using the host id

• IP destination-based next-hop routing paradigm
  – Each host/router has IP forwarding table
    • Entries like <network prefix, next-hop, output interface>
  – Try out “/usr/bin/netstat –rn” command

• Where does forwarding/routing table come from?
Getting a datagram from source to dest.

IP datagram:

<table>
<thead>
<tr>
<th>misc fields</th>
<th>source IP addr</th>
<th>dest IP addr</th>
<th>data</th>
</tr>
</thead>
</table>

• datagram remains unchanged, as it travels source to destination
• addr fields of interest here

<table>
<thead>
<tr>
<th>Dest. Net.</th>
<th>next router</th>
<th>Nhops</th>
</tr>
</thead>
<tbody>
<tr>
<td>223.1.1</td>
<td>223.1.1.4</td>
<td>1</td>
</tr>
<tr>
<td>223.1.2</td>
<td>223.1.1.4</td>
<td>2</td>
</tr>
<tr>
<td>223.1.3</td>
<td>223.1.1.4</td>
<td>2</td>
</tr>
</tbody>
</table>

routing table in A
Getting a datagram from source to dest.

| misc fields | 223.1.1.1 | 223.1.1.3 | data |

Starting at A, given IP datagram addressed to B:
- look up net. address of B
- find B is on same net. as A
- link layer will send datagram directly to B inside its frame
  - B and A are directly connected

<table>
<thead>
<tr>
<th>Dest. Net.</th>
<th>next router</th>
<th>Nhops</th>
</tr>
</thead>
<tbody>
<tr>
<td>223.1.1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>223.1.2</td>
<td>223.1.1.4</td>
<td>2</td>
</tr>
<tr>
<td>223.1.3</td>
<td>223.1.1.4</td>
<td>2</td>
</tr>
</tbody>
</table>
Getting a datagram from source to dest.

Starting at A, dest. E:
- look up network address of E
- E on different network
  - A, E not directly attached
- routing table: next hop router to E is 223.1.1.4
- link layer sends datagram to 223.1.1.4 inside its frame
- datagram arrives at 223.1.1.4
Getting a datagram from source to dest.

Arriving at 223.1.4, destined for 223.1.2.2

- look up network address of E
- E on same network as router’s interface 223.1.2.9
  - router, E directly attached
- link layer sends datagram to 223.1.2.2 inside link-layer frame via interface 223.1.2.9
- datagram arrives at 223.1.2.2!!! (hooray!)
Network Layer Service: Virtual Circuit

• Connection-oriented network service
  – Virtual circuit: looks like a circuit but isn’t

• All packets with the same VC or connection
  – Follow the same route

• Establishment of VC
  – Setup request flows from sender to receiver
  – Forwarding tables updated at intermediate nodes
Pros and Cons of Virtual Circuit

- Key issue: Per-VC state at each router/switch
- Suitable for traffic engineering
  - Multipath routing between source-destination pair
- Can support Quality of Service
  - Reserve resources per VC
  - Accept/Reject VC setup request based on resource availability along a path
Virtual Circuit: How Does It Work

• Two phases

• VC setup before data transmission
  – Signaling to setup forwarding table

• Packet transmission after VC has been setup
  – Each router looks up forwarding table
    • Finds the outgoing port using incoming VCI
    • Performs incoming VCI to outgoing VCI translation
Virtual circuits: signaling protocols

- used to setup, maintain, teardown VC
- used in ATM, frame-relay, X.25
- not used in today’s Internet
### Forwarding Table (VCI Translation Table)

<table>
<thead>
<tr>
<th>Input Port</th>
<th>Input VCI</th>
<th>Output Port</th>
<th>Output VCI</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

### Input/Output VCI Translation during Packet Transmission
Virtual Circuit Setup

• Select a path from source to destination
• Send VC setup request control packet
• Each router along the path
  – Choose a local VC id (VCI) for the connection
    • Need to ensure that no two distinct VCs leaving the same output port have the same VCI
  – Update forwarding table
    • Mapping between incoming VCI & port no. and outgoing VCI & port no.
Case Study: ATM Networks

• Asynchronous Transfer Mode
  – Single technology for handling voice, video, and data

• Connection-oriented service using virtual circuits
  – In-sequence but unreliable

• Cell switching using fixed-size cells: 53 bytes
  – Statistical multiplexing of cells of different circuits

• Provide QoS guarantees/assurance
  – Variety of services such as CBR, VBR, ABR etc
ATM Cell Format

<table>
<thead>
<tr>
<th>Bits:</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>FLOW CONTROL</td>
<td>VPI (FIRST 4 BITS)</td>
<td>VPI (LAST 4 BITS)</td>
<td>VCI (FIRST 4 BITS)</td>
<td>VCI (MIDDLE 8 BITS)</td>
<td>VCI (LAST 4 BITS)</td>
<td>PAYLOAD TYPE</td>
<td>PRIO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CYCLIC REDUNDANCY CHECK</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>48 DATA OCTETS START HERE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Virtual Circuit Switching

- VCI: 16 bits, local to a link
- VCI of each VC must be unique on each link
- Incoming VCI to outgoing VCI translation
  - Using a forwarding table
  - (in VCI, in port) \(\rightarrow\) (out VCI, out port)
VC Switching Example
Virtual Paths and VP Switch

- Why use Virtual Paths (VPs)?
- VCs of different VPs can have same VCIs
- VPI/VCI translation
  - Cells are routed using VPI/VCI pairs in the header
- VP Switch
  - Routing based on VPI only, VCI not translated
Datagram vs Virtual Circuit

Internet
• data exchange among computers
  – “elastic” service, no strict timing req.
• “smart” end systems (computers)
  – can adapt, perform control, error recovery
  – simple inside network, complexity at “edge”
• many link types
  – different characteristics
  – uniform service difficult

ATM
• evolved from telephony
• human conversation:
  – strict timing, reliability requirements
  – need for guaranteed service
• “dumb” end systems
  – telephones
  – complexity inside network
Datagram or Virtual Circuit?

• Burning question: to VC or not to VC?
  – Support both service models
    • Best effort service: datagrams
    • QoS guarantees: virtual circuits

• New IP Forwarding Paradigm
  – Multiple Protocol Label Switching
  – VC-based layer 2+1/2 switching
    • Resides between layer 2 & 3
  – For traffic engineering and QoS