ARP, IP, and TCP
IP and MAC Addresses

• Devices on a local area network have
  – IP addresses (network layer)
  – MAC addresses (data link layer)
• IP addresses are used for high level protocols
• MAC addresses are used for low level protocols
• How to translate IP Addresses in MAC addresses?
Address Resolution Protocol (ARP)

• Connects the network layer to the data link layer
• Maps IP addresses to MAC addresses
• Based on broadcast messages and local caching
• Does not support confidentiality, integrity, or authentication
• Defined as a part of RFC 826
ARP Messages

• ARP broadcasts requests of type who has <IP addressC> tell <IP addressA>
• Machine with <IP addressC> responds <IP addressC> is at <MAC address>
• Requesting machine caches response
• Network administrator configures IP address and subnet on each machine
ARP Cache

• The Linux, Windows and OSX command `arp - a` displays the ARP table

<table>
<thead>
<tr>
<th>Internet Address</th>
<th>Physical Address</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>128.148.31.1</td>
<td>00-00-0c-07-ac-00</td>
<td>dynamic</td>
</tr>
<tr>
<td>128.148.31.15</td>
<td>00-0c-76-b2-d7-1d</td>
<td>dynamic</td>
</tr>
<tr>
<td>128.148.31.71</td>
<td>00-0c-76-b2-d0-d2</td>
<td>dynamic</td>
</tr>
<tr>
<td>128.148.31.75</td>
<td>00-0c-76-b2-d7-1d</td>
<td>dynamic</td>
</tr>
<tr>
<td>128.148.31.102</td>
<td>00-22-0c-a3-e4-00</td>
<td>dynamic</td>
</tr>
</tbody>
</table>

• Command `arp -a -d` flushes the cache ARP

• ARP cache entries are stored for a configurable amount of time
From the LAN to the Internet

LAN

192.168.0.5
GTW 192.168.0.1
SN 255.255.255.0

192.168.0.6
GTW 192.168.0.1
SN 255.255.255.0

192.168.0.1

router

138.16.160.252

Internet

4/11/17

ARP, IP and TCP
Brown’s IP Space

• Brown separates the network connecting dorms and the network connecting offices and academic buildings

• Dorms
  – Class B network 138.16.0.0/16 (64K addresses)

• Academic buildings and offices
  – Class B network 128.148.0.0/16 (64K addresses)

• CS department
  – Several class C (/24) networks, each with 254 addresses
  – Tstaff supported machines: 128.148.31.0/24, 128.148.33.0/24, 128.148.38.0/24
  – Unsupported machines: 128.148.36.0/24
User Datagram Protocol

• UDP is a stateless, unreliable datagram protocol built on top of IP, that is it lies on level 4
• It does not provide delivery guarantees, or acknowledgments, but is significantly faster (i.e. dns uses on port 53)
• Can however distinguish data for multiple concurrent applications on a single host.
• A lack of reliability implies applications using UDP must be ready to accept a fair amount of error packages and data loss.
  – Most applications used on UDP will suffer if they have reliability. VoIP, Streaming Video and Streaming Audio all use UDP.
Transmission Control Protocol

• TCP is a transport layer protocol for reliable data transfer, in-order delivery of messages and ability to distinguish multiple applications on same host
• WWW, FTP and SSH are built on top of TCP
• TCP takes a data stream, packages it into segments and calls on IP to transmit these packets
• Delivery order is maintained by marking each packet with a sequence number
• Every time TCP receives a packet, it sends out an ACK to indicate successful receipt of the packet.
• TCP generally checks data transmitted by comparing a checksum of the data with a checksum encoded in the packet
Ports

• TCP supports multiple concurrent applications on the same server
• Ports are 16 bit numbers identifying where data is directed
• The TCP header includes space for both a source and a destination port, thus allowing TCP to route all data
• In most cases, both TCP and UDP use the same port numbers for the same applications
• Ports 0 through 1023 are reserved for use by known protocols.
• Ports 1024 through 49151 are known as user ports, and should be used by most user programs for listening to connections and the like
• Ports 49152 through 65535 are private ports used for dynamic allocation by socket libraries
## TCP Packet Format

<table>
<thead>
<tr>
<th>Bit Offset</th>
<th>0-3</th>
<th>4-7</th>
<th>8-15</th>
<th>16-18</th>
<th>19-31</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Source Port</td>
<td></td>
<td></td>
<td>Destination Port</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td></td>
<td></td>
<td></td>
<td>Sequence Number</td>
<td></td>
</tr>
<tr>
<td>64</td>
<td></td>
<td></td>
<td></td>
<td>Acknowledgment Number</td>
<td></td>
</tr>
<tr>
<td>96</td>
<td>Offset</td>
<td>Reserved</td>
<td>Flags</td>
<td>Window Size</td>
<td></td>
</tr>
<tr>
<td>128</td>
<td></td>
<td></td>
<td>Checksum</td>
<td></td>
<td>Urgent Pointer</td>
</tr>
<tr>
<td>160</td>
<td></td>
<td></td>
<td></td>
<td>Options</td>
<td></td>
</tr>
<tr>
<td>&gt;= 160</td>
<td></td>
<td></td>
<td></td>
<td>Payload</td>
<td></td>
</tr>
</tbody>
</table>

**Legend:**
- Source Port: 0-3
- Destination Port: 16-18
- Sequence Number: 32
- Acknowledgment Number: 64
- Window Size: 96
- flags: 64-95
- Offset: 0-15
- Reserved: 16-63
- Checksum: 64-95
- Options: 128-159
Establishing TCP Connections

- TCP connections are established through a three way handshake.
- The server generally has a passive listener, waiting for a connection request.
- The client requests a connection by sending out a SYN packet.
- The server responds by sending a SYN/ACK packet, indicating an acknowledgment for the connection.
- The client responds by sending an ACK to the server thus establishing a connection.

SYN
Seq = x

SYN-ACK
Seq = y
Ack = x + 1

ACK
Seq = x + 1
Ack = y + 1
TCP Data Transfer

- During connection initialization using the three way handshake, initial sequence numbers are exchanged.
- The TCP header includes a 16 bit checksum of the data and parts of the header, including the source and destination.
- Acknowledgment or lack thereof is used by TCP to keep track of network congestion and control flow and such.
- TCP connections are cleanly terminated with a 4-way handshake:
  - The client which wishes to terminate the connection sends a FIN message to the other client.
  - The other client responds by sending an ACK.
  - The other client sends a FIN.
  - The original client now sends an ACK, and the connection is terminated.
TCP Data Transfer and Teardown

Data
seq=x

Ack seq=x+1

Data
seq=y

Ack seq=y+1

Client

Server

Fin seq=x

Ack seq=x+1

Fin seq=y

Ack seq=y+1

Client

Server