Administrative Stuff

• iPAQs handed out after class today!
  – CSE 391, 10:30-noon
  – Load Fishnet 1 solution, bring to class next Wed.
• Fishnet 2 out next Wed.
  – Routing: How can we get packets to their destinations more efficiently than with flooding?
  – Neighbor discovery is a start on this.

Last time

• How do we decide when to retransmit?

• Topics
  – RTT estimation
  – Karn/Partridge algorithm
  – Jacobson/Karels algorithm

This Lecture

• How can we connect processes?

• Topics
  – The transport layer
  – How can we name processes?
  – How can we reliably start & end conversations?
The Transport Layer

- Provide communication between applications running on hosts
- Provide abstractions that are useful to applications

Internet Transport Protocols

- UDP
  - Datagram abstraction between processes
- TCP
  - Bytestream abstraction between processes
  - Reliability
  - In-order delivery
  - Flow control
  - Congestion control (later)

Naming Processes

- Process here is an abstract term for a service or a client
  - e.g.

- How can we identify processes for remote communications?
  - OS: PID, memory address
  - Transport protocols: port number

What’s a port number?

- 16-bit integer representing mailbox that processes “rent”
- Identify process uniquely as

- You are using port numbers in Fishnet!
  - ruby trawler.rb 9000 topo
  - ruby fishnet.rb 10000 host:9000
Picking Port Numbers

- What port should a web server process use?
- What about a web browser?

- Servers typically bind to “well-known” port numbers
  - e.g. HTTP 80, SMTP 25, SSH 22,…
  - Under 1024 reserved for “well-known” services
- Clients use OS-assigned temporary ports

User Datagram Protocol (UDP)

- Provides message delivery between processes
  - Source port:
  - Destination port:

```
+--------+     +--------+
| SrcPort|   ->  | DstPort|
+--------+     +--------+
| Checksum|   ->  | Length|
+--------+     +--------+
| Data   |       |        |
+--------+     +--------+
```

UDP Delivery

```
+-------------------------+-------------------------+-------------------------+
|                       |                       |                       |
| Application process    | Application process    | Application process    |
| Message Queues         | Message Queues         | Message Queues         |
| Kernel boundary        | Kernel boundary        | Kernel boundary        |
| DeMux                  | DeMux                  | DeMux                  |
+-------------------------+-------------------------+-------------------------+
| Ports                  | Ports                  | Ports                  |
| Packets arrive         | Packets arrive         | Packets arrive         |
```

UDP Checksum

- UDP includes optional error detection
  - Is this message really for me?
  - Checksum covers IP pseudoheader, UDP header, data

```
+--------+     +--------+
| SrcPort|   ->  | DstPort|
+--------+     +--------+
| Checksum|   ->  | Length|
+--------+     +--------+
| Data   |       |        |
+--------+     +--------+
```
TCP Delivery

Application process

Write bytes

TCP

Send buffer

Transmit segments

TCP

Receive buffer

Application process

Read bytes

TCP Header Format

Options (variable)

Data

SrcPort  0 4 10 16 31

SequenceNum

Acknowledgment

HdrLen  0  Flags  AdvertisedWindow

Checksum  UrgPtr

Options (variable)

Data

TCP Header Format

Options (variable)

Data

SrcPort  0 4 10 16 31

SequenceNum

Acknowledgment

HdrLen  0  Flags  AdvertisedWindow

Checksum  UrgPtr

Options (variable)

Data
TCP Header Format

<table>
<thead>
<tr>
<th>0</th>
<th>4</th>
<th>10</th>
<th>16</th>
<th>21</th>
</tr>
</thead>
<tbody>
<tr>
<td>SrcPort</td>
<td>DstPort</td>
<td>SequenceNum</td>
<td>Acknowledgment</td>
<td>Flags</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Flags may be ACK, RST, SYN, FIN, URG, PSH

Connection Establishment

- Both sender and receiver must be ready before we start to transfer the data
  - Sender and receiver need to agree on a set of parameters
  - e.g., the Maximum Segment Size (MSS)
- This is signaling
  - It sets up state at the endpoints
  - Compare to “dialing” in the telephone network

Three-Way Handshake

Active participant (client)

Passive participant (server)

SYN, SequenceNum =

SYN = ACK, SequenceNum =

ACK, Acknowledgment =

+data

Some Comments

- Three way handshake was chosen to be robust
  - especially against delayed duplicates

- Initial sequence number (ISN) is randomized
  - Minimizes the chance of getting confused by packets from a previous incarnation of a connection
## TCP State Transitions

- **Passive open**
- **Active open**
- **SYN_SENT**
- **ESTABLISHED**
- **CLOSED**
- **FIN_WAIT_2**
- **FIN_WAIT_1**
- **CLOSING**
- **TIME_WAIT**
- **LAST_ACK**
- **CLOSE_WAIT**
- **SYN_RCVD**
- **LISTEN**
- **SYN + ACK**
- **FIN/ACK**
- **ACK + FIN**
- **FIN/ACK**
- **ACK**
- **SYN + ACK**

### TCP State Transitions Diagram

1. Passive open -> Close
2. Active open -> SYN
3. SYN/SYN + ACK -> Close
4. SYN + ACK/ACK -> ESTABLISHED
5. CLOSE_WAIT -> LAST_ACK
6. LAST_ACK -> TIME_WAIT
7. TIME_WAIT -> CLOSE

---

## Again, with States

- **Active participant (client)**
- **Passive participant (server)**

### States Diagram

- **SYN, SequenceNum = x**
- **SYN + ACK, SequenceNum = y**
- **ACK, Acknowledgment = y + 1**
- **ACK, Acknowledgment = x + 1**

---

## Connection Teardown

- Orderly release by sender and receiver
  - Delivers all pending data and “hangs up”
  - Cleans up state in sender and receiver

- TCP provides a “symmetric” close
  - both sides shutdown independently

---

## TCP Connection Teardown

- **Web server**
  - FIN
  - ACK
- **Web browser**
  - FIN
  - ACK
The TIME_WAIT State

- We wait 2MSL (two times the maximum segment lifetime of 60 seconds) before completing the close
- Why?

Key Concepts

- We use ports to name processes in TCP/UDP
  - “Well-known” ports are used for popular services
- Connection setup and teardown complicated by the effects of the network on messages
  - TCP uses a three-way handshake to set up a connection
  - TCP uses a symmetric disconnect