CNT 4007 Computer Networks
- Chapter 3: Transport Layer

Jonathan Kavalan, Ph.D.
Department of Computer, Information Science and Engineering (CISE),
University of Florida
Our goals:

- understand principles behind transport layer services:
  - multiplexing/de-multiplexing
  - reliable data transfer
  - flow control
  - congestion control

learn about transport layer protocols in the Internet:
- UDP: connectionless transport
- TCP: connection-oriented transport
- TCP congestion control
• provide *logical communication* between app processes running on different hosts

• transport protocols run in end systems
  – send side: breaks app messages into *segments*, passes to network layer
  – rcv side: reassembles segments into messages, passes to app layer

• more than one transport protocol available to apps
  – Internet: TCP and UDP
• **network layer:** logical communication between hosts
  
• **transport layer:** logical communication between processes
  - relies on, enhances, network layer services

**Household analogy:**

12 kids sending letters to 12 kids
processes = kids
app messages = letters in envelopes
hosts = houses
transport protocol = Ann and Bill
network-layer protocol = postal service
Internet transport-layer protocols

- reliable, in-order delivery (TCP)
  - congestion control
  - flow control
  - connection setup
- unreliable, unordered delivery: UDP
  - no-frills extension of “best-effort” IP
- services not available:
  - delay guarantees
  - bandwidth guarantees
Demultiplexing at rcv host: delivering received segments to correct socket

Multiplexing at send host: gathering data from multiple sockets, enveloping data with header (later used for demultiplexing)

- = socket  = process

<table>
<thead>
<tr>
<th>application</th>
<th>P3</th>
</tr>
</thead>
<tbody>
<tr>
<td>transport</td>
<td></td>
</tr>
<tr>
<td>network</td>
<td></td>
</tr>
<tr>
<td>link</td>
<td></td>
</tr>
<tr>
<td>physical</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>application</th>
<th>P1</th>
<th>application</th>
<th>P2</th>
</tr>
</thead>
<tbody>
<tr>
<td>transport</td>
<td></td>
<td>network</td>
<td></td>
</tr>
<tr>
<td>link</td>
<td></td>
<td>link</td>
<td></td>
</tr>
<tr>
<td>physical</td>
<td></td>
<td>physical</td>
<td></td>
</tr>
</tbody>
</table>

host 1

host 2

host 3
How demultiplexing works

- host receives IP datagrams
  - each datagram has source IP address, destination IP address
  - each datagram carries 1 transport-layer segment
  - each segment has source, destination port number

- host uses IP addresses & port numbers to direct segment to appropriate socket

TCP/UDP segment format

<table>
<thead>
<tr>
<th>source port #</th>
<th>dest port #</th>
</tr>
</thead>
<tbody>
<tr>
<td>other header fields</td>
<td></td>
</tr>
<tr>
<td>application data (message)</td>
<td></td>
</tr>
</tbody>
</table>
Connectionless demultiplexing

- Create sockets with port numbers:
  ```java
  DatagramSocket mySocket1 = new DatagramSocket(12534);
  DatagramSocket mySocket2 = new DatagramSocket(12535);
  ```

- UDP socket identified by two-tuple:
  
  \((\text{dest IP address}, \text{dest port number})\)

When host receives UDP segment:
- checks destination port number in segment
- directs UDP segment to socket with that port number

IP datagrams with different source IP addresses and/or source port numbers directed to same socket
DatagramSocket serverSocket = new DatagramSocket(6428);

SP provides "return address"
Connection-oriented demux

- TCP socket identified by 4-tuple:
  - source IP address
  - source port number
  - dest IP address
  - dest port number

- recv host uses all four values to direct segment to appropriate socket

- Server host may support many simultaneous TCP sockets:
  - each socket identified by its own 4-tuple

- Web servers have different sockets for each connecting client
  - non-persistent HTTP will have different socket for each request
Connection-oriented demux

Client
IP: A

SP: 9157
DP: 80
S-IP: A
D-IP: C

Server
IP: C

SP: 9157
DP: 80
S-IP: B
D-IP: C

Client
IP: B

P1

P2

P3

P4

P5

P6

SP: 5775
DP: 80
S-IP: B
D-IP: C
Connection-oriented demux: Threaded Web Server