Transport Layer Overview

Kai Shen
Dept. of Computer Science, University of Rochester

Network layer: host-to-host logical communication between hosts.

Transport layer: logical communication between applications:
- multiple comm. applications can reside in one host.
- comm. applications can be a Web browser/server, a FTP client/server, etc.

Transport layer: end-host-only implementation.

Multiplexing/demultiplexing

P1 on host A is communicating with P3 on host B; while at the same time, P2 on host A is communicating with P4 on host B.

Transport-layer Service Model

Transport layer: logical communication between applications:
- multiplexing/demultiplexing

Additional services:
- reliable data transfer (guaranteed arrival, no error, in-order)
- flow control (keep sender from overrunning receiver): good for myself
- congestion control (keep sender from overrunning network): good for the community
- delay/bandwidth guarantee
- ...
Internet Transport-layer Protocols

- **UDP:**
  - multiplexing/demultiplexing
  - error detection

- **TCP:**
  - multiplexing/demultiplexing
  - error detection
  - reliable data transfer
  - flow control
  - congestion control

- Services not available:
  - delay guarantees
  - bandwidth guarantees
  - why?

Internet Transport-layer Protocols (cont.)

- What is a **connection**?
  - logical relationship among data segments between the same pair of comm. end-points
  - allow for shared resources, basis for services such as in-order delivery, flow control, et al.

- **UDP is connectionless:**
  - each UDP segment handled independently of others
  - datagram-based service

- **TCP is connection-oriented:**
  - TCP segments between the same pair of comm. end-points are considered as belonging to a connection
  - stream-based service

Outline

- Overview of transport-layer services/protocols
  - Multiplexing and demultiplexing
  - Connectionless transport: UDP
  - Principles of reliable data transfer
  - Connection-oriented transport: TCP
    - reliable data transfer
    - flow control
    - connection management
  - Principles of congestion control
  - Congestion control in TCP

How multiplexing/demultiplexing works?

- IP packet format
  - IP header
  - TCP/UDP header
  - application data

- Using port numbers
  - each IP packet has source IP address, destination IP address
  - each IP packet carries a transport-layer segment
  - each segment has source, destination port number

- Destination IP address for routing to the host; IP addresses and port numbers for going to appropriate socket in the dest. host.
UDP Demultiplexing: An Example

Machine A

Machine B

Machine C

UDP socket identified by port number

Port 1234

Port 5678

TCP Demultiplexing (Connection-oriented)

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

Why?
- each socket corresponds to a TCP connection
- Server at a port may support multiple simultaneous TCP connections/sockets
- Web server (on default port 80) has multiple sockets, one for each connecting client

UDP: User Datagram Protocol

UDP is connectionless
UDP does multiplexing/demultiplexing
simple error detection
UDP does not do
reliable data transfer, flow control, congestion control...

UDP segment format

<table>
<thead>
<tr>
<th>source port #</th>
<th>dest port #</th>
</tr>
</thead>
<tbody>
<tr>
<td>length</td>
<td>checksum</td>
</tr>
</tbody>
</table>

32 bits

Application data (message)
UDP Checksum

Goal: detect "errors" (e.g., flipped bits) in transmitted segments

Sender:
- treat segment contents as sequence of 16-bit integers
- checksum: addition (1's complement sum) of segment contents
- sender puts checksum value into UDP checksum field

Receiver:
- compute checksum of received segment
- check if computed checksum equals checksum field value:
  - NO: error detected
  - YES: no error detected.

What is good about UDP?

- TCP features may not be needed by some applications
- less overhead:
  - no connection establishment (which can add delay)
  - small segment header
  - no congestion control: UDP can blast away as fast as desired (may not be good for others, but not bad for my own connection 😊)
- simple: no connection state at sender, receiver

UDP: More

- Often used for streaming multimedia apps
  - loss tolerant
  - rate sensitive
- In general, UDP is used when TCP features are not important
- What if you want features not in UDP while the full TCP is overkill?
  - implemented at application-level
  - flow control and loss recovery in some multimedia apps

Disclaimer

Parts of the lecture slides contain original work of James Kurose, Larry Peterson, and Keith Ross. The slides are intended for the sole purpose of instruction of computer networks at the University of Rochester. All copyrighted materials belong to their original owner(s).