Transport Layer Overview

Kai Shen

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, an FTP client/server, etc.

Multiplexing/demultiplexing

Transport-layer Service Model

- Transport layer: logical communication between applications.
  - multiplexing/demultiplexing
- Possible 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
  - ... ...
Transport-layer in the Internet Architecture

- End-host only transport implementation
- Simple network layer, complex transport

Benefits:
- Flexible transport layer
- Fast, stable network core

Challenge:
- Transport functions without router assistance

Internet Transport-layer Protocols

- UDP: minimum service transport
  - multiplexing/demultiplexing
  - error detection
- TCP: full service transport
  - multiplexing/demultiplexing
  - error detection
  - reliable data transfer
  - flow control
  - congestion control

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

Connection-oriented /Connectionless Transport

- What is a connection?
  - logical grouping/sequencing of data segments for a comm. session between the same pair of end-points
  - state maintained for such session

- Why is a connection useful?
  - conn. state-based reliable communication
  - resource differentiation
  - track network performance using a history of performance measurement
  - ... ...

- UDP is connectionless
- TCP is connection-oriented

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
Multiplexing/demultiplexing

- Host A
  - Application
  - Transport
  - Network
  - Link
  - Physical
- Host B
  - Communication application
  - Socket

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.

How multiplexing/demultiplexing works?

- 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
  - Dest. 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

- UDP socket identified by dest. IP address and port number

TCP Demultiplexing (Connection-oriented)

- Multiple TCP connections may simultaneously share a single port:
  - Example: web server (on default port 80) has multiple connections, one for each client
- 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
### TCP Demultiplexing: An Example

- **client IP:** A
- **server IP:** C
- **client IP:** B

**P1:**
- **IP:** A
- **SP:** 9157
- **DP:** 80

**P2:**
- **IP:** A
- **SP:** 80
- **DP:** 9157

**P3:**
- **IP:** B
- **SP:** 80
- **DP:** 5775

**P4:**
- **IP:** C
- **SP:** 9157
- **DP:** 80

**P5:**
- **IP:** B
- **SP:** 5775
- **DP:** 80

**Listening socket:** all on port 80

---

### UDP: User Datagram Protocol

**UDP segment format**

- **Length:** 32 bits, in bytes of UDP segment, including header
- **source port #**
- **dest port #**
- **Application data (message)**
- **checksum**

---

### 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 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.

**Why are there error detections in transport, network, and link layers?**
What is good about UDP?

- Less overhead:
  - no running cost for connection management
  - no space cost for connection state
  - small segment header

- Flexibility in doing things your own way
  - 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 ⇒ more robust, less security vulnerability
  - less susceptible to denial-of-service attacks

UDP In Practice

- When to use UDP?
  - TCP services are not useful, or even hurtful
  - TCP services are useful, but not worth the cost
  - TCP services are useful in concept, but need to be customized

- Used for:
  - routing exchanges between routers
  - network performance measurements
  - streaming multimedia apps
    - loss tolerant, rate sensitive
    - flow control and loss recovery sometimes implemented at the application level

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).