The Transport Layer: TCP & Reliable Data Transfer

Smith College, CSC 249
February 17, 2017

Chapter 3: Transport Layer

- TCP Transport layer services:
  - Multiplexing/demultiplexing
  - Connection management
  - Reliable data transfer
    - SEQ and ACK numbers
  - Congestion control
  - Flow control

Error Checking: Checksum

Practice in HW - straightforward calculation

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

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

Receiver:
- compute checksum of received segment - including the sender's checksum 16-bit word in the sum
- If receiver's sum is all '1's then there were no errors (probably)
- If a bit is 0 then the packet has errors

Internet Checksum Example

Note
- When adding numbers, a carryout from the most significant bit needs to be added to the result, for 1's complement

Example: add two 16-bit integers

Sender:

Receiver:

Sum: 000000000000
Checksum: 100000000000

Wraparound: 011111101010101110

Practice in HW - straightforward calculation

Goal: detect "errors" (e.g., flipped bits) in transmitted segment
Checksum example

- Is it possible for a 1-bit error to go undetected?
- Is it possible for a 2-bit error to go undetected?

TCP Connection Management: Set up

Recall: TCP senders and receivers establish a “connection” before exchanging data segments

Three way handshake:

- **Step 1:** client host sends TCP SYN segment to server
  - “SYN” for “synchronize”
  - Specifies (random) initial sequence #
  - No data is sent
- **Step 2:** server host receives SYN, replies with SYNACK segment
  - Server allocates buffers and variables
  - Specifies its own, server initial sequence #
- **Step 3:** client receives SYNACK, replies with ACK segment
  - Client allocates buffers and variables
  - This packet may contain data

TCP segment structure

- **Source port #**
- **Destination port #**
- **Sequence number**
- **Acknowledgement number**
- **Data offset**: indicates number of bytes
- **Reserved**: always 0
- **Control flags**: URG, ACK, PSH, RST, SYN, FIN
  - **URG**: urgent data (generally not used)
  - **ACK**: ACK # valid
  - **PSH**: push data now (generally not used)
  - **RST**: reset connection (setup, teardown commands)
  - **SYN**: synchronize initial sequence #
  - **FIN**: end of connection
- **Window**: receiver willing to accept bytes of data
- **Checksum**: as in UDP
- **Urgent pointer**: points to byte following urgent data
- **Options**: variable length
- **Total length**: 32 bits
- **Header length**: 20 bits
- **Data**: variable length
- **Internet checksum**: as in UDP

Set Up:

- **Step 1:** client sends TCP SYN segment
  - Actions at self?
  - Sends data?
- **Step 2:** server receives SYN and responds with SYNACK
  - Actions at self?
  - Sends data?
- **Step 3:** client receives SYNACK and responds with ACK
  - Actions at self?
  - Sends data?
TCP Connection Management: Close

Closing a connection:

How many steps?
What are they?

TCP Connection Management (cont.)

Step 3: client receives FIN, replies with ACK.
- Enters “timed wait” - client able to resend final ACK in case it is lost
- Why??

Step 4: server, receives ACK. Connection closed.

TCP: SEQ and ACK numbers

Host A
Host B

SendBase = 100
SendBase = 120

Seq=20, 8 bytes data
Seq=100, 20 bytes data

ACK=100
ACK=120

Seq=20, 8 bytes data
Seq=100, 20 bytes data

ACK=100
ACK=120

SendBase = 100
SendBase = 120

time
TCP: retransmission scenarios

TCP segment structure

TCP possible sender events:

1. data received from application:
   1. Create a segment and assign a SEQ number
   2. Start timer if it is not already running
   3. Timer is for the oldest un-acked segment
   4. Expiration interval: TimeOutInterval

2. timeout:
   1. Retransmit segment that caused the timeout
   2. Restart the timer

3. ACK received:
   1. For previously unacked segments
      1. update what is known to be acked
      2. start timer if there are outstanding segments

1) What is/was ‘A’s next step?
2) What does ‘B’ then do?
TCP retransmission scenarios (more)

What does ‘A’ do next?

Discussion Question 1

- Suppose Host A sends two TCP segments back to back to Host B over a TCP connection.
  - What might be the first sequence number?
  - If 20 bytes are sent, what is the second sequence number?
  - Suppose that the first segment is lost but the second segment arrives at B. In the acknowledgment that Host B sends to Host A, what will be the acknowledgment number?

b) The random initial number, plus 20 bytes

c) ACK number = initial sequence number - i.e., “still waiting for the first segment”

Summary

- Error checking - checksum
- Connection Management
  - SYN and SYNAck packets
- TCP segment header format
- Reliable data transfer
  - Sequence and acknowledgement numbers
  - Use of “NAK”
  - Purpose of 3 duplicate ACKs