Agenda

- CRC calculation and Endianness
- Review Sliding Window Protocol
- Some debugging techniques
What is Endianness?

- Ordering of bytes within a data structure
  - 4 byte int, 0x01020304 can be stored in memory as
    - |01|02|03|04| -> MSByte first ("Biggest" byte first - big endian)
    - |04|03|02|01| -> LSByte first ("Littlest" byte first - little endian)
Why do we care about Endianness?

- Are these operations equivalent?
  a. `uint16_t x = ((uint16_t*) char_buf)[0];`
  b. `uint16_t x = char_buf[0] << 8 + char_buf[1];`
Why do we care about Endianness?

Are these operations equivalent?

a. `uint16_t x = ((uint16_t*) char_buf)[0];`
   ← Depends on endianness

b. `uint16_t x = char_buf[0] << 8 + char_buf[1];`
   ← Endian agnostic
Why do we care about Endianness?

- Are these operations equivalent?
  a. `uint16_t x = ((uint16_t*) char_buf)[0];`
     ← Depends on endianness
  b. `uint16_t x = char_buf[0] << 8 + char_buf[1];`
     ← Endian agnostic

- Part of CRC initialization steps.
Sliding Window Protocol
Sliding Window Protocol

- Improve link usage by not waiting for RTT between 2 packets.
**Sender Window**

- Contains un-ACKd frames.
- ACKs are cumulative.

- ← ACKd
- ← Sent but not ACKd
- ← Not sent yet
Sender Window

- Contains un-ACKd frames.
- (Outside window) ACKd
- (Inside window) Sent but not ACKd
- (Inside window) Not sent yet
- (Outside window) Not sent yet
- ACKs are cumulative.
Receiver Window

- Contains frames received out-of-order

- Received (and next packet received)
  - Received (next packet not received)
  - Not received
  - Will be accepted
  - Will not be accepted
Example - Initial step

Sender

Last ACKd

Receiver

Last Recvd

Last ACKd
Example - Initial step (corrected)

Sender

Receiver
Step 1 - Receiver gets #5

Sender

Receiver

Last Recvd

Last ACKd

Last sent

Last ACKd

Sending 5

1 2 3 4 5 6

1 2 3 4 5
Step 1 - Receiver gets #5 - outcome

Sender

Receiver

Receiver sends ACK for #5
Step 2 - Sender sends #7

Sender

Receiver
Step 2 - Receiver gets #7

Sender

Receiver
Step 2 - Receiver gets #7 - outcome

Sender

Receiver

Send Duplicate ACK[6]
Step 3 - Sender gets DupACK[6]

Sender

Receiver
Step 3 - Sender gets DupACK[6]

Resend packet 6
Step 4 - Receiver gets #6

Resend packet 6
Step 4 - Receiver gets #6 - outcome

Sender

Receiver

Resend packet 6
Debugging hands on
Q & A