Comp3331 Week 5 Lecture 1

Transport Layer part 2 - TCP

# Questions from previous lecture

* how do the sender and receiver keep track of outstanding pipelined segments?
* how many segments should be pipelined?
* how do we choose sequence number?
* what does connection establishment and teardown look like?
* how should we choose timeout values?

# Overview of TCP

* **point to point:** one sender, one receiver
* **reliable, in-order byte stream:** no "message boundaries", all in fixed size data packet
* **pipelined:** TCP congestion and flow control set window size
* **send and receive buffers:** TCP makes sure that buffer is not overflowed, like selective repeat
* **full duplex data:** maximum segment size
* **connection-oriented:** handshaking inits sender, receiver state before data exchange
* **flow controlled:** sender will not overwhelm receiver, tell the sender buffer status

# TCP Segment Structure

Diagram

Description automatically generated

**If we ignore the options, the TCP header is 20 bytes (compared to UDP 8 bytes)**

# TCP reliable data transfer

1. checksum: the checksum in TCP is exactly same as UDP checksum (able to check bit error)
2. Timers (for loss detection)
3. Cumulative Acknowledgments
4. Sequence numbers
5. Sliding Windows

# TCP Segment Size

Graphical user interface, application

Description automatically generated

MTU size is determined by the link layer

* IP packet: No bigger than Maximum Transmission Unit
* TCP packet: IP packet with a TCP header and data inside
* TCP segment: No more than Maximum Segment Size (MSS) bytes,

MSS = MTU – 20 (min IP header) – 20 (min TCP header)

# Sequence Number

* *Sequence number = 1st byte in segment = ISN + k*
* *ACK sequence number = next expected byte = seqno + length(data)*

Chart, bar chart

Description automatically generated

Alice’s end point selects the initial sequence number as 0 while Bob’s end point selects the initial sequence number as 10. Alice’s ACK is 10, this is what Alice is expecting from Bob. Alice sends a data of 5 bytes. Bob receives 5 bytes, responds back with ACK 5 (for hello). Bob sends back 4 bytes segment. The ACK number = sequence number + length of data

Chart

Description automatically generated with medium confidence

Connection establishment not shown. Alice’s end point selects the initial sequence number as 0 while Bob’s end point selects the initial sequence number as 10 HTTP response split into 3 segments (MSS = 1500 bytes)

**Why do we choose random ISN?**

* Avoid ambiguity with back-to-back connections between same endpoints
* Potential security issue if the ISN is known

# Cumulative ACKS (like GBN)

Chart

Description automatically generated with low confidence

The sequence number in this case is going to be 300 because ACK is 300, meaning 250 is successfully received even the ACK is lost. There is not going to be retransmission.

Diagram

Description automatically generated

# Piggybacking

In reality, usually both sides of a connection send some data, this is what we call piggybacking

Diagram

Description automatically generated

# TCP timeout estimate

Sender maintains a single retransmission timer (like GBN) and retransmits on timeout (how much?)

Text

Description automatically generated

Text

Description automatically generated

Timeline

Description automatically generated

**Why exclude retransmissions in RTT computation?**

Ambiguity at sample RTT below, we don’t want this to happen.

Diagram

Description automatically generated

# TCP retransmission and cumulative acks

Table

Description automatically generated

## Timeout

long delay before resending lost packet

## Triple duplicate acks

if sender receives 3 duplicate ACKs for same data (“triple duplicate ACKs”), resend unacked segment with smallest seq #