Transmission Control Protocol

TCP: Reliable Reusable Responsible

What is TCP?

- Connection-oriented
- Point-to-point protocol; one sender, one receiver
- Reliable connection, with in-order stream of data
- Pipelined
- Flow control
- Full-duplex
- Support for send/receive buffers

Advantages

- Sender and receiver know when a data connection is established or broken
- Data will be resent automatically if not received, without involvement of application
- Data always arrives in-order
- Pipelining allows congestion control
- Flow control ensures that sender cannot overwhelm receiver
- Supports sending and receiving at the same time over the same connection

How it works

- Handshaking – opening a connection
  - Step 1: Client sends TCP SYN segment, specifies initial sequence number
  - Step 2: Server receives SYN, replies with SYNACK segment
    - Server allocates buffers for connection
    - Server specifies initial sequence number
  - Step 3: Client receives SYNACK, replies with ACK. ACK may contain data.

How it works

- Closing a connection
  - Step 1: Client sends TCP FIN to server
  - Step 2: Server receives FIN, responds with ACK, closes connection, sends FIN
  - Step 3: Client receives FIN, replies with ACK
  - Step 4: Server receives ACK, closes connection

Congestion Management

- Congestion
  - Can result in lost packets and long delays
  - Management/TCP implementation:
    - No feedback needed from network
    - Congestion inferred by end-system observed loss and delay
  - Lost packets will be resent
  - Max network speed is found by decreasing window until first loss

Flow Control

- Flow control/Fairness
  - Multiple TCP connections sharing the same connection
    - TCP will ensure that connections get average equal bandwidth
    - Can be circumvented by requesting more TCP connections for the same task

Advantages

- Reliable connection at the transport layer
- Guarantees:
  - Data reaches destination
  - Data arrives in-order
  - Data is not duplicated
- Operating system provides an implementation for applications to use
- Controlled by the OS kernel, resulting in fewer context switches and better performance
- Routers may treat TCP packets with a higher priority
- Good relative throughput

Issues

- Speed
  - Faster physical networks
  - TCP code is implemented by OS
- Poor performance on bad links
  - UDP will also have poor performance, with respect to the amount of data received
  - TCP includes congestion control, to alleviate this situation
- OS code may be poorly implemented
  - TCP has been widely implemented by different operating systems for many years
  - Methods and implementations are well-tested
  - OS is more rigorously tested than developed applications

Issues

- Startup latency is significant
  - Noticeable latency for startup is still low
  - Reasonable price for reliability at the transport layer and congestion control

Disadvantages of UDP

- No guarantees at the transport layer
- No flow control
- Routers are more prone to drop UDP packets, no retransmissions for dropped packets
- A datagram must contain all information that the receiver needs to interpret it

UDP - No guarantees

- No built-in reliability
  - Packets can be dropped and not resent, sent out-of-order, or duplicated
  - Any reliability controls need to be implemented at the application layer
  - Leads to longer development time, longer testing time, chance of improper implementation

UDP – Importance of flow control

- Flow control
  - If the flow is properly managed, all applications would be able to use the network faster
  - Congested routers are a large cause of dropped packets, could be related to lack of flow control in UDP transfers

UDP – Application development

- Chance of improper implementation
- Often used additional features already specified by TCP must be added to the application – increased development time
- OS provides “library” for TCP
- Developers spend less time on application