Lecture Note 04/30/2002

TCP Basic Design

- Port – mailboxes on machines
  - www.iana.org (port assignment)
  - reserved ports < 1024
- Packet
  - To-port
  - From-port
  - Checksum
    - for TCP header + TCP data only, IP checksum is for IP header only
    - Is it even the right algorithm to detect network problems?
    - Can't detect "bit-flips"
- API : 2-way byte stream
  - Clients / servers
  - Connect / listen
  - Read / write (or vice versa)
  - From application's perspective, it should be able to write/read arbitrary size of data, but does not need to be the same size on either sides (i.e. client or server)
  - Send / receive buffers
  - Data are divided into segments to be sent to the receiver
  - Receiver's application will retreive data from the receiver's buffer
  - How to pick TCP segment size? IP MTU size – TCP header size
- Packet loss / ARQ
  - Timeouts and restransmissions
  - If checksum detects problem for the packet, host will not send an ACK
  - IP only hands to TCP the entire TCP packet after all fragments are ensembled
  - Need some unique number mechanism to distinguish retransmission packets (from sender and receiver ends)
- SCTP: transport protocol based on object model
- Sliding Window
  - To increase throughput of pipe
  - Propagation delay, transmission delay, bandwidth
  - Allow multiple packets to be send
  - Sliding window size? Depends on RTT * bandwidth of pipe.
    - Has to be smaller than both the send and receive window sizes
  - Identify each packet by a sequence number
  - Also to need to identify ACK numbers
  - Label sequence # for packet or bytes? TCP: bytes
    - Packet sequence #
      - Problem: if MTU estimation has changed in between retransmssion, do we still retransmit the same amount of data?

        - Why do we have separate ACK #?
            - Performance improvement
            - Ability to combile sending data with ACKs
            - Piggy back data along with ACK (e.g. GET request, ACK for the GET request as well as reply data)
            - Delay ACKs
                - waits for 200ms to see if there are data to reply along with ACKs; if timeouts, just send back ACKs
                - design and implementation are much implemented
            - Negative ACK: sent by receiver when it does not receive an expected packet (i.e. out-of-order packets)
- Fast retransmit
- Timeouts
    - Start with a fixed number (3 seconds)
    - Measure (exponentially weighted)
    - RTT – est = RTT-old * _ + RTT-new (1 – _)
    - Problems:
        - What if loss? Timeout, then double length of timeout
        - Congestion – increasing RTT
            - RTT – variance
            - Total timeout = RTT-est + variance * constant (e.g. 4)
- Flow control
    - If receiver is slow, sender is not allowed to send more packets than what the receiver can handle
    - Receive sends back (advertises) remaining receive window size back to sender
    - Once receiver buffer is full, sender will periodically asks receiver for the new receive window size
    - Silly window syndrome avoidance
        - Only send back new receive window size if it is more than half full
    - Nagle's algorithm:
        - The slower the connection (or longer latency), the more bytes should be packaged up (versus sending one byte at a time) while transmitting data for applications like Telnet.