

Mastering TCP/IP section 6

KUMO / cat

The transport layer

Defines how to transmit or receive network packets

Protocols mainly used: TCP / UDP



Port number

- There are multiple applications running simultaneously on a computer.
- A port number is required to determine an application to receive the packet.
- An application can open multiple ports.
- A port can accept multiple connections
- A port can separately be opened as each protocol

Example: port number

Port to connect | service program

80 | http daemon

22 | ssh daemon

* These daemons could be forked from the internet daemon

TCP and UDP

- Two protocols to transmit/receive network packets
 - TCP: Transmission Control Protocol
 - UDP: User Datagram Protocol
- *A protocol is specified in the IP header field.

TCP

Features:

- Establishes a solid connection
- performs packet sequencing, retransmission control and many other processes
- Guarantees the correctly-organized unbroken data is streamed to the receiver
- Tries to optimize the throughput while avoiding too much bandwidth occupation
- Slower than UDP

UDP

Features:

- Does not establish a connection
- Does nothing after sending packets
- Does not care about bandwidth at all
- Packets may be lost on the way to the destination
- Faster than TCP

Which protocol should we use?

TCP:

- Where reliable data transmissions are required
E.G. retrieving website contents, file transferring

UDP:

- Where the transmission speed has more priority than reliability
- Where the amount of packet to transmit is small
- where slight packet losses do not affect the overall performance of the task
E.G. Video streaming, realtime voice chat

Socket API

A set of Application Programming Interface(API) for handling TCP / UDP



Connecting multiple nodes to the same port

- Youtube can be streamed by many users simultaneously
- Every SFC student can tweet like crazy
- A server does not explode with a little flood of F5 attack

Why?

Five elements to distinguish incoming transmissions

- Destination IP address
- Sender IP address
- Destination port number
- Sender port number
- Protocol number

- Compare each element
- Associate to a connection that the five info perfectly matched
- Destination IP address
- Sender IP address
- Destination port number
- Sender port number
- Protocol number

Allocation of the port number

Static:

- Use 0 to 1023 (well-known ports) for well-known services
- Use 1024-49151 for your original applications
- Implemented for the server side in most cases

Dynamic:

- Use a port number that the operating system returned as a usable port (in range of 49152-65535)
- Implemented for the client side in most cases

Basics of TCP

Various tricks that TCP is performing for us



Connections

Connection

- The most basic theory of TCP
- A kind of virtual pipe that connects two nodes (technically called virtual circuit)
- Everything put in the pipe certainly reaches the exit

ACK / NACK (1)

- A special packet that the receiver sends to the sender
- Represents the successful reception of the data
- The receiver sends NACK when there is something wrong with the data
- If the sender doesn't receive an ACK within a specified period of time or receives NACK, it resends the associated data

ACK / NACK (2)

- ACK / NACK packet itself may be delayed or lost
- The receiver may receive the same data
- Duplicated data must be automatically disposed, but how?

Sequence number

- Consequential numbers linked to the octet of the data
- With an ACK response, the receiver returns the consequential number that it wants next



Retransmission timeout

- Specifies how long the sender should wait for an ACK before retransmitting
- TCP automatically updates the value using round trip time and their variance
- If a retransmission occurs, TCP doubles the timeout
- If three consequential retransmission occur, TCP closes the connection

Establishing / closing a connection

SYN: "I want to connect to you!"

ACK: "Sure you can."



three-way handshake

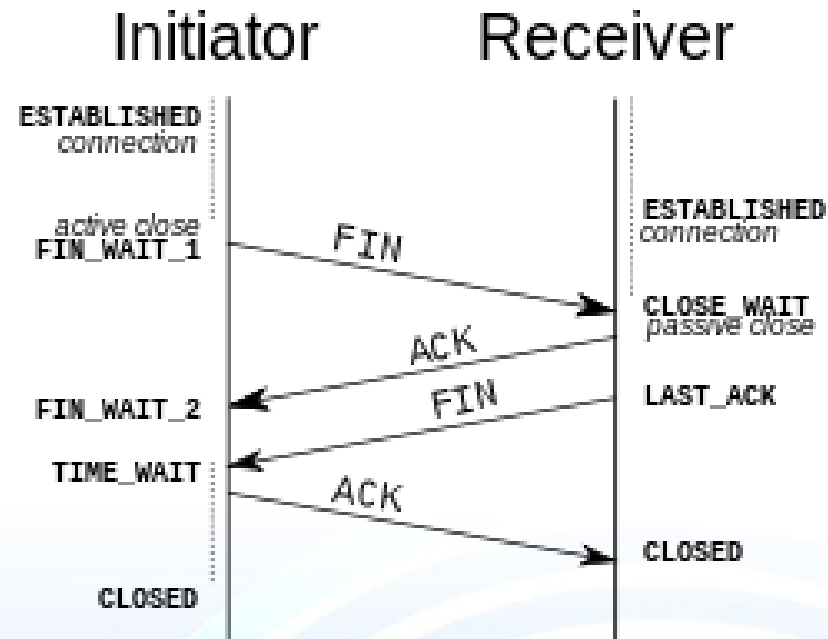


some transmissions



FIN: "That's all today, can I leave?"

ACK: "Yes, thanks!" or NACK: "No, there's some data remaining."



MSS

- Maximum segment size
- Specifies how much data to be sent per transmission
- Also used as the data retransmission unit
- Determined at the three-way handshake procedure

Window handling (1)

send some data

wait for an ACK

If an ACK arrived, send the next data, if not, resend the current data



We want more speed! Don't want to wait!

Window handling (2)

send data chunk 1

Send data chunk 2

send data chunk 3

...

ACK for data chunk 3 has arrived → forget about data chunk 1 2
3 and load 4 5 6

*If an ACK arrives, forget about the data chunks up to the
response number

send data chunk 4

send data chunk 5

send data chunk 6

...

Lessening the number of ACK's

- The receiver knows the sequence number of the data that it wants next
- The receiver does not receive anything other than that
- If the receiver gets anything it doesn't want, it requests the sequence number that it really wants



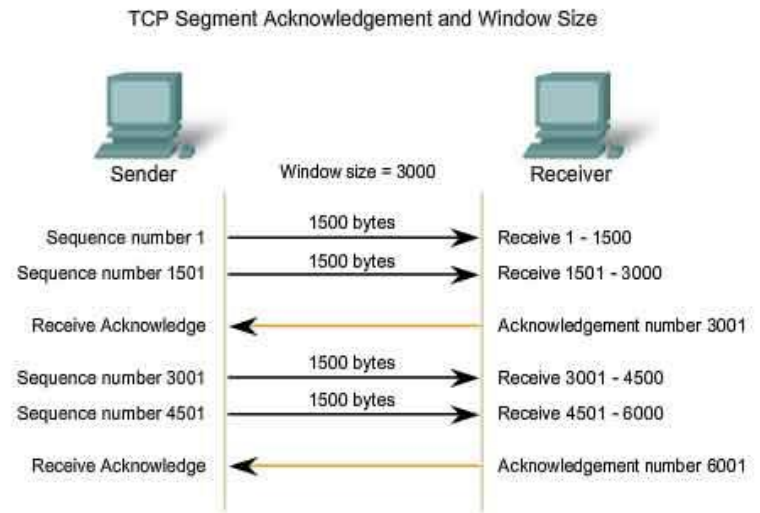
- The correct Arrival of the data prior to the sequence number is ensured



- ACK's for all segments are not necessarily required

Window size

- Window size in the previous example was 3
- Window size is specified by the receiver and dynamically changed
- The receiver periodically notifies updates of window size



The **window size** determines the number of bytes sent before an acknowledgment is expected.

The **acknowledgement** number is the number of the next expected byte.

Slow start algorithm

- An algorithm to avoid bandwidth occupation
- Starts with a slow speed
- Gradually increases the window size
- drastically decreases the window size when a retransmission is detected
- The amount of decline takes different values by the type of retransmission caused: timeout or duplication response

Piggy back

- An ACK that contains external data such as the echo-backed data of user input
- The algorithm can reduce the total amount of packet to be transferred



Other transport protocols

- UDP-Lite
- SCTP
- DCCP

