Computer Networks

Mohammed El-Hajj

Jacobs University Bremen

October 10, 2021

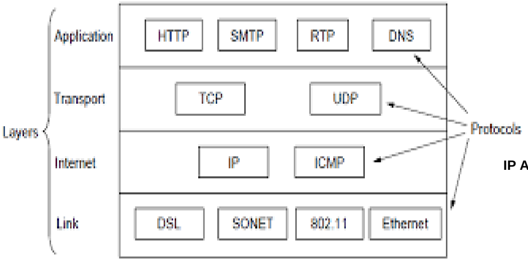
Course Content

1. Introduction
2. Fundamental Networking Concepts
3. Local Area Networks (IEEE 802)
4. Internet Network Layer (IPv4, IPv6)
5. Internet Routing (RIP, OSPF, BGP)
6. Internet Transport Layer (UDP, TCP)
7. Firewalls and Network Address Translators
8. Domain Name System (DNS)
9. Abstract Syntax Notation 1 (ASN.1)
10. External Data Representation (XDR)
11. Augmented Backus Naur Form (ABNF)
12. Electronic Mail (SMTP, IMAP)
13. Document Access and Transfer (HTTP, FTP)

Part 6: Internet Transport Layer (UDP, TCP)

Section 24: Transport Layer Overview

Internet Transport Layer



IP Address + Port Number

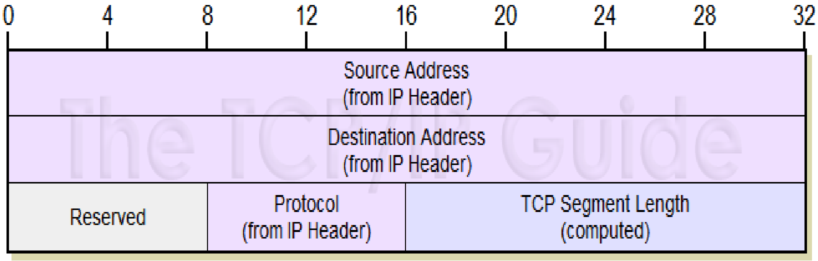
IP Address

- Network layer addresses identify interfaces on nodes (node-to-node significance).
- Transport layer addresses identify communicating application processes (end-to-end significance).
- 16-bit port numbers enable the multiplexing and demultiplexing of packets at the transport layer.

Internet Transport Layer Protocols Overview

- The *User Datagram Protocol* (UDP) provides a simple unreliable best-effort datagram service.
- The *Transmission Control Protocol* (TCP) provides a bidirectional, connection-oriented and reliable data stream.
- The *Stream Control Transmission Protocol* (SCTP) provides a reliable transport service supporting sequenced delivery of messages within multiple streams, maintaining application protocol message boundaries (application protocol framing).
- The *Datagram Congestion Control Protocol* (DCCP) provides a congestion controlled, unreliable flow of datagrams suitable for use by applications such as streaming media.

IPv4 Pseudo Header

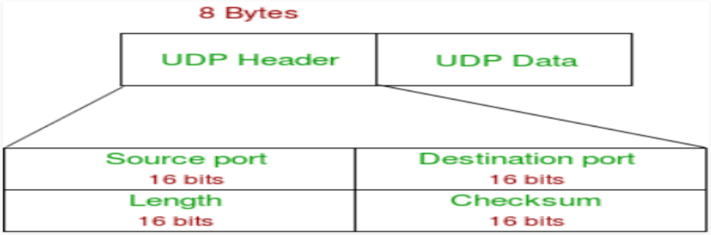


- Pseudo headers are used during checksum computation.
- A pseudo header excludes header fields that are modified by routers.

IPv6 Pseudo Header

Section 25: User Datagram Protocol (UDP)

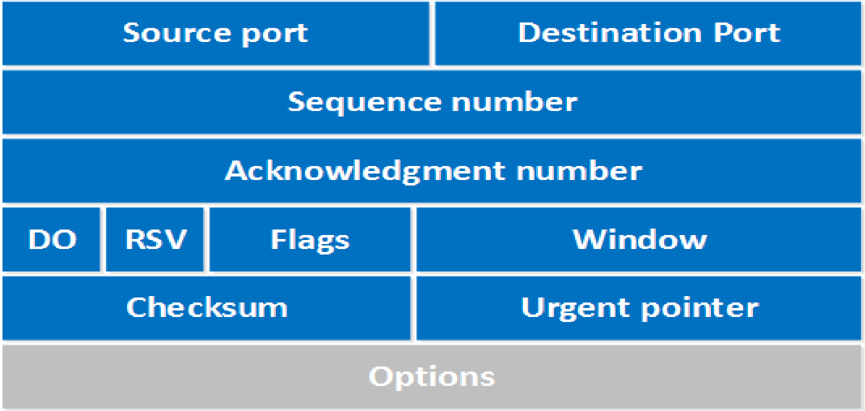
User Datagram Protocol (UDP)



- UDP (RFC 768) provides an unreliable datagram transport service.
- The UDP header simply extends the IP header with source and destination port numbers and a checksum.
- UDP adds multiplexing services to the best effort packet delivery services provided by the IP layer.
- UDP datagrams can be multicasted to a group of receivers.

Section 26: Transmission Control Protocol (TCP)

Transmission Control Protocol (TCP)

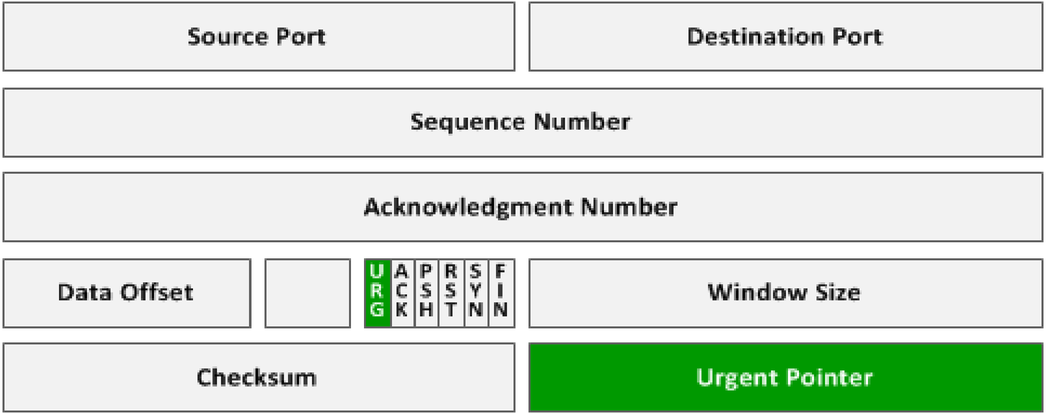


- TCP (RFC 793) provides a bidirectional connection-oriented and reliable data stream over an unreliable connection-less network protocol.

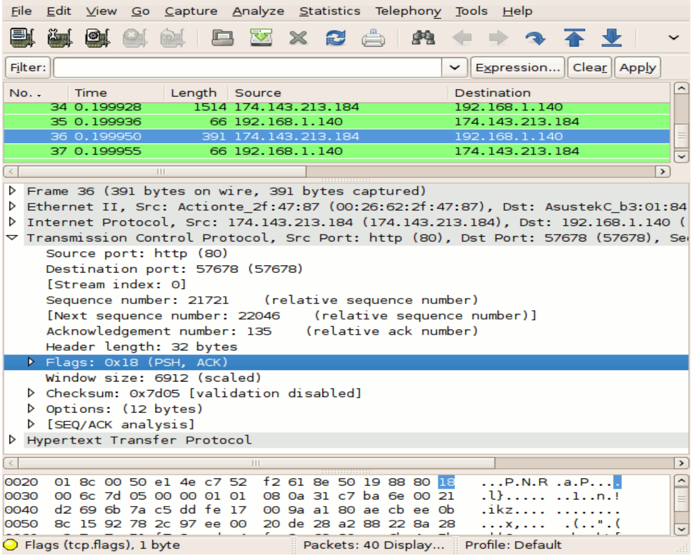
Transmission Control Protocol (TCP)

```
⊞ Frame 1: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
⊞ Ethernet II, Src: c2:01:0c:b4:00:00 (c2:01:0c:b4:00:00), Dst: c2:02:13:98:00:00 (c2:02:13:98:00:00)
⊞ Internet Protocol Version 4, Src: 192.168.12.1 (192.168.12.1), Dst: 192.168.12.2 (192.168.12.2)
⊟ Transmission Control Protocol, Src Port: 41417 (41417), Dst Port: 23 (23), Seq: 0, Len: 0
    Source Port: 41417 (41417)
    Destination Port: 23 (23)
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence number: 0    (relative sequence number)
    Acknowledgment number: 0
    Header Length: 24 bytes
  ⊟ .... 0000 0000 0010 = Flags: 0x002 (SYN)
      000. .... .... = Reserved: Not set
      ...0 .... .... = Nonce: Not set
      .... 0... .... = Congestion Window Reduced (CWR): Not set
      .... .0.. .... = ECN-Echo: Not set
      .... ..0. .... = Urgent: Not set
      .... ...0 .... = Acknowledgment: Not set
      .... .... 0... = Push: Not set
      .... .... .0.. = Reset: Not set
    ⊞ .... .... ..1. = Syn: Set
      .... .... ...0 = Fin: Not set
    Window size value: 4128
    [Calculated window size: 4128]
  ⊟ Checksum: 0xe46a [validation disabled]
      [Good Checksum: False]
      [Bad Checksum: False]
    Urgent pointer: 0
  ⊟ Options: (4 bytes), Maximum segment size
    ⊟ Maximum segment size: 1460 bytes
        Kind: Maximum Segment Size (2)
        Length: 4
        MSS Value: 1460
```

Transmission Control Protocol (TCP) - Flags



-Flags – PSH

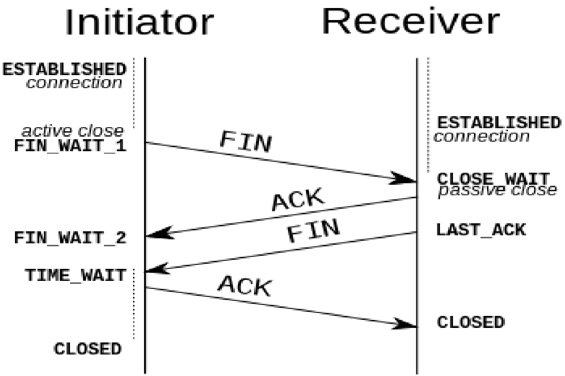

File Edit View Go Capture Analyze Statistics Telephony Tools Help
Filter:
Expression... Clear Apply
No. Time Length Source Destination
34 0.199928 1514 174.143.213.184 192.168.1.140
35 0.199936 66 192.168.1.140 174.143.213.184
36 0.199950 391 174.143.213.184 192.168.1.140
37 0.199955 66 192.168.1.140 174.143.213.184
Frame 36 (391 bytes on wire, 391 bytes captured)
Ethernet II, Src: Actionte_2f:47:87 (00:26:62:2f:47:87), Dst: AsustekC_b3:01:84
Internet Protocol, Src: 174.143.213.184 (174.143.213.184), Dst: 192.168.1.140 (
Transmission Control Protocol, Src Port: http (80), Dst Port: 57678 (57678), Se
Source port: http (80)
Destination port: 57678 (57678)
[Stream index: 0]
Sequence number: 21721 (relative sequence number)
[Next sequence number: 22046 (relative sequence number)]
Acknowledgement number: 135 (relative ack number)
Header length: 32 bytes
Flags: 0x18 (PSH, ACK)
Window size: 6912 (scaled)
Checksum: 0x7d05 [validation disabled]
Options: (12 bytes)
[SEQ/ACK analysis]
Hypertext Transfer Protocol
0020 01 8c 00 50 e1 4e c7 52 f2 61 8e 50 19 88 80 18 ...P.N.R .a.P....
0030 00 6c 7d 05 00 00 01 01 08 0a 31 c7 ba 6e 00 21 .l}..... ..1..n.!
0040 d2 69 6b 7a c5 dd fe 17 00 9a a1 80 ae cb ee 0b .ikz.... ....
0050 8c 15 92 78 2c 97 ee 00 20 de 28 a2 88 22 8a 28 ...x,... .(..".(
Flags (tcp.flags), 1 byte
Packets: 40 Display...
Profile: Default

TCP Connection Establishment

	Originator					Recipient
1.	CLOSED					LISTEN
2.	SYN-SENT	→	<SEQ=999><CTL=SYN>			SYN-RECEIVED
3.	ESTABLISHED		<SEQ=100><ACK=1000>	<CTL=SYN,ACK>	←	SYN-RECEIVED
4.	ESTABLISHED	→	<SEQ=1000><ACK=101>	<CTL=ACK>		ESTABLISHED
5.	ESTABLISHED	→	<SEQ=1000><ACK=101>	<CTL=ACK><DATA>		ESTABLISHED

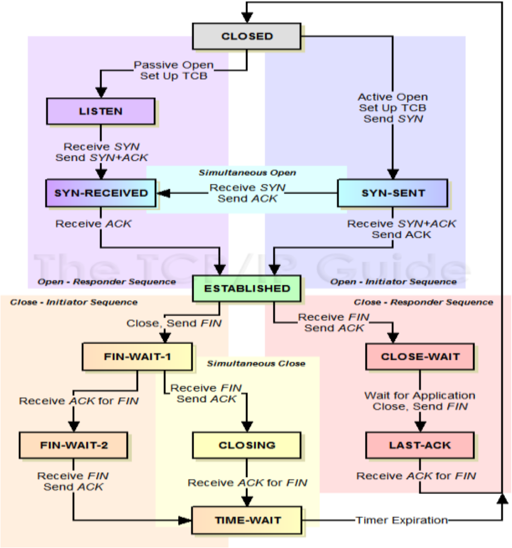
- Handshake protocol establishes TCP connection parameters and announces options.
- Guarantees correct connection establishment, even if TCP packets are lost or duplicated.

TCP Connection Tear-down

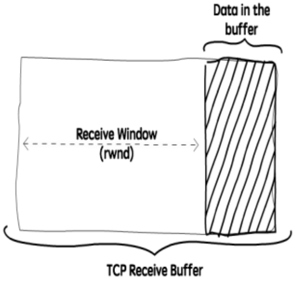


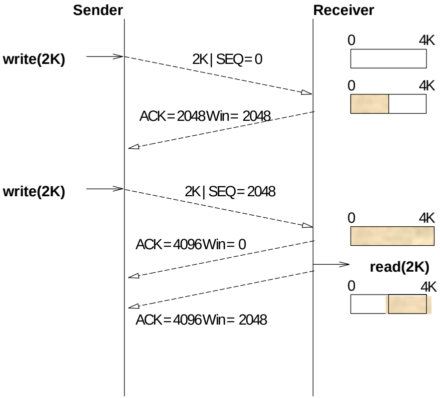
- TCP provides initially a bidirectional data stream.
- A TCP connection is terminated when both unidirectional connections have been closed. (It is possible to close only one half of a connection.)

TCP State Machine (Part #1)



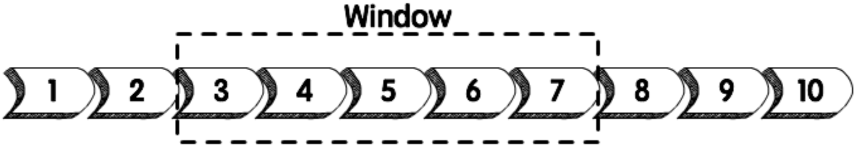
TCP Flow Control





- Both TCP engines advertise their buffer sizes during connection establishment.
- The available space left in the receiving buffer is advertised as part of the acknowledgements.

TCP Flow Control



TCP Flow Control Optimizations

- Nagle's Algorithm
  - When data comes into the sender one byte at a time, just send the first byte and buffer all the rest until the byte in flight has been acknowledgement.
  - This algorithm provides noticeable improvements especially for interactive traffic where a quickly typing user is connected over a rather slow network.
- Clark's Algorithm
  - The receiver should not send a window update until it can handle the maximum segment size it advertised when the connection was established or until its buffer is half empty.
  - Prevents the receiver from sending a very small window updates (such as a single byte).

TCP Congestion Control

- TCP's congestion control introduces the concept of a congestion window (*cwnd*) which defines how much data can be in transit.
- The congestion window is maintained by a TCP sender in addition to the flow control receiver window (*rwnd*), which is advertised by the receiver.
- The sender uses these two windows to limit the data that is sent to the network and not yet received (*flightsize*) to the minimum of the receiver and the congestion window:

$$flightsize \leq min(cwin, rwin)$$

- The key problem to be solved is the dynamic estimation of the congestion window.

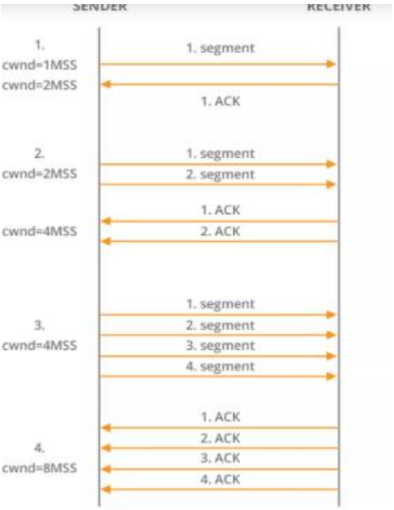
TCP Congestion Control (cont.)

- The initial window (IW) is usually initialized using the following formula:

$$IW = min(4 \cdot SMSS, max(2 \cdot SMSS, 4380 bytes))$$

SMSS is the sender maximum segment size, the size of the largest sement that the sender can transmit (excluding TCP/IP headers and options).

- During slow start, the congestion window *cwnd* increases by at most *SMSS* bytes for every received acknowledgement that acknowledges data. Slow start ends when *cwnd* exceeds *ssthresh* or when congestion is observed.
- Note that this algorithm leads to an exponential increase if there are multiple segments acknowledged in the *cwnd*.



TCP Congestion Control (cont.)

- During congestion avoidance, *cwnd* is incremented by one full-sized segment per round-trip time (RTT). Congestion avoidance continues until congestion is detected. One formula commonly used to update *cwnd* during congestion avoidance is given by the following equation:

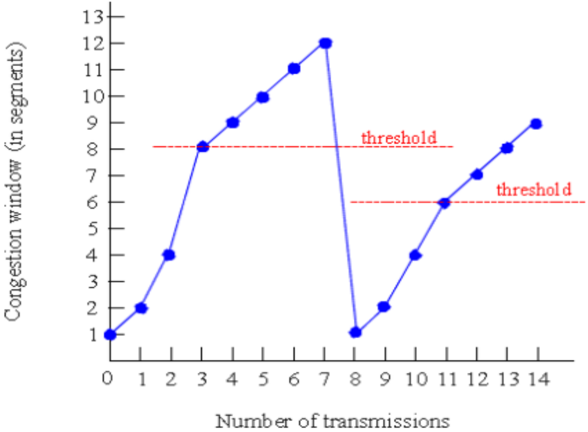
$$cwnd = cwnd + (SMSS * SMSS / cwnd)$$

  This adjustment is executed on every incoming non-duplicate ACK.

- When congestion is noticed (the retransmission timer expires), then *cwnd* v is reset to one full-sized segment and the slow start threshold *ssthresh* is updated as follows:

$$ssthresh = max(flightsize/2, 2 \cdot SMSS)$$

TCP Congestion Control (cont.)



- Congestion control with an initial window size of 2K.

Retransmission Timer

- The retransmission timer controls when a segment is resend if no acknowledgement has been received.
- The retransmission timer *RTT* needs to adapt to round-trip time changes.
- General idea:
  - Measure the current round-trip time
  - Measure the variation of the round-trip time
  - Use the estimated round-trip time plus the measured variation to calculate the retransmit timeout
  - Do not update the estimators if a segment needs to be retransmitted (Karn's algorithm).

Retransmission Timer

- If an acknowledgement is received for a segment before the associated retransmission timer expires:

$$RTT = \alpha \cdot RTT + (1 - \alpha)M$$

M is the measured round-trip time; α is typical 7/8 .

- The standard deviation is estimated using:

$$D = \alpha \cdot D + (1 - \alpha)|RTT - M|$$

α is a smoothing fact

- The retransmission timeout *RTO* is determined as follows:

$$RTO = RTT + 4 \cdot D$$

The factor 4 has been choosen empirically.

Fast Retransmit / Fast Recovery

- TCP receivers should send an immediate duplicate acknowledgement when an out-of-order segment arrives.
- The arrival of four identical acknowledgements without the arrival of any other intervening packets is an indication that a segment has been lost.
- The sender performs a fast retransmission of what appears to be the missing segment, without waiting for the retransmission timer to expire.
- Upon a fast retransmission, the sender does not exercise the normal congestion reaction with a full slow start since acknowledgements are still flowing.
- See RFC 2581 section 3.1 for details.

Karn’s Algorithm

- The dynamic estimation of the *RTT* has a problem if a timeout occurs and the segment is retransmitted.
- A subsequent acknowledgement might acknowledge the receipt of the first packet which contained that segment or any of the retransmissions.
- Karn suggested that the *RTT* estimation is not updated for any segments which were retransmitted and that the *RTO* is doubled on each failure until the segment gets through.
- The doubling of the *RTO* leads to an exponential back-off for each consecutive attempt.

Selected TCP Options

- Maximum Segment Size (MSS):
  - Communicates the maximum receive segment size of the sender of this option during connection establishment
- Window Scale (WS):
  - The number carried in the 16-bit Window field of a TCP header is scaled (shifted) by a certain constant to enable windows larger than 2^{16} octets
- TimeStamps (TS):
  - Timestamps exchanged in every TCP header to deal with the 32-bit sequence number space limitation (and to enhance round-trip time measurements)
- Selective Acknowledgment (SACK):
  - Indicate which blocks of the sequence number space are missing and which blocks are not (to improve cummulative acknowledgements)

Explicit Congestion Notification

- Idea: Routers signal congestion by setting some special bits in the IP header.
- The ECN bits are located in the Type-of-Service field of an IPv4 packet or the Traffic-Class field of an IPv6 packet.
- TCP sets the ECN-Echo flag in the TCP header to indicate that a TCP endpoint has received an ECN marked packet.
- TCP sets the Congestion-Window-Reduced (CWR) flag in the TCP header to acknowledge the receipt of and reaction to the ECN-Echo flag.

$\Longrightarrow$ ECN uses the ECT and CE flags in the IP header for signaling between routers and connection endpoints, and uses the ECN-Echo and CWR flags in the TCP header for TCP-endpoint to TCP-endpoint signaling.

TCP Performance

- Goal: Simple analytic model for steady state TCP behavior.
- We only consider congestion avoidance (no slow start).
- $W(t)$ denotes the congestion window size at time t.
- In steady state, $W(t)$ increases to a maximum value W where it experiences congestion. As a reaction, the sender sets the congestion window to $\frac{1}{2}W$.
- The time interval needed to go from $\frac{1}{2}W$ to W is T and we can send a window size of packets every RTT.
- Hence, the number N of packets is:

$$N = \frac{1}{2} \frac{T}{RTT} \left(\frac{W}{2} + W \right)$$

TCP Performance

- The time T between two packet losses equals $T = RTT \cdot W/2$ since the window increases linearly.
- By substituting T and equating the total number of packets transferred with the packet loss probability, we get

$$\frac{W}{4} \cdot \left(\frac{W}{2} + W\right) = \frac{1}{p} \Longleftrightarrow W = \sqrt{\frac{8}{3p}}$$

  where p is the packet loss probability (thus 1/p packets are transmitted between each p
  packet loss).
- The average sending rate $\bar{X}(p)$, that is the number of packets transmitted during each period, then becomes:

$$\bar{X}(p) = \frac{1/p}{RTT \cdot W/2} = \frac{1}{RTT}\sqrt{\frac{3}{2p}}$$