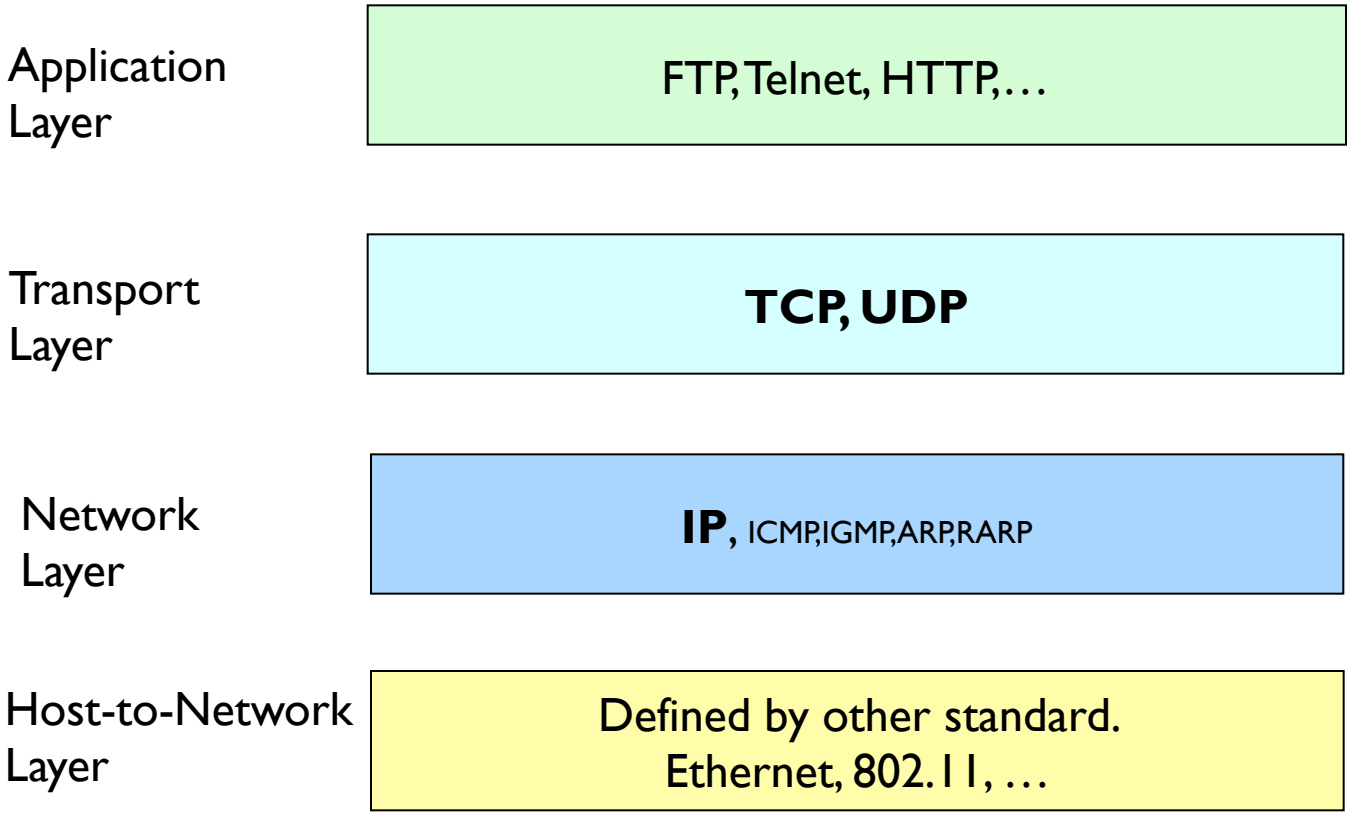
TCP: Transmission Control Protocol RFC 793,1122,1223

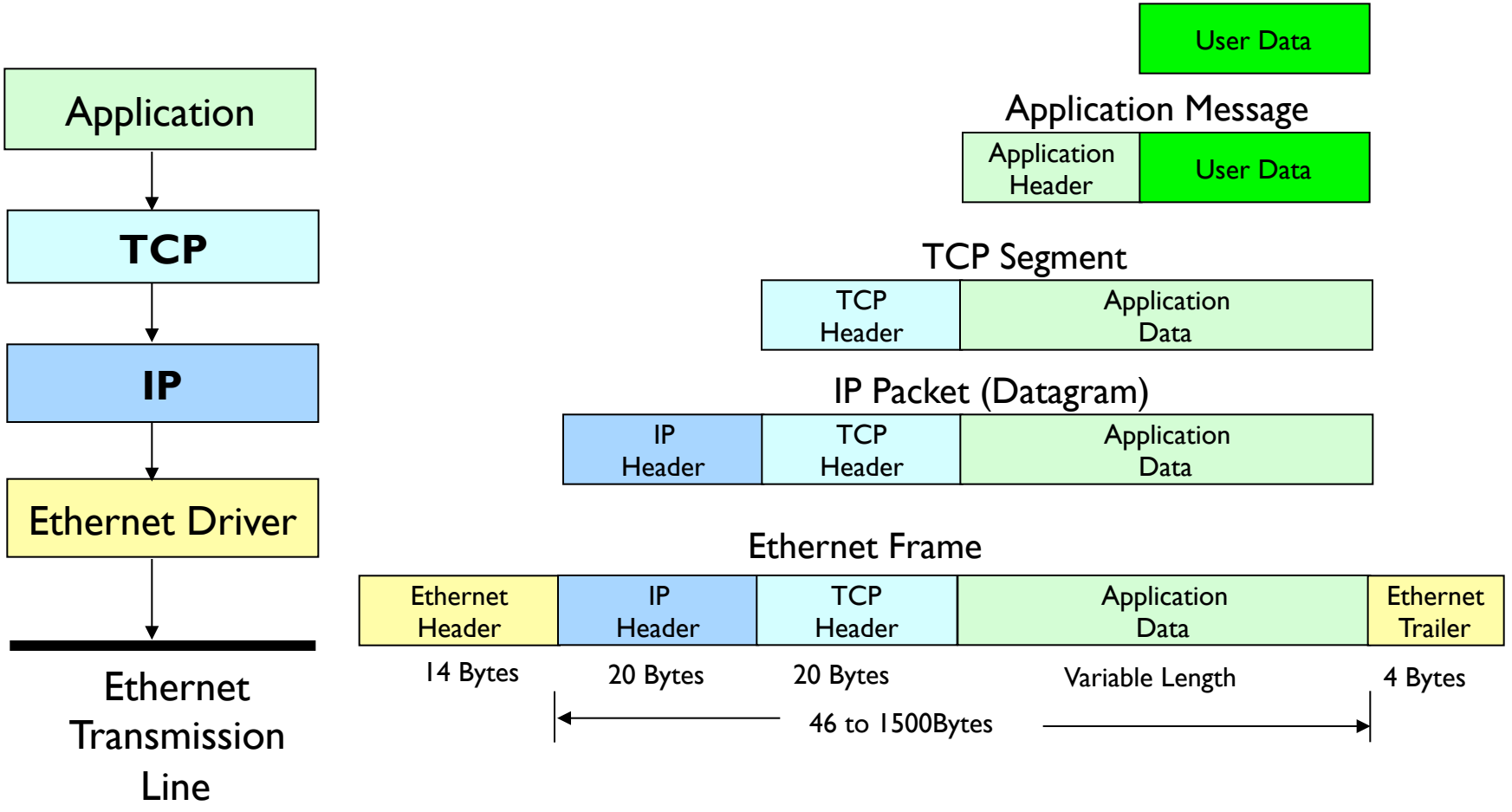
Prof. Lin Weiguo

Nov. 2013

TCP/IP Protocol Stack



Protocol Data Unit Encapsulation



TCP/IP over Ethernet

linwei@cuc.edu.cn

Purpose of TCP

- Provide a reliable end-to-end byte stream over an unreliable internetwork

Addressing

- **Socket**: TCP service is obtained by both the sender and receiver creating end points, called sockets.
- **Port** :Each socket has a socket number consisting of the IP address of the host and a 16-bit number local to that host, called a port. **A port is the TCP name for a TSAP**.
- A socket may be used for multiple connections at the same time. **Connections** are identified by the socket identifiers at both ends, that is, (**socket1, socket2**).

Well-known ports

Port	Protocol	Use
21	FTP	File transfer
23	Telnet	Remote login
25	SMTP	E-mail
69	TFTP	Trivial File Transfer Protocol
79	Finger	Lookup info about a user
80	HTTP	World Wide Web
110	POP-3	Remote e-mail access
119	NNTP	USENET news

Some assigned ports

TCP Connection

- All TCP connections are **full duplex** and **End-to-End**.
  - Full duplex means that traffic can go in both directions at the same time.
  - End-to-End means that each connection has exactly two end points. TCP does not support multicasting or broadcasting.
- A TCP connection is a **byte stream**. Not a message stream.
  - TCP software has no idea of what the bytes mean and no interest in finding out. A byte is just a byte.

The TCP Protocol

- Sequence Number
- TCP Segment
- Sliding window

Sequence Number

- Every **byte** on a TCP connection has its own **32-bit sequence number**.
- Separate 32-bit sequence numbers are used for acknowledgements and for the window mechanism

TCP segment

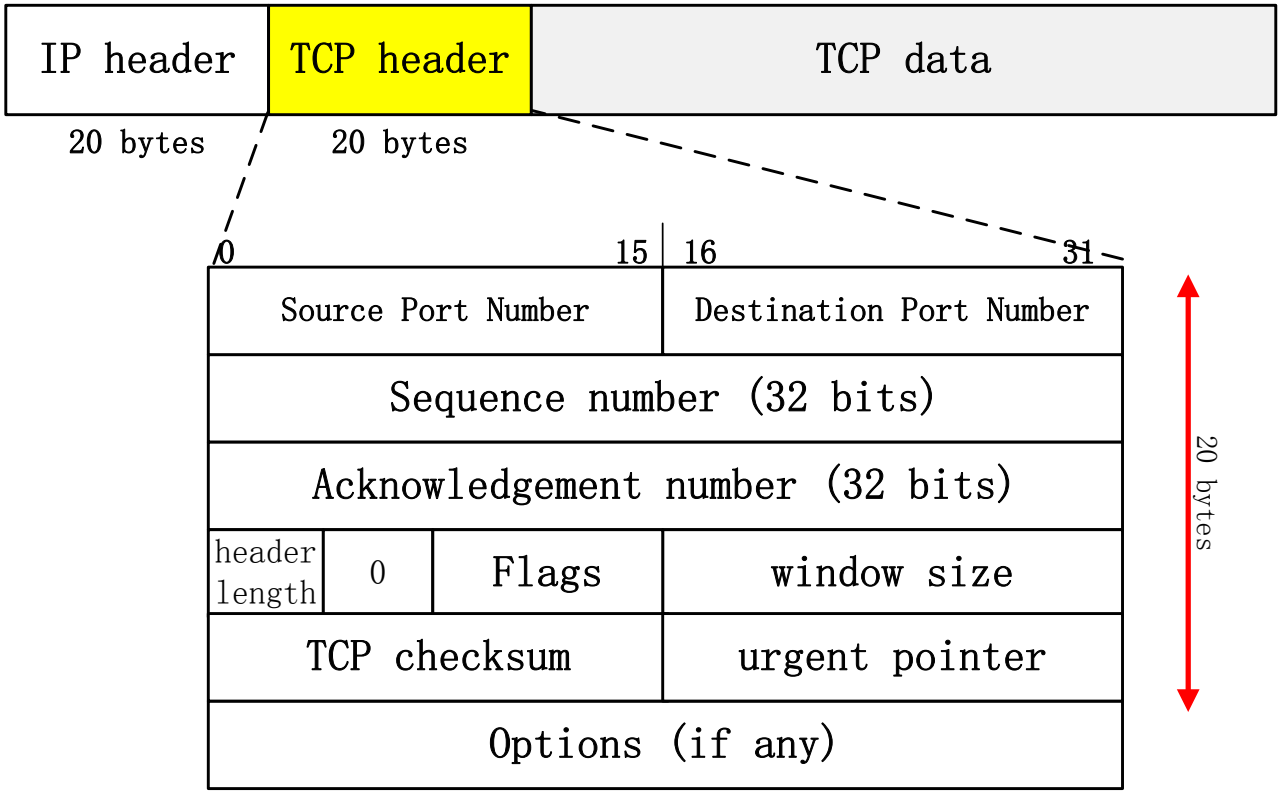
- TCP segment :consists of a fixed 20-byte header (plus an optional part) followed by zero or more data bytes.
- Two limits restrict the segment size :
  - each segment, including the TCP header, must fit in the 65,515-byte IP payload
  - each network has a maximum transfer unit, or MTU, and each segment must fit in the MTU (1500 bytes in Ethernet)

Sliding Window

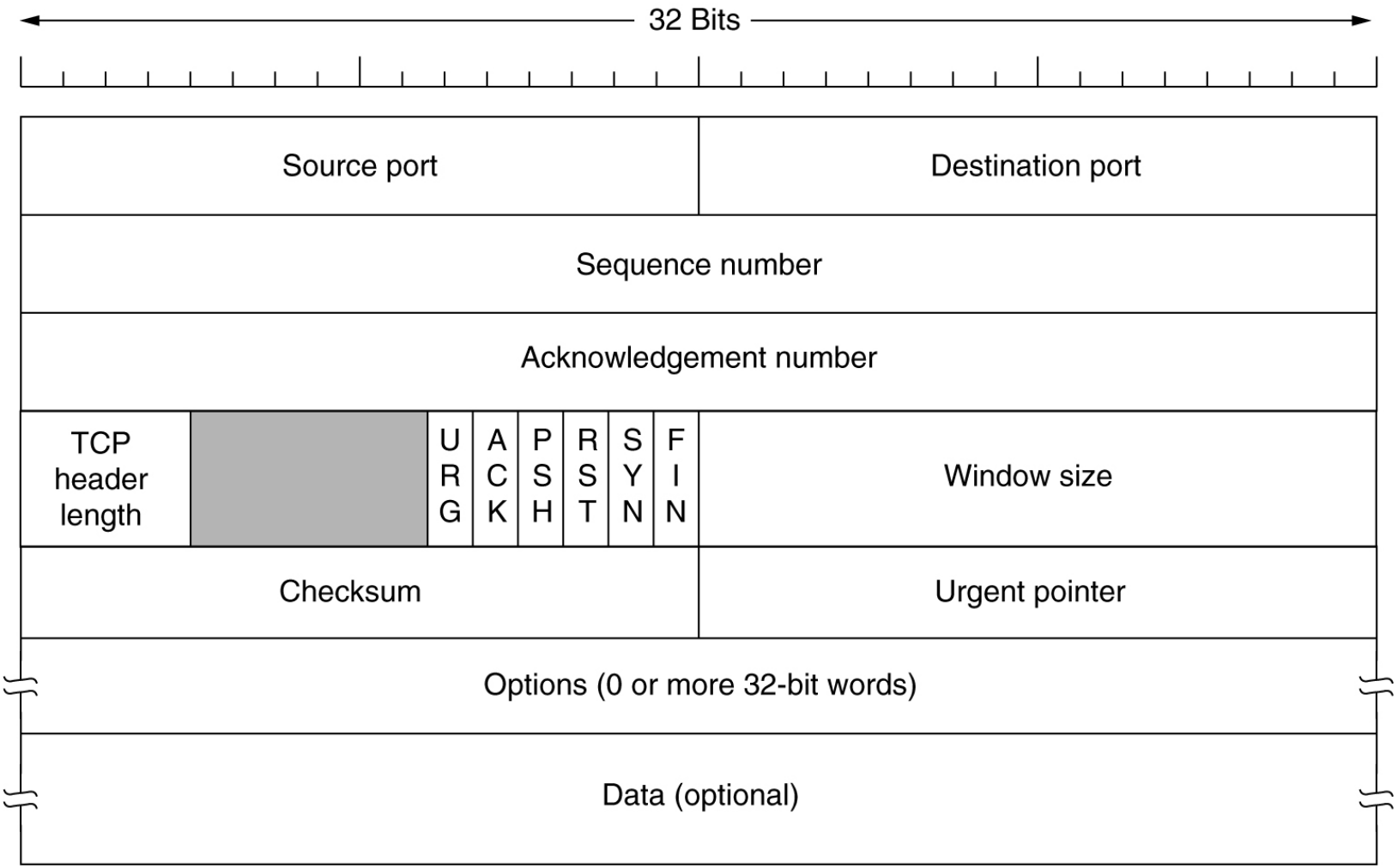
- Uses **sliding window**. The receiver sends back an **ACK** containing the sequence number of the data it **NEXT** expects to receive.
- The sender maintains a **timer** to alert to lost transmissions.

The TCP Segment Format

- TCP segments have a 20 byte header with >= 0 bytes of data.



The TCP Header Format



Header fields: Ports

- Source and destination port (16 bits each):
  - A port number identifies the endpoint of a connection.
  - A pair `<IP address, port number>` identifies one endpoint of a connection.
  - Two pairs `<client IP address, client port number>` and `<server IP address, server port number>` identify a TCP connection.

Header fields: Sequence number (32 bits)

- So the range of SeqNo is

  $0 <= SeqNo <= 2^{32} - 1 \approx 4.3$ Gbyte

- The sequence number of the first byte of data in the data portion of the segment.
- *Initial Sequence Number of a connection is set during connection establishment*

Header fields: Acknowledgement number (32 bits)

- ACK number specifies the next byte expected, not the last byte correctly received.
- Acknowledgements are piggybacked.(If a host sends an AckNo in a segment it sets the "**ACK flag**")
- TCP uses the sliding window flow protocol to regulate the flow of traffic from sender to receiver

Example: The acknowledgement for a segment with sequence numbers 0–1500 is AckNo = 1501

Header fields: TCP Header Length (4 bits)

- The number of 32-bit words in the TCP header. Used to locate the start of the data section (if any).

Header fields: Various Flags (6 bits)

- **Urgent pointer (URG)** :If set, the urgent pointer field contains a valid pointer. The urgent message in the range:

  SeqNo <= urgent message <= SeqNo+urgent pointer

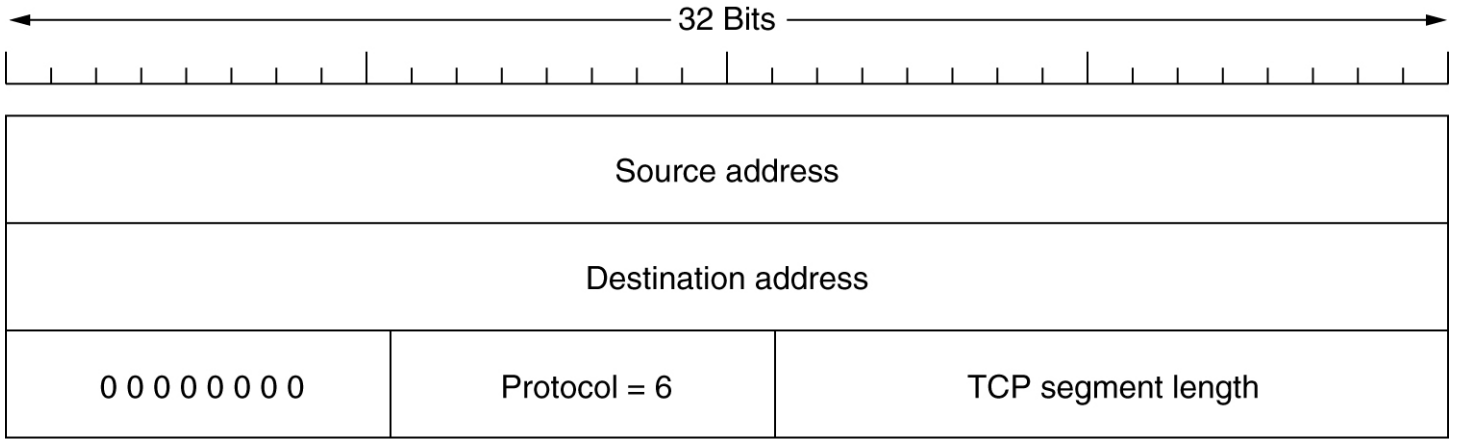
- The **ACK** bit is set to 1 to indicate that the Acknowledgement number is valid.
- The **PSH** bit indicates PUSHed data. The receiver is hereby kindly requested to deliver the data to the application upon arrival and not buffer it until a full buffer has been received
- The **RST** bit is used to reset a connection that has become confused due to a host crash or some other reason. In general, if you get a segment with the RST bit on, you have a problem on your hands.
- The **SYN** bit is used to establish connections. The connection request has SYN = 1 and ACK = 0 to indicate that the piggyback acknowledgement field is not in use.
- The **FIN** bit is used to release a connection. Both sides of a connection must send a **FIN**

Header fields: window size (16 bits)

- Flow control in TCP is handled using a variable-sized sliding window. The Window size field tells how many bytes may be sent starting at the byte acknowledged(the maximum number of bytes that a receiver can accept).
- Each side of the connection advertises the window size
- Maximum window size is 2^{16}-1= 65535 bytes

Header fields: Checksum (16 bits)

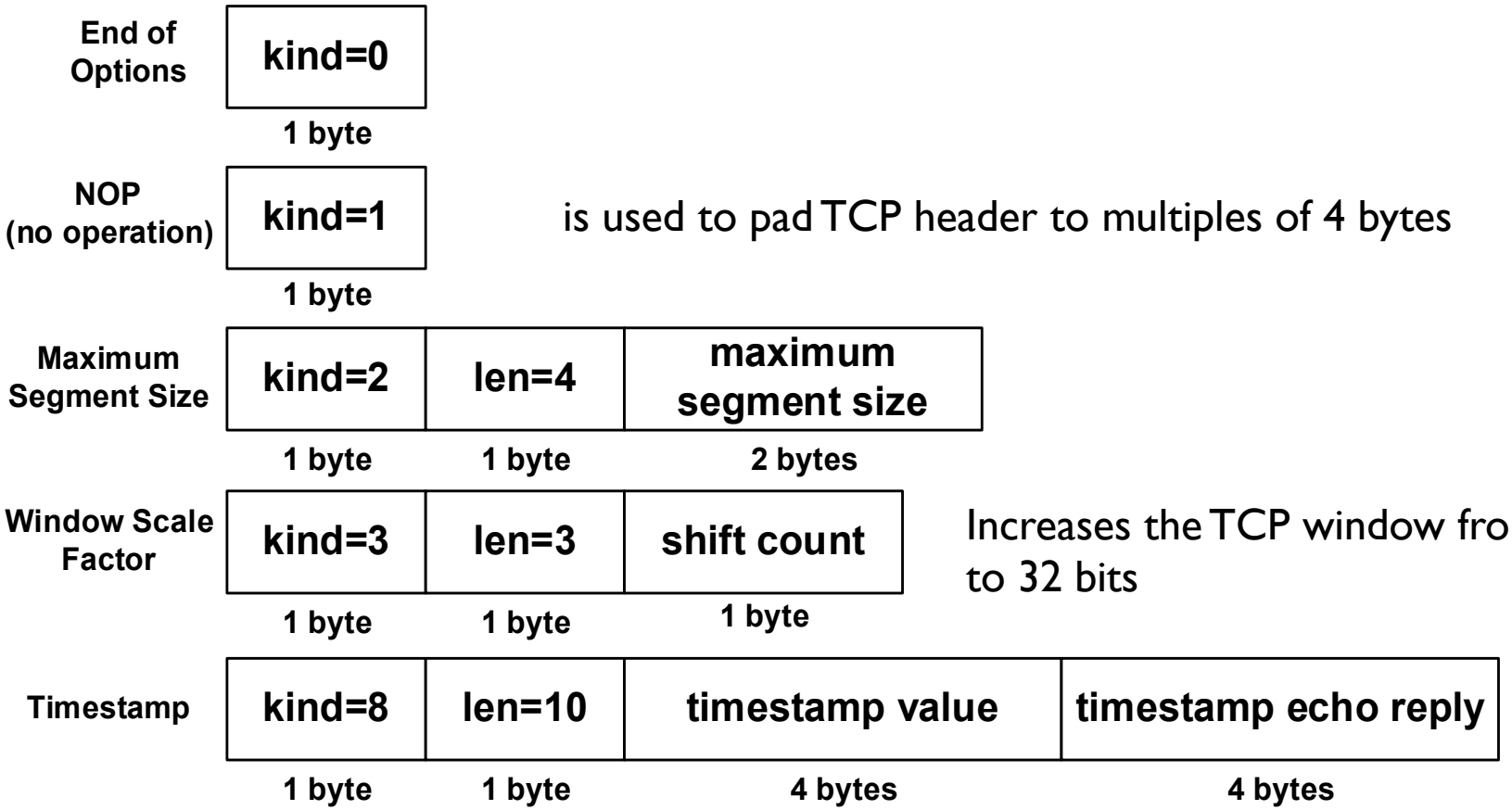
- Checksum of the pseudo-header and the TCP Segment (TCP header and data).
- The information in the pseudo header comes from the IP datagram header. This is data passed down to the IP Layer and is included in the checksum.
- To calculate the TCP segment header's Checksum field, the TCP pseudo header is first constructed and placed, logically, before the TCP segment. The checksum is then calculated over both the pseudo header and the TCP segment. The pseudo header is then discarded.



Header fields: Options (variable length)

- The Options field provides a way to add extra facilities not covered by the regular header.
- The most important option is the one that allows each host to specify the maximum TCP payload it is willing to accept.
  - During connection setup, each side can announce its maximum and see its partner's. If a host does not use this option, it defaults to a 536-byte payload. All Internet hosts are required to accept TCP segments of 536 + 20 = 556 bytes. The maximum segment size in the two directions need not be the same.

Some Options



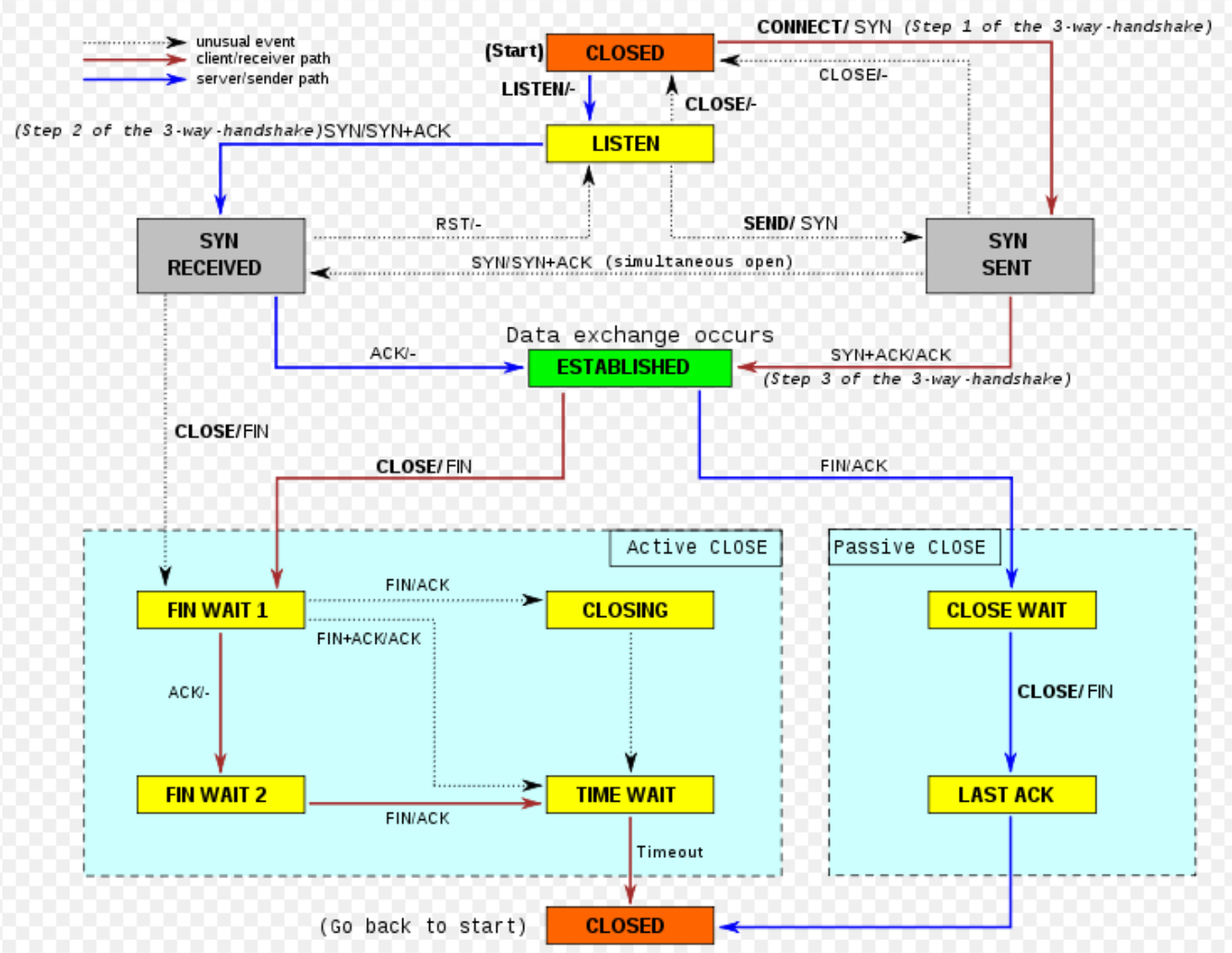
TCP extensions

- RFC 1323: big window size
  - For lines with high bandwidth, high delay, or both, the 64-KB window is often a problem. On a T3 line (44.736 Mbps), it takes only 12 ms to output a full 64-KB window. If the round-trip propagation delay is 50 ms , the sender will be idle 3/4 of the time waiting for acknowledgements. In RFC 1323, a **Window scale option** was proposed, allowing the sender and receiver to negotiate a window scale factor. Allowing windows of up to 2^{30} bytes
- RFC 2018: Selective acknowledgments
  - TCP employs the selective acknowledgment (SACK) option, defined in RFC 2018, which allows the receiver to acknowledge discontiguous blocks of packets that were received correctly, in addition to the sequence number of the last contiguous byte received successively, as in the basic TCP acknowledgment.

Protocol operation

- TCP protocol operations may be divided into three phases.
  - Connections must be properly established in a multi-step handshake process (connection establishment) before entering
  - the data transfer phase. After data transmission is completed,
  - the connection termination closes established virtual circuits and releases all allocated resources.

A Simplified TCP State Diagram



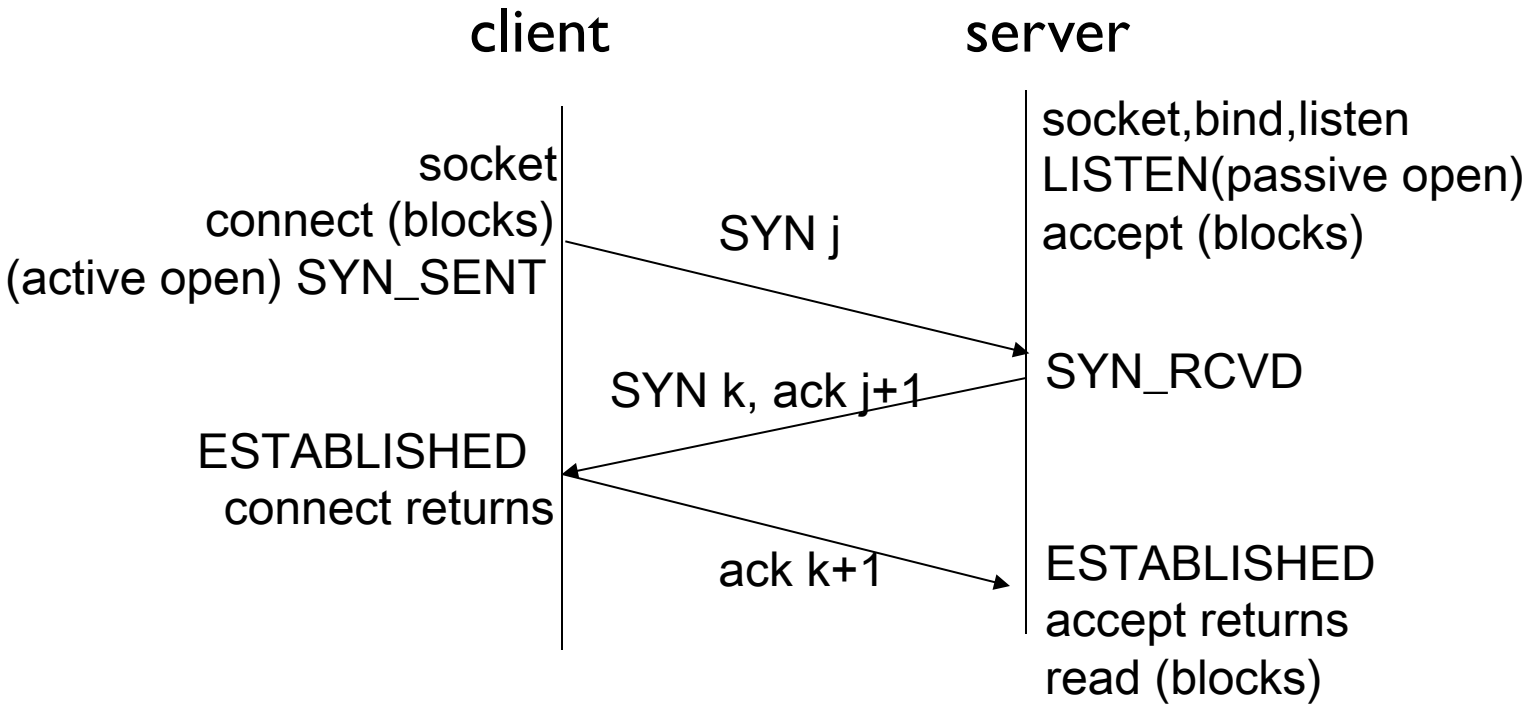
States

- LISTEN :
  - In case of a server, waiting for a connection request from any remote client.
- SYN-SENT :
  - waiting for the remote peer to send back a TCP segment with the SYN and ACK flags set. (usually set by TCP clients)
- SYN-RECEIVED :
  - waiting for the remote peer to send back an acknowledgment after having sent back a connection acknowledgment to the remote peer. (usually set by TCP servers)
- ESTABLISHED :
  - the port is ready to receive/send data from/to the remote peer.
- FIN-WAIT-1
- FIN-WAIT-2
- CLOSE-WAIT
- CLOSING
- LAST-ACK
- TIME-WAIT :
  - represents waiting for enough time to pass to be sure the remote peer received the acknowledgment of its connection termination request. According to RFC 793 a connection can stay in TIME-WAIT for a maximum of four minutes.
- CLOSED

Connection establishment

- To establish a connection, TCP uses a three-way handshake. Before a client attempts to connect with a server, the server must first bind to a port to open it up for connections: this is called a passive open. Once the passive open is established, a client may initiate an active open. To establish a connection, the three-way (or 3-step) handshake occurs:
  1. The active open is performed by the client sending a SYN to the server. It sets the segment's sequence number to a random value.
  2. In response, the server replies with a SYN-ACK. The acknowledgment number is set to one more than the received sequence number, and the sequence number is random.
  3. Finally, the client sends an ACK back to the server. The sequence number is set to the received acknowledgement value, and the acknowledgement number is set to one more than the received sequence number.
- At this point, both the client and server have received an acknowledgment of the connection.

Three-way handshake



TCP options (in SYN): MSS (maximum segment size) option, window scale option (advertized window up to 65535x2^14, 1GB), timestamp option (the latter two: long fat pipe options)

Normal Case Example

1. TCP A picks an initial sequence number (A_SEQ) and sends a segment to B containing:

   SYN_FLAG = 1, ACK_FLAG = 0, and SEQ = A_SEQ. *

2. When TCP B receives the SYN, it chooses its initial sequence number (B_SEQ) and sends a TCP segment to A containing:

   SYN_FLAG = 1, ACK_FLAG = 1, ACK = (A_SEQ + 1), SEQ = B_SEQ. *

3. When A receives B 's response, it acknowledges B 's choice of an initial sequence number by sending a dataless third segment containing:

   SYN_FLAG = 0, ACK_FLAG = 1, ACK = (B_SEQ + 1), SEQ = A_SEQ+1 (data length was 0).

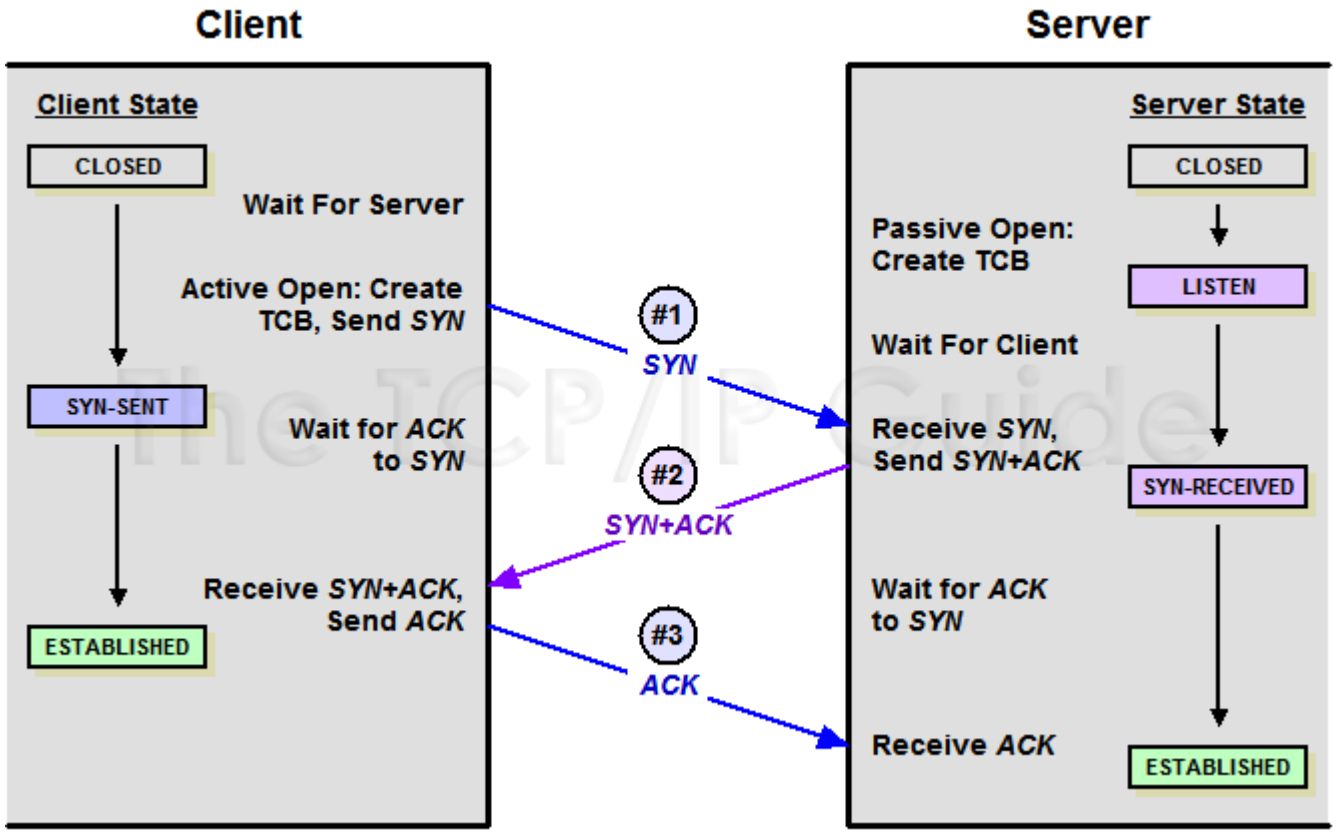
4. Data transfer may now begin.

(* counts as one byte), see next

SN in SYN Segments

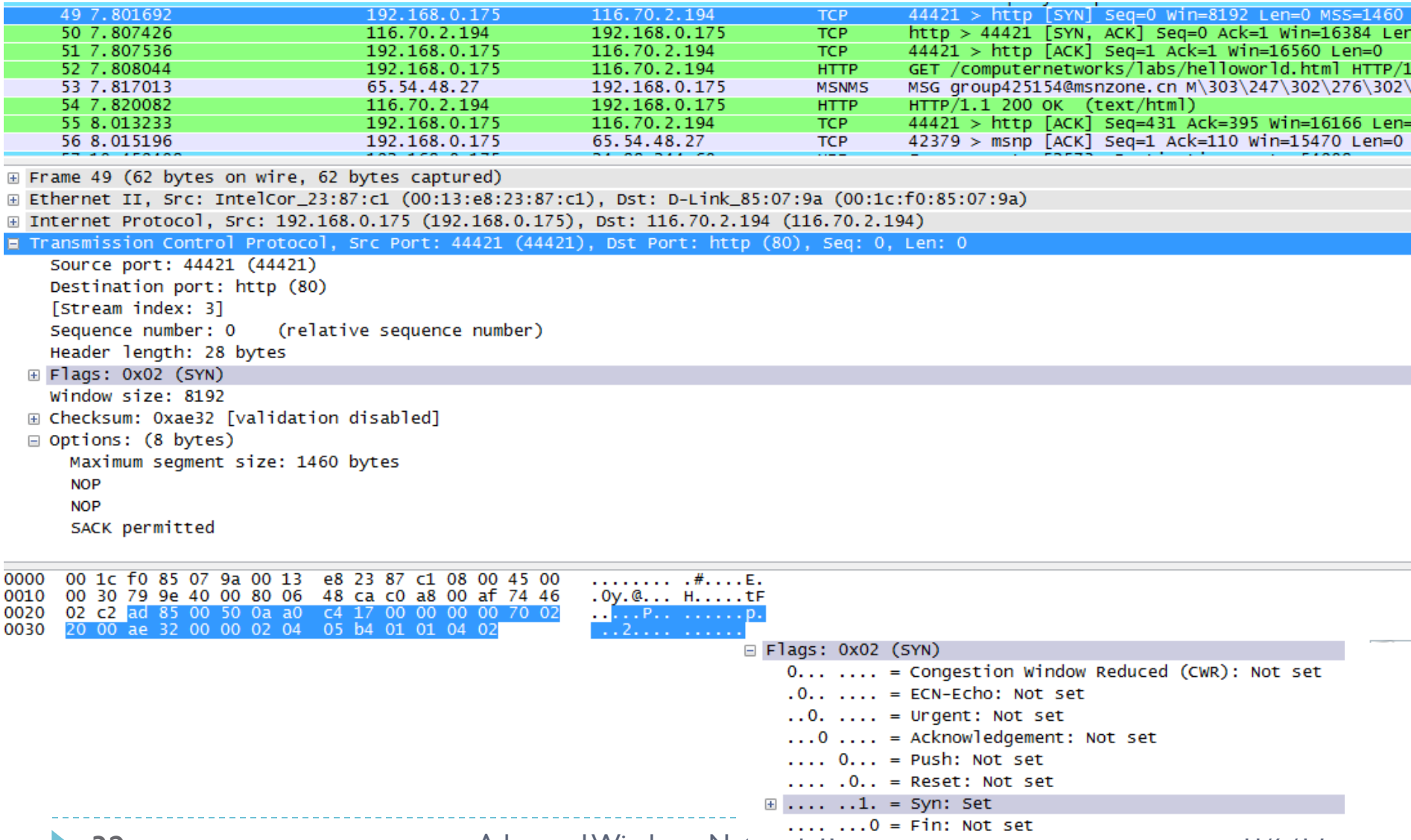
- Note: The sequence number used in SYN segments are actually part of the sequence number space.
  - That is why the third segment that A sends contains SEQ=(A_SEQ+1).
  - This is required so that we don't get confused by old SYNs that we have already seen.
  - To ensure that old segments are ignored, TCP ignores any segments that refer to a sequence number outside of its receive window.
  - This includes segments with the SYN bit set.

A real example

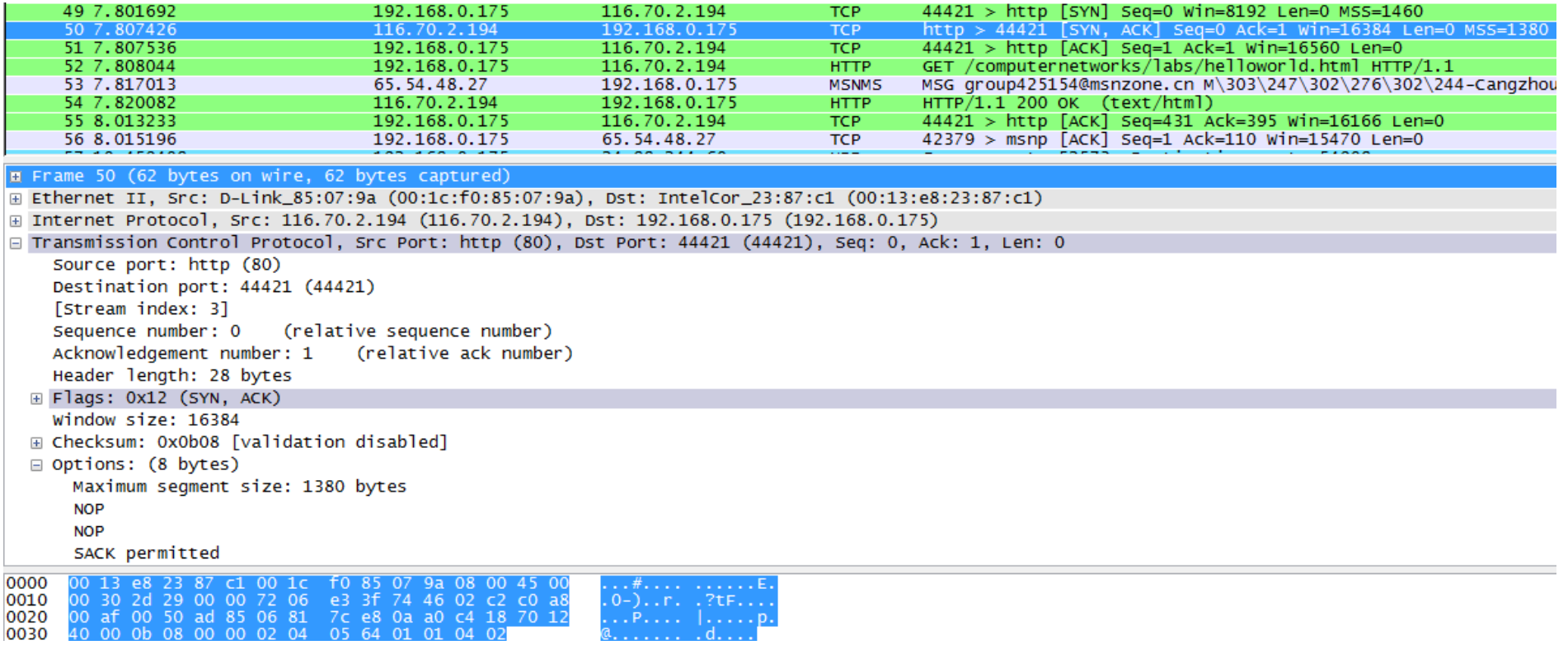



```
49 7.801692   192.168.0.175   116.70.2.194    TCP   44421 > http [SYN] Seq=0 Win=8192 Len=0 MSS=1460
50 7.807426   116.70.2.194    192.168.0.175   TCP   http > 44421 [SYN, ACK] Seq=0 Ack=1 Win=16384 Len=0 MSS=1380
51 7.807536   192.168.0.175   116.70.2.194    TCP   44421 > http [ACK] Seq=1 Ack=1 Win=16560 Len=0
```

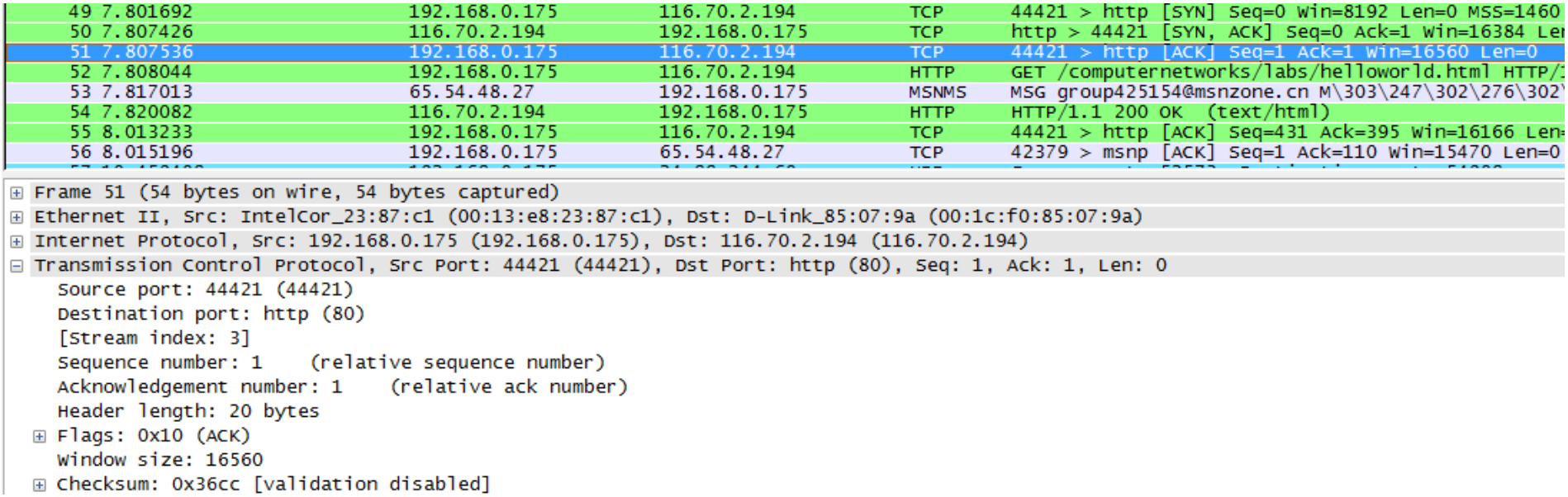
SYN_SENT


49 7.801692 192.168.0.175 116.70.2.194 TCP 44421 > http [SYN] Seq=0 Win=8192 Len=0 MSS=1460
50 7.807426 116.70.2.194 192.168.0.175 TCP http > 44421 [SYN, ACK] Seq=0 Ack=1 Win=16384 Len
51 7.807536 192.168.0.175 116.70.2.194 TCP 44421 > http [ACK] Seq=1 Ack=1 Win=16560 Len=0
52 7.808044 192.168.0.175 116.70.2.194 HTTP GET /computernetworks/labs/helloworld.html HTTP/1
53 7.817013 65.54.48.27 192.168.0.175 MSNMS MSG group425154@msnzone.cn M\303\247\302\276\302\
54 7.820082 116.70.2.194 192.168.0.175 HTTP HTTP/1.1 200 OK (text/html)
55 8.013233 192.168.0.175 116.70.2.194 TCP 44421 > http [ACK] Seq=431 Ack=395 Win=16166 Len=
56 8.015196 192.168.0.175 65.54.48.27 TCP 42379 > msnp [ACK] Seq=1 Ack=110 Win=15470 Len=0
Frame 49 (62 bytes on wire, 62 bytes captured)
Ethernet II, Src: IntelCor_23:87:c1 (00:13:e8:23:87:c1), Dst: D-Link_85:07:9a (00:1c:f0:85:07:9a)
Internet Protocol, Src: 192.168.0.175 (192.168.0.175), Dst: 116.70.2.194 (116.70.2.194)
Transmission Control Protocol, Src Port: 44421 (44421), Dst Port: http (80), Seq: 0, Len: 0
Source port: 44421 (44421)
Destination port: http (80)
[Stream index: 3]
Sequence number: 0 (relative sequence number)
Header length: 28 bytes
Flags: 0x02 (SYN)
Window size: 8192
Checksum: 0xae32 [validation disabled]
Options: (8 bytes)
Maximum segment size: 1460 bytes
NOP
NOP
SACK permitted
0000 00 1c f0 85 07 9a 00 13 e8 23 87 c1 08 00 45 00 ..... .#....E.
0010 00 30 79 9e 40 00 80 06 48 ca c0 a8 00 af 74 46 .0y.@... H.....tF
0020 02 c2 ad 85 00 50 0a a0 c4 17 00 00 00 00 70 02 .....P.. ......p.
0030 20 00 ae 32 00 00 02 04 05 b4 01 01 04 02 ..2.... ......
Flags: 0x02 (SYN)
0... .... = Congestion Window Reduced (CWR): Not set
.0.. .... = ECN-Echo: Not set
..0. .... = Urgent: Not set
...0 .... = Acknowledgement: Not set
.... 0... = Push: Not set
.... .0.. = Reset: Not set
.... ..1. = Syn: Set
....0 = Fin: Not set

SYN-RECEIVED



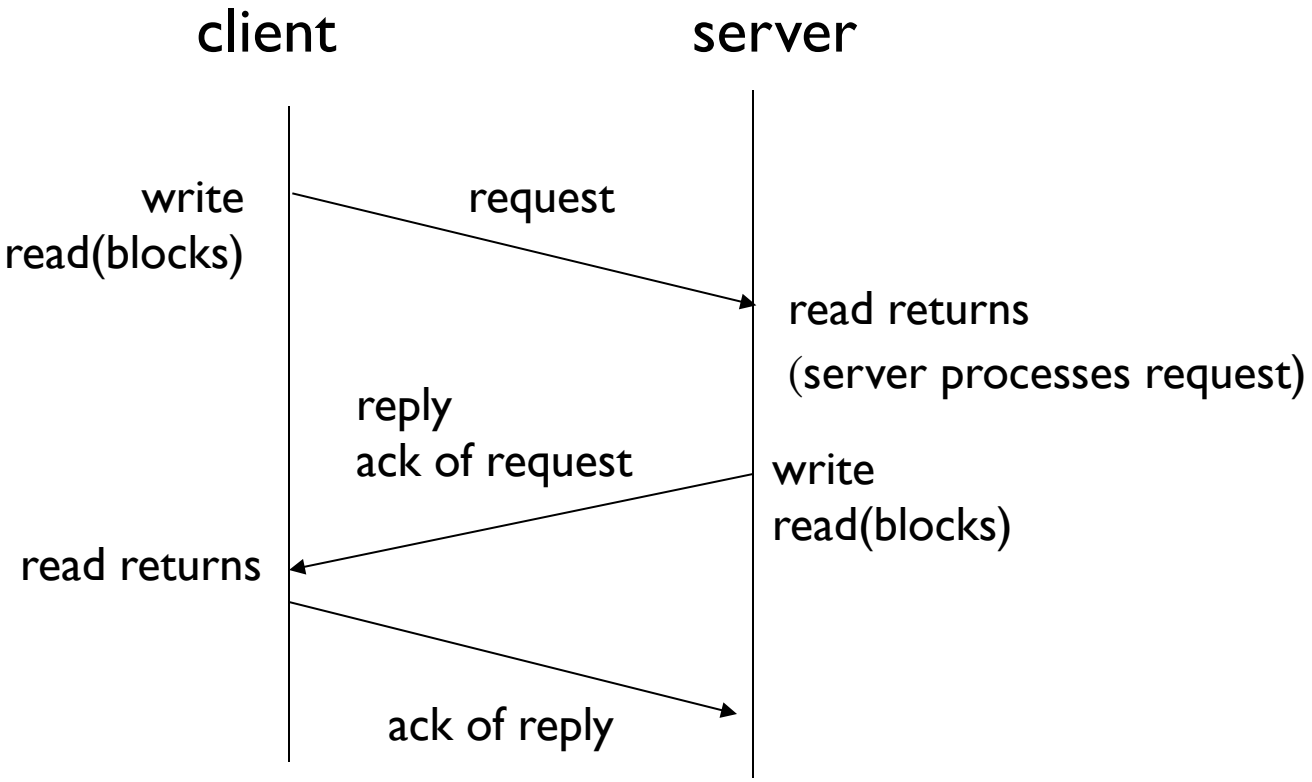
ESTABLISHED



```
    49 7.801692    192.168.0.175   116.70.2.194    TCP     44421 > http [SYN] Seq=0 Win=8192 Len=0 MSS=1460
    50 7.807426    116.70.2.194    192.168.0.175   TCP     http > 44421 [SYN, ACK] Seq=0 Ack=1 Win=16384 Le
    51 7.807536    192.168.0.175   116.70.2.194    TCP     44421 > http [ACK] Seq=1 Ack=1 Win=16560 Len=0
    52 7.808044    192.168.0.175   116.70.2.194    HTTP    GET /computernetworks/labs/helloworld.html HTTP/
    53 7.817013    65.54.48.27     192.168.0.175   MSNMS   MSG group425154@msnzone.cn M\303\247\302\276\302
    54 7.820082    116.70.2.194    192.168.0.175   HTTP    HTTP/1.1 200 OK  (text/html)
    55 8.013233    192.168.0.175   116.70.2.194    TCP     44421 > http [ACK] Seq=431 Ack=395 Win=16166 Len=
    56 8.015196    192.168.0.175   65.54.48.27     TCP     42379 > msnp [ACK] Seq=1 Ack=110 Win=15470 Len=0

⊞ Frame 51 (54 bytes on wire, 54 bytes captured)
⊞ Ethernet II, Src: IntelCor_23:87:c1 (00:13:e8:23:87:c1), Dst: D-Link_85:07:9a (00:1c:f0:85:07:9a)
⊞ Internet Protocol, Src: 192.168.0.175 (192.168.0.175), Dst: 116.70.2.194 (116.70.2.194)
⊟ Transmission Control Protocol, Src Port: 44421 (44421), Dst Port: http (80), Seq: 1, Ack: 1, Len: 0
    Source port: 44421 (44421)
    Destination port: http (80)
    [Stream index: 3]
    Sequence number: 1    (relative sequence number)
    Acknowledgement number: 1    (relative ack number)
    Header length: 20 bytes
  ⊞ Flags: 0x10 (ACK)
    Window size: 16560
  ⊞ Checksum: 0x36cc [validation disabled]
```

Data Transfer



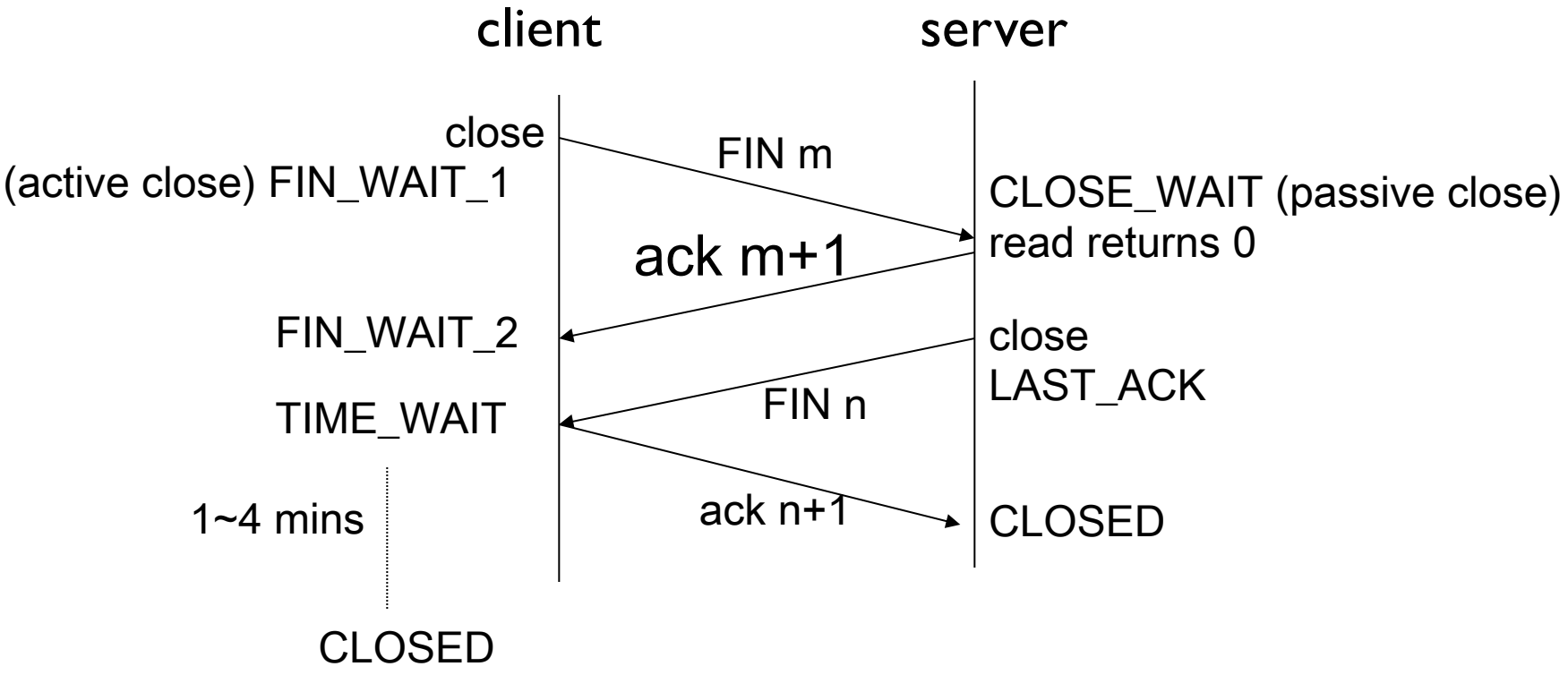
Data transfer features

- There are a few key features that set TCP apart from UDP:
  - Ordered data transfer - the destination host rearranges according to sequence number
  - Retransmission of lost packets - any cumulative stream not acknowledged will be retransmitted
  - Discarding duplicate packets
  - Error-free data transfer
  - Flow control - limits the rate a sender transfers data to guarantee reliable delivery. When the receiving host's buffer fills, the next acknowledgement contains a 0 in the window size, to stop transfer and allow the data in the buffer to be processed.
  - Congestion control - sliding window

Connection termination

- The connection termination phase uses, at most, a four-way handshake, with each side of the connection terminating independently.
  - When an endpoint wishes to stop its half of the connection, it transmits a FIN packet, which the other end acknowledges with an ACK. Therefore, a typical tear-down requires a pair of FIN and ACK segments from each TCP endpoint.
- A connection can be "half-open", in which case one side has terminated its end, but the other has not. The side that has terminated can no longer send any data into or receive any data from the connection, but the other side can (but generally if it tries, this should result in no acknowledgment and therefore a timeout, or else result in a positive RST, and either way thereby the destruction of the half-open socket).
- It is also possible to terminate the connection by a 3-way handshake, when host A sends a FIN and host B replies with a FIN & ACK (merely combines 2 steps into one) and host A replies with an ACK. This is perhaps the most common method.
- It is possible for both hosts to send FINs simultaneously then both just have to ACK. This could possibly be considered a 2-way handshake since the FIN/ACK sequence is done in parallel for both directions.

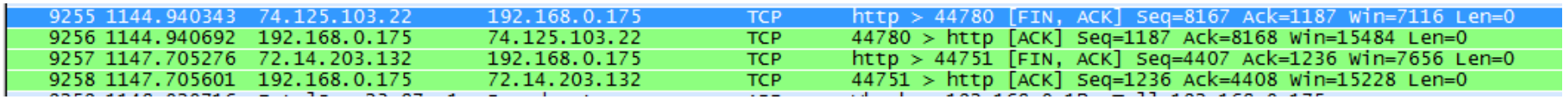
four-way handshake



TIME_WAIT to allow old duplicate segment to expire for reliable termination (the end performing active close might have to retransmit the final ACK)

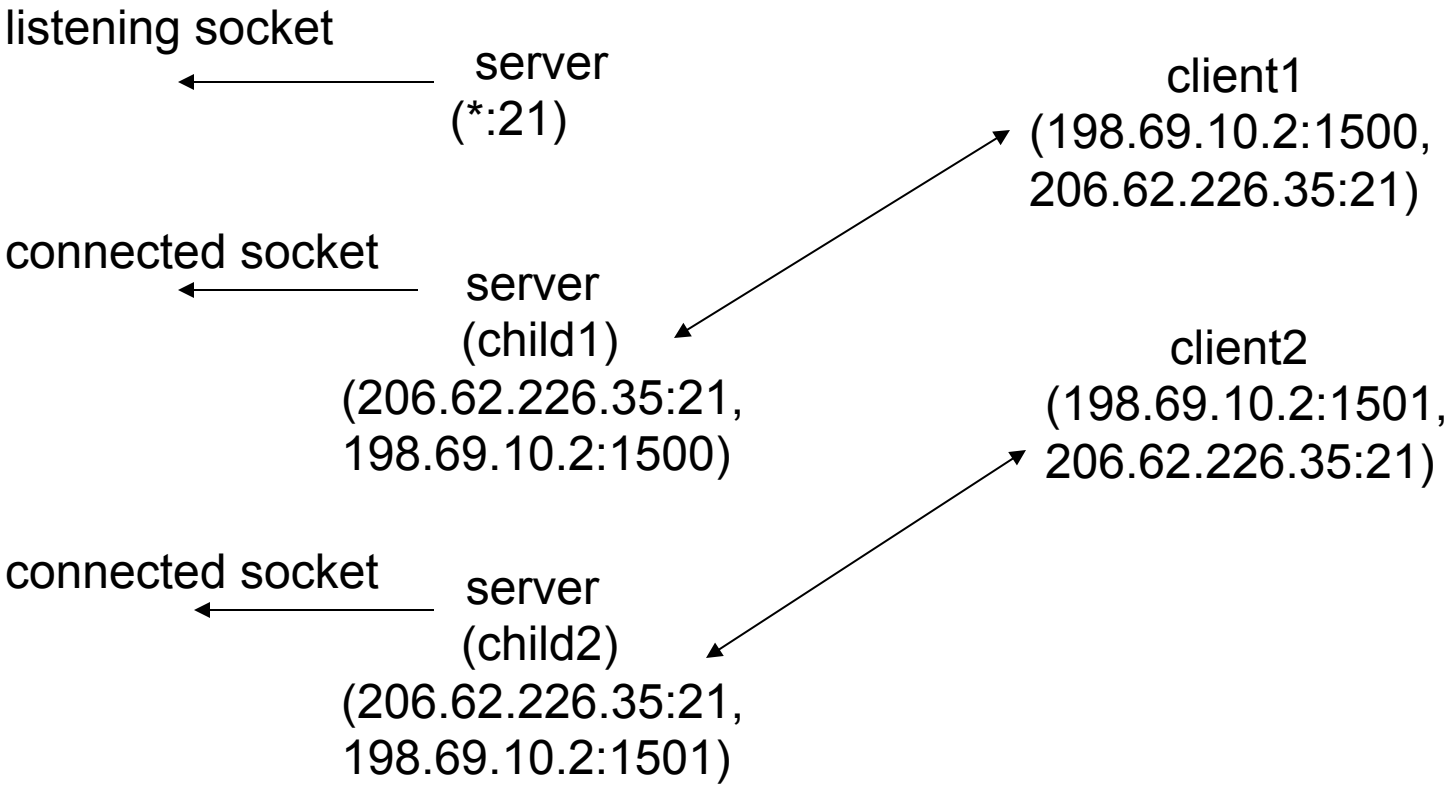
Timers to avoid the two-army problem

- If a response to a FIN is not forthcoming within two maximum packet lifetimes, the sender of the FIN releases the connection. The other side will eventually notice that nobody seems to be listening to it any more and will time out as well. While this solution is not perfect, given the fact that a perfect solution is theoretically impossible, it will have to do. In practice, problems rarely arise.



9255	1144.940343	74.125.103.22	192.168.0.175	TCP	http > 44780 [FIN, ACK] Seq=8167 Ack=1187 Win=7116 Len=0
9256	1144.940692	192.168.0.175	74.125.103.22	TCP	44780 > http [ACK] Seq=1187 Ack=8168 Win=15484 Len=0
9257	1147.705276	72.14.203.132	192.168.0.175	TCP	http > 44751 [FIN, ACK] Seq=4407 Ack=1236 Win=7656 Len=0
9258	1147.705601	192.168.0.175	72.14.203.132	TCP	44751 > http [ACK] Seq=1236 Ack=4408 Win=15228 Len=0

Multiple Sockets with the Same Port (in Concurrent Server)



All TCP segments destined for port 21, with *socket pairs* different from (206.62.226.35:21,198.69.10.2:1500) and (206.62.226.35:21,198.69.10.2:1501), are delivered to the original server with the listening socket.

Size Matters: MTU, datagram, TCP MSS, buffer

- *Link* MTU (maximum transmission unit): Ethernet MTU: 1500 bytes, PPP MTU: configurable
- *Path* MTU: the smallest link MTU in the path, can be discovered by IP DF (don't fragment) bit
- Maximum IP datagram: 65535 (IPv4), 65575 (IPv6) (IPv6 has 32-bit jumbo payload option), minimum IP reassembly buffer size
- TCP MSS (maximum segment size): actual value of reassembly buffer size, often the link MTU minus IP and TCP headers, to avoid fragmentation

TCP port scanner using socket

- Normal Three-way handshake Full connection:
  - First the host A sends the SYN packet (TCP packet with SYN flag set) to host B.
  - If the port is open then host B responds by sending SYN+ACK packet. else it sends the RST+ACK packet to host B.
  - Now host A sends the ACK packet to host B. (if SYN+ACK packet is received).
- Some of the simple port scanners use this technique.
  - It can be implemented by creating socket and calling Connect method on each port. This is simple to implement but quite slow and moreover it can be easily detected.

Half Open scanning by Winpcap

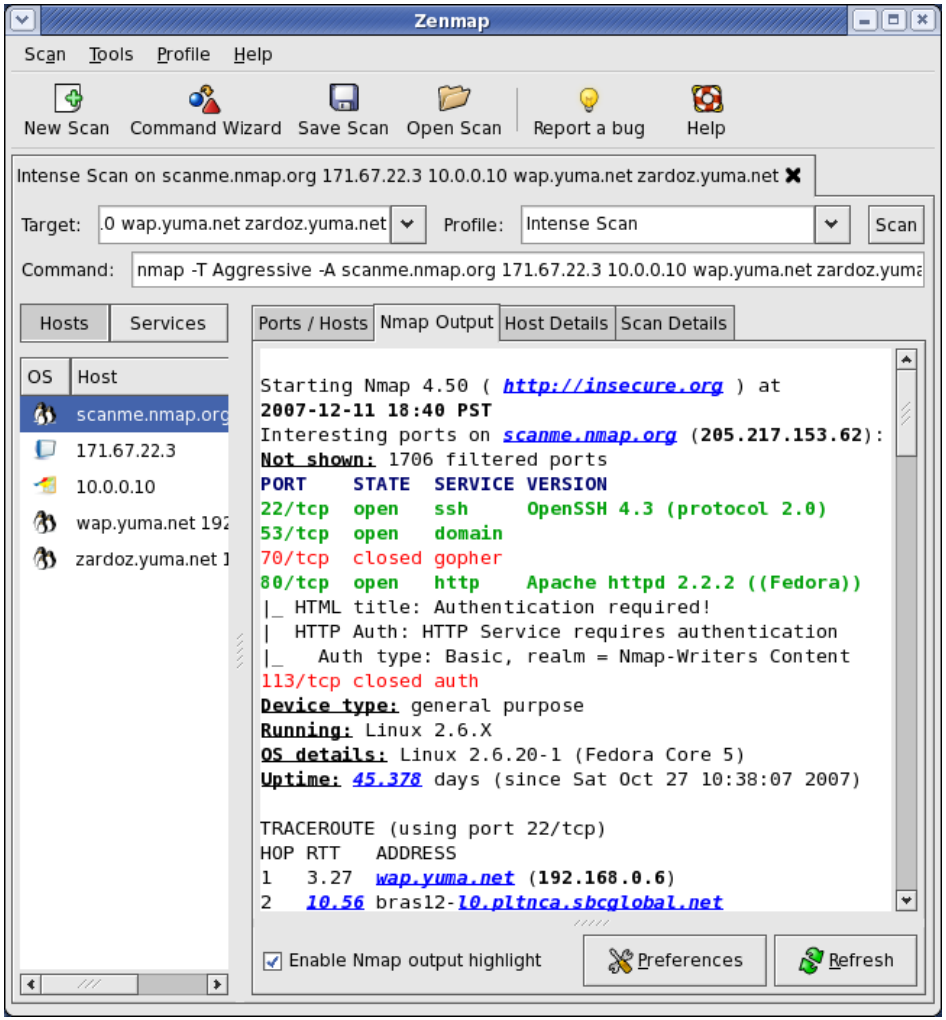
- Half scanning is more fast and efficient than full scanning technique. Half open connection is explained below.
  - First the host A sends the SYN packet (TCP packet with SYN flag set) to host B.
  - If the port is open then host B responds by sending SYN+ACK packet. else it sends the RST+ACK packet to host B.
- Since the host A does not send any additional ACK packet , it is called half open connection. Now the host can easily find out if the target port is open or closed. If it receives TCP packet with SYN+ACK flag set, then it means that target port is open. If it receives RST+ACK packet ,it implies that target port is closed.

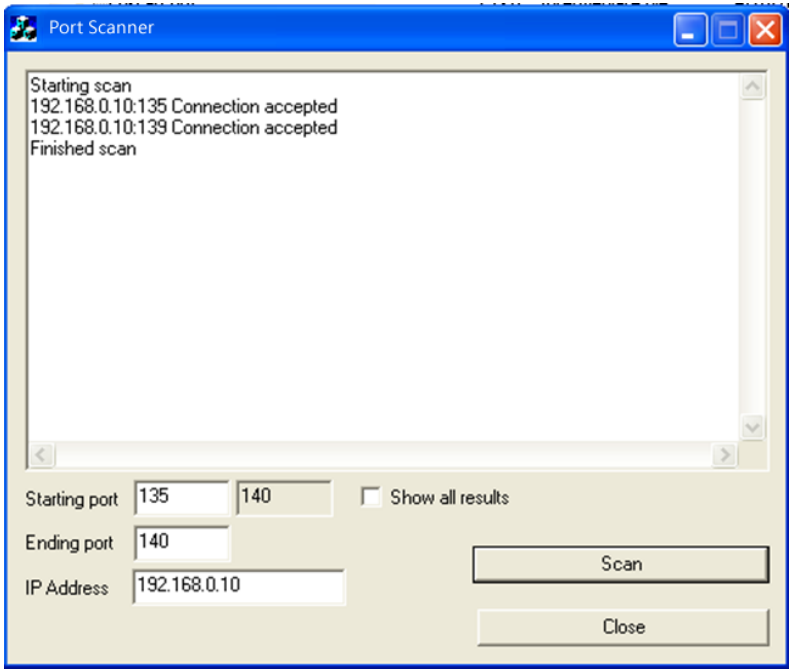
No Response? Filtered Ports

- If no response is received after several retransmissions, the port is marked as filtered.
- The port is also marked filtered if an ICMP unreachable error (type 3, code 1, 2, 3, 9, 10, or 13) is received.
- ICMP Message Type 3:

1 Host Unreachable [RFC792]
2 Protocol Unreachable [RFC792]
3 Port Unreachable [RFC792]
9 Communication with Destination Network is Administratively Prohibited [RFC1122]
10 Communication with Destination Host is Administratively Prohibited [RFC1122]
13 Communication Administratively Prohibited [RFC1812]

PortScanner





http://nmap.org

References

- John Kristoff's Overview of TCP (Fundamental concepts behind TCP and how it is used to transport data between two endpoints):
  http://condor.depaul.edu/~jkristof/technotes/tcp.html
- RFC 793 - TCP v4
- RFC 1122 - includes some error corrections for TCP
- RFC 1323 - TCP-Extensions
- RFC 2018 - TCP Selective Acknowledgment Options
- RFC 4614 - A Roadmap for TCP Specification Documents
- TCP, Transmission Control Protocol : http://www.networksorcery.com/enp/protocol/tcp.htm
- **TCP Checksum Calculation and the TCP "Pseudo Header"** :
  http://www.tcpipguide.com/free/t_TCPChecksumCalculationandtheTCPPseudoHeader-2.htm.
- Checksum example : http://mathforum.org/library/drmath/view/54379.html
- IANA Port Assignments : http://www.iana.org/assignments/port-numbers
- W. Richard Stevens. TCP/IP Illustrated, Volume 1/II
- Andrew S. Tanenbaum. Computer Networks
- Wiki : Transmission Control Protocol :
  http://en.wikipedia.org/wiki/Transmission_Control_Protocol