TCP: Transport Control Protocol

End-to-end reliable transmission with flow and congestion control

Based on textbook Conceptual Computer Networks

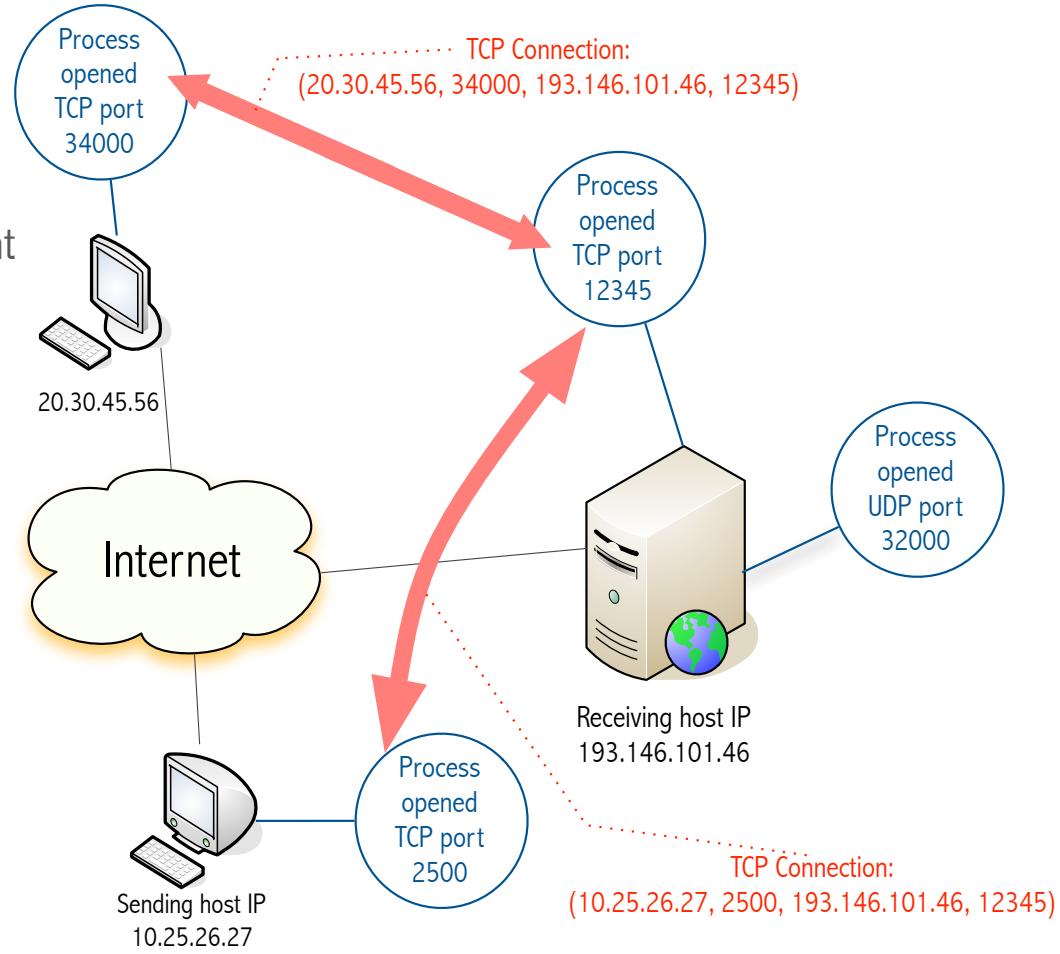
Reliable Byte Stream (TCP)

In contrast to UDP, TCP offers the following distinctive characteristics:

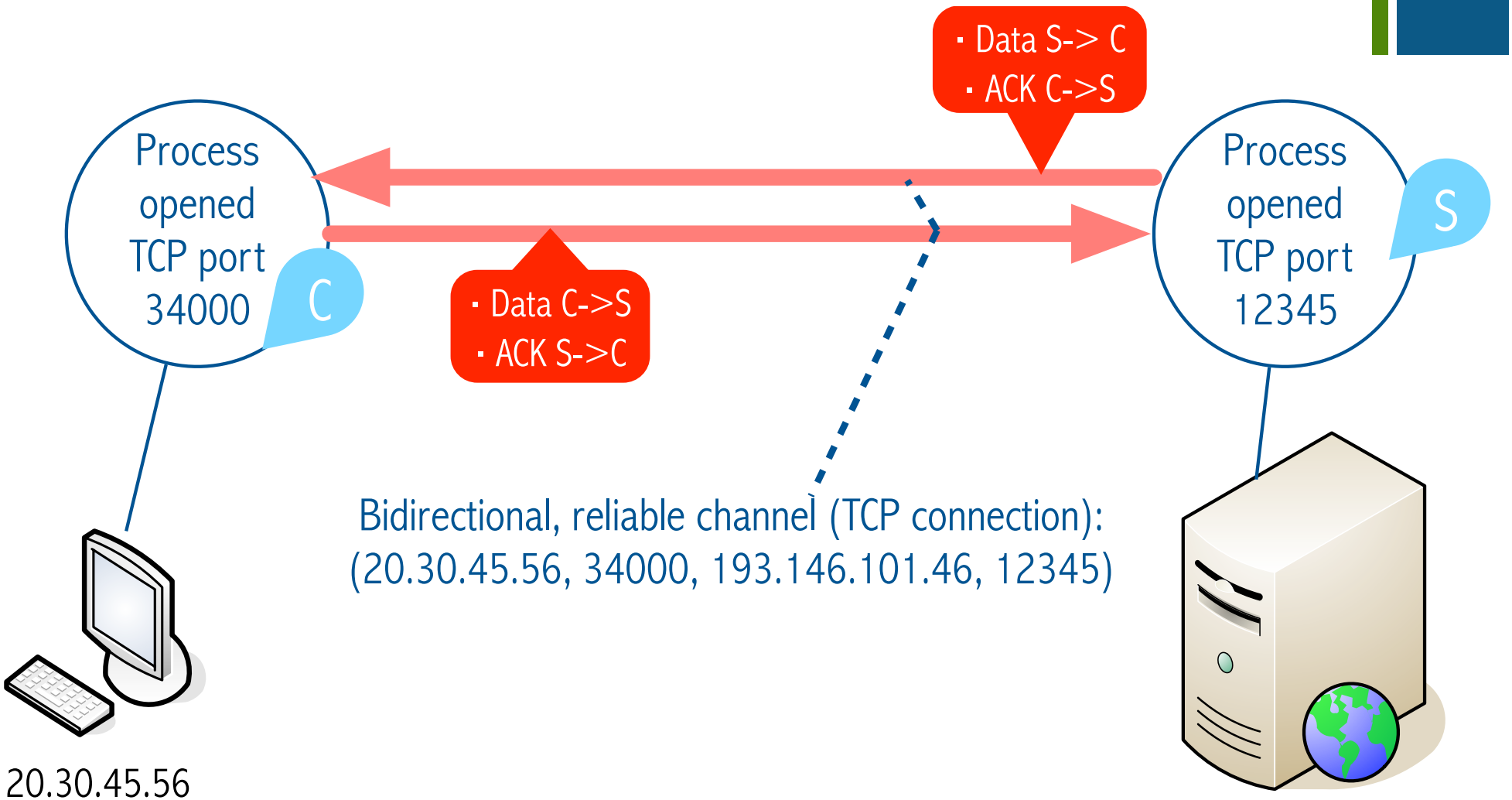
- Reliable
- Connection oriented
- Byte-stream service
- Flow control involves preventing senders from overrunning the capacity of the receivers
- Congestion control involves preventing too much data from being injected into the network, thereby causing switches or links to become overloaded

End-to-end issues in TCP

- TCP supports logical connections between processes that are running on two different computers in the Internet
- TCP connections are likely to have widely different RTT times
- Packets may get reordered in the Internet
- Flow Control
  - Each side of a connection will learn what resources the other side is able to apply to the connection
- Congestion Control
  - Each sending side of a connection will learn the capacity of the network

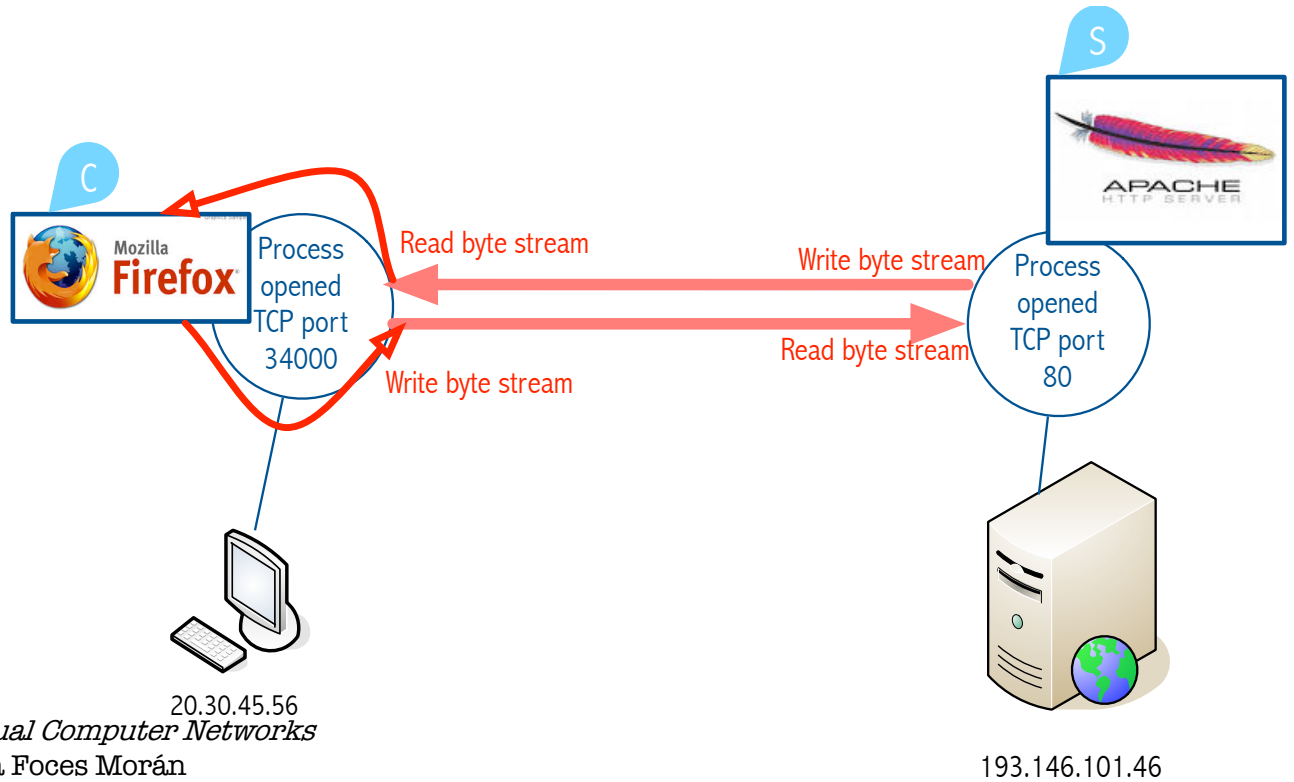


TCP Connection: Reliable channel



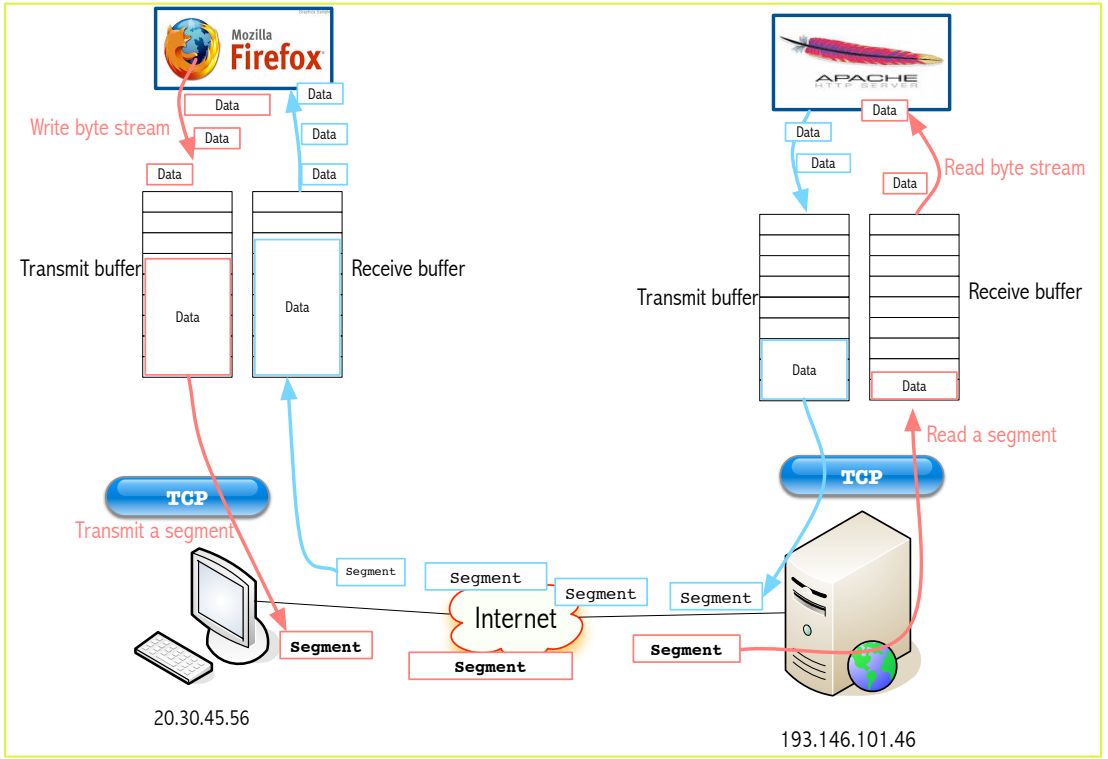
TCP is a byte oriented protocol

- C and S send bytes into the stream and also receive bytes

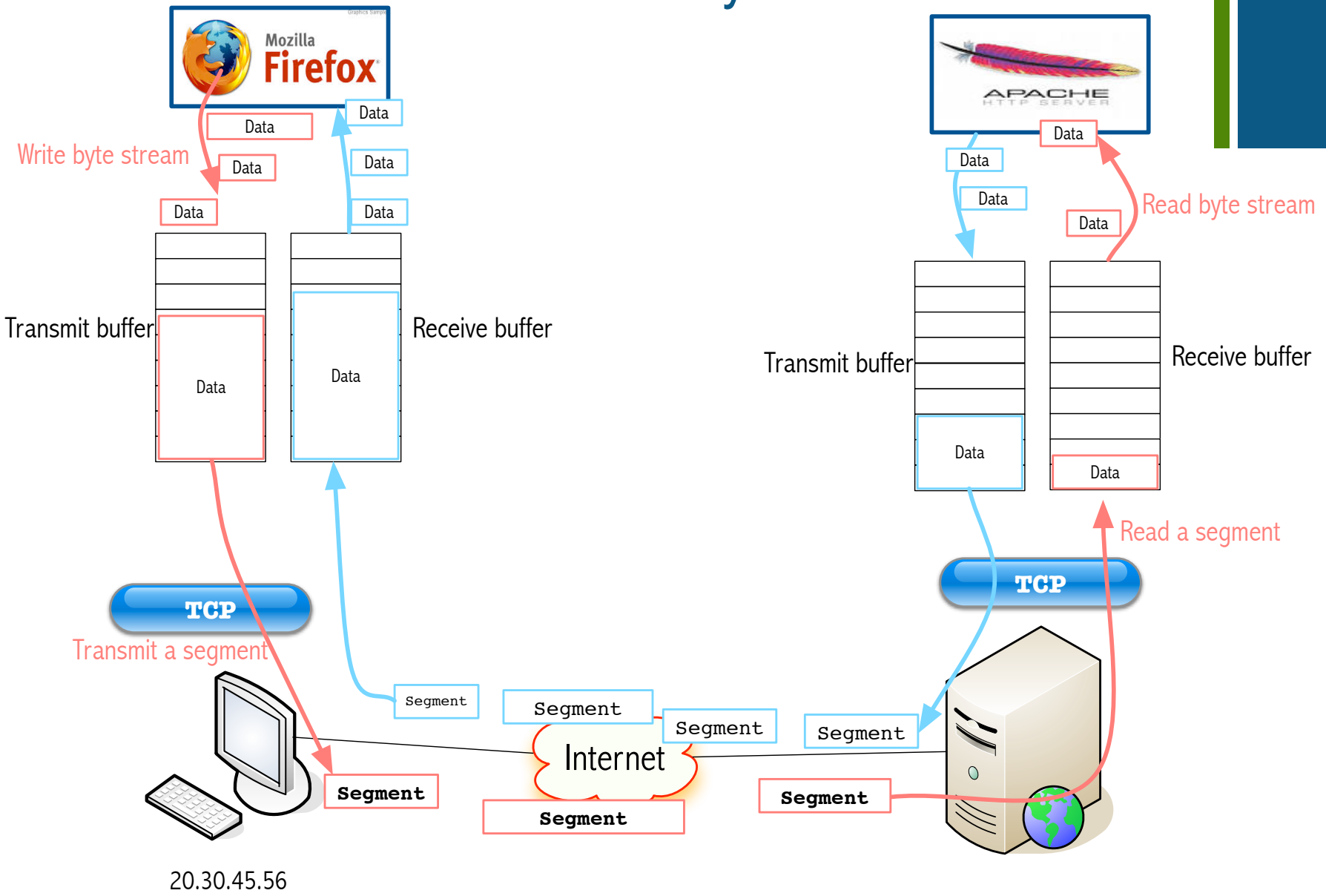


TCP buffers data before sending or receiving

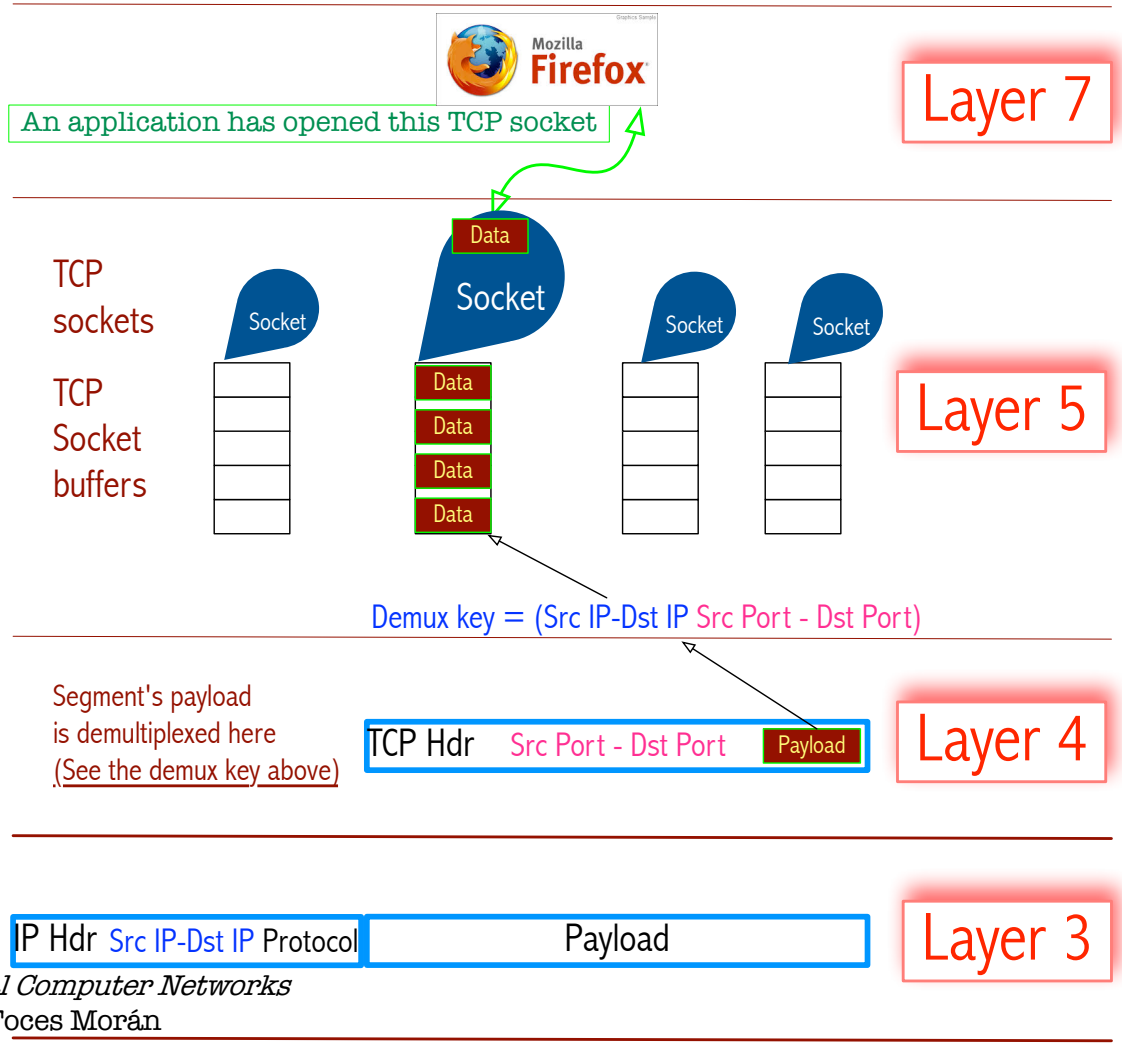
- TCP on the source host buffers enough bytes from the sending process to fill a reasonably sized packet and then sends this packet to its peer on the destination host.
- TCP on the destination host then empties the contents of the packet into a **receive buffer**, and the receiving process reads from this buffer at its leisure.
- The packets exchanged between TCP peers are called **segments**



TCP: Thread to thread delivery



TCP Demultiplexing key: (Src Ip, Src port, Dest Ip, Dest port)



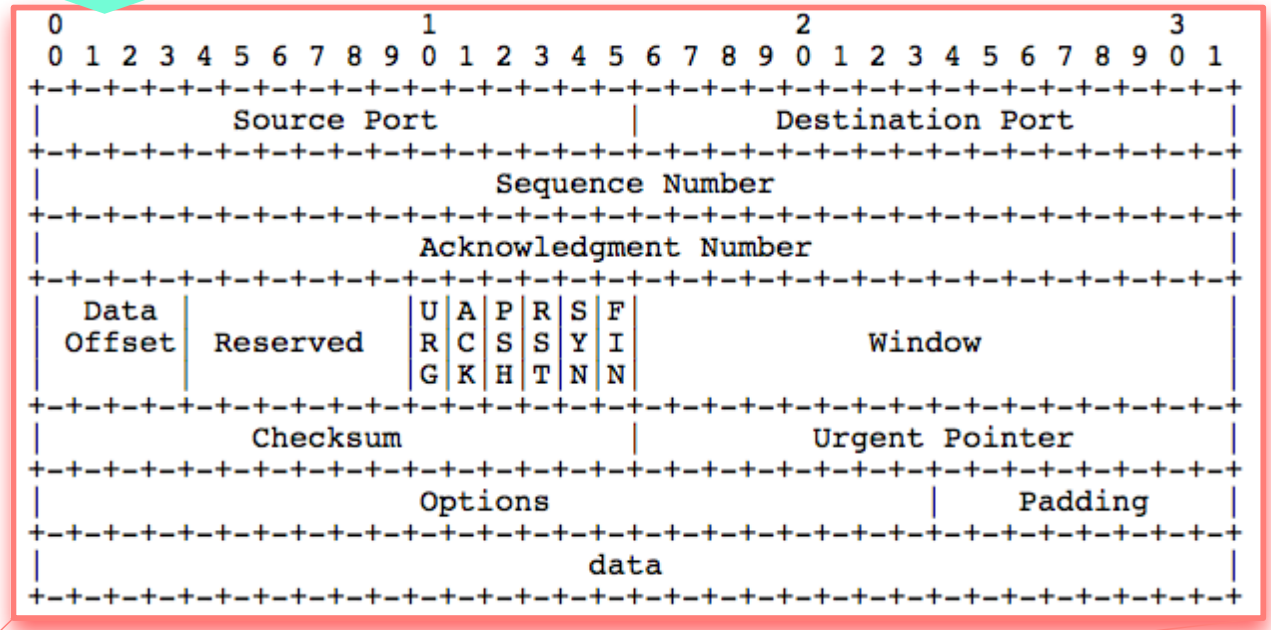
Based on textbook Conceptual Computer Networks

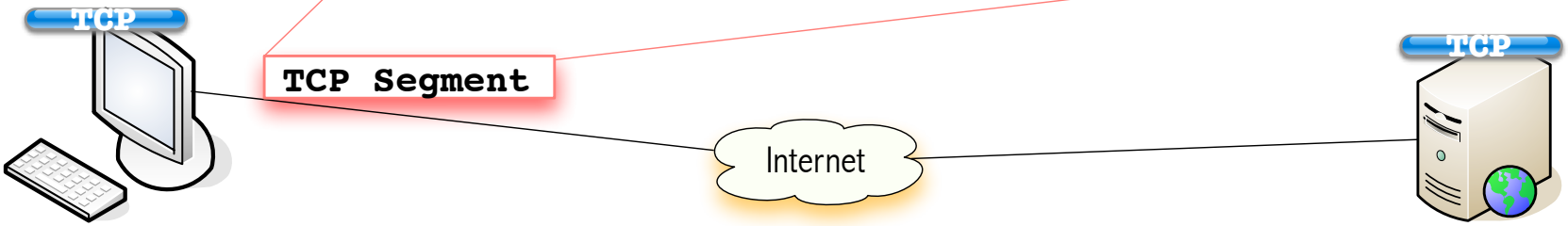
The TCP protocol data unit

Segment

TCP Segment Header

From RFC 793 by Jon Postel, 1981



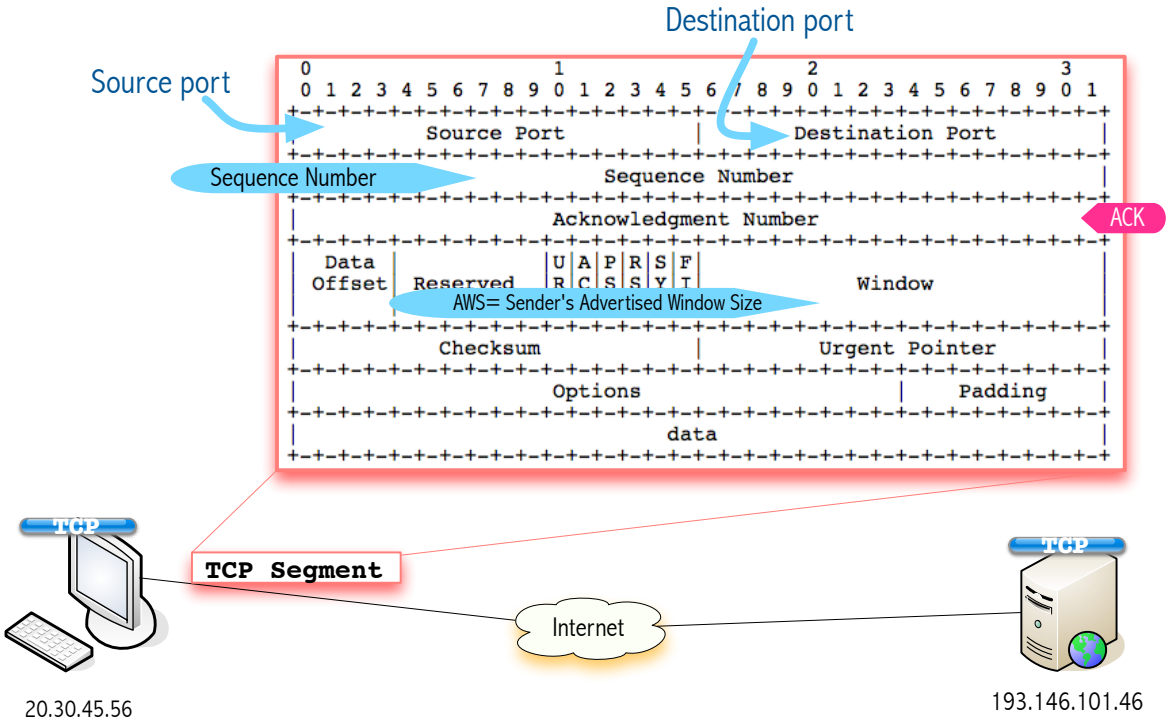


20.30.45.56

193.146.101.46

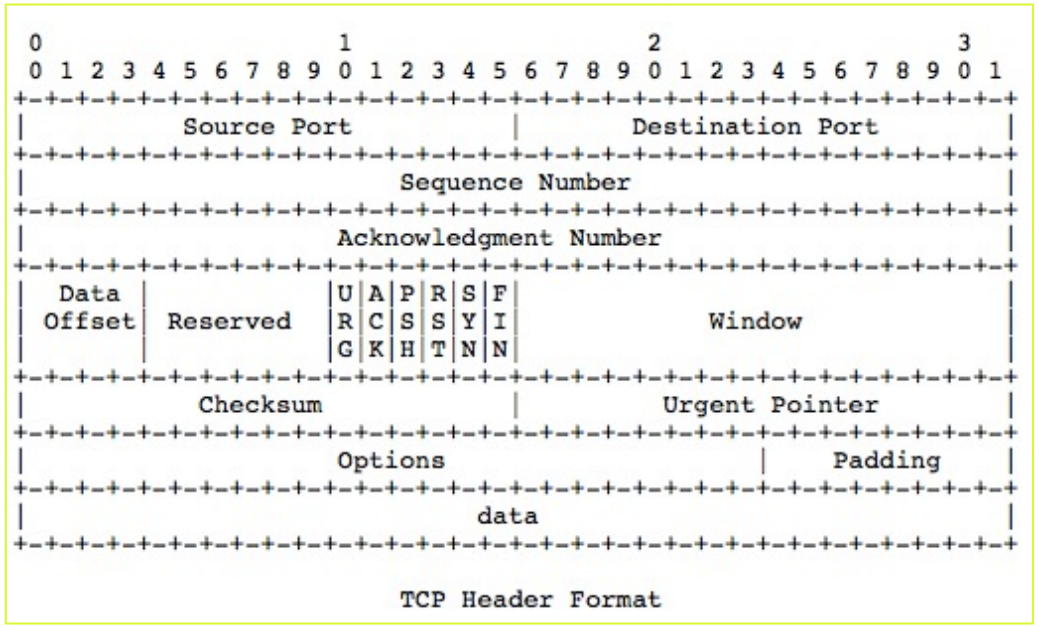
TCP Segment Header

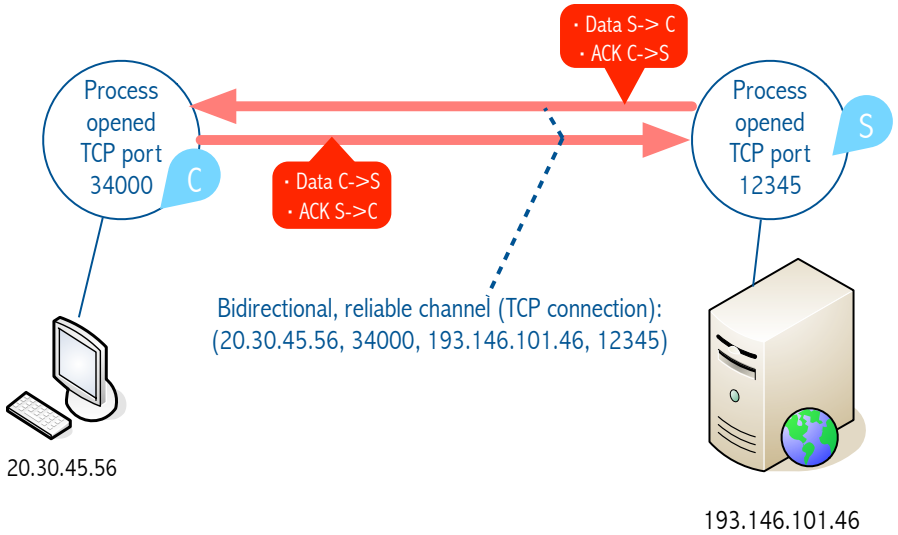
- SrcPort
  - Source port opened by sending thread
- DstPort
  - Destination port opened by receiving thread
- SequenceNum, Acknowledgment
  - Piggybacking data + acks
- AdvertisedWindow
  - Flow control on the sender



TCP Segment Header
Acknowledgements (ACKs)

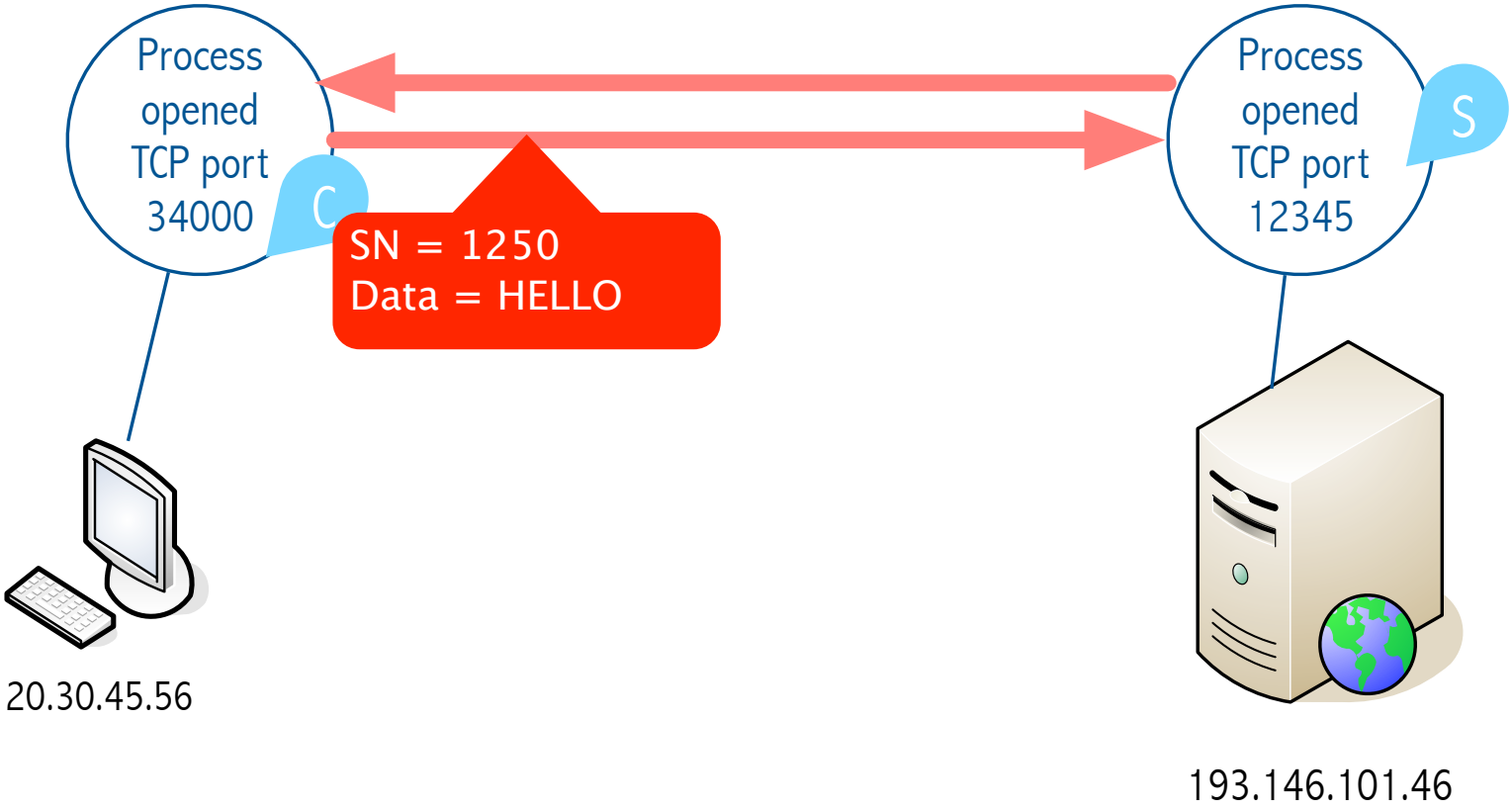
- TCP is a byte-oriented protocol: **each byte of data has an *implicit* sequence number**
  - SequenceNum: the sequence number for the first byte of data carried in a segment.
- Acknowledgment and AdvertisedWindow fields: information about the flow of data going in the other direction: **PiggyBacking**





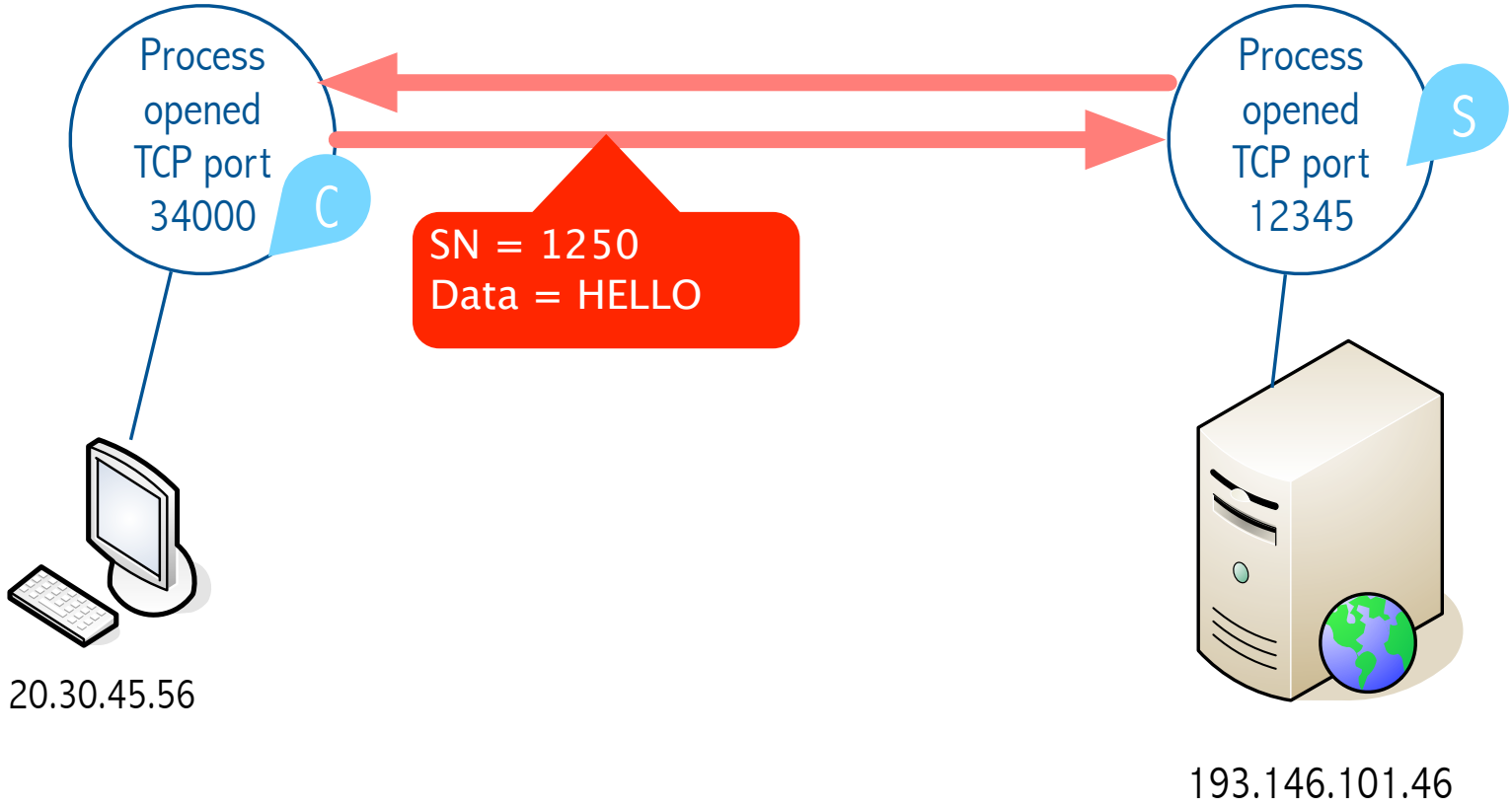
Essential dynamics of TCP

- When TCP receives a new segment, *soon*, it will ***ack it:***
  - Send a segment acknowledging the received segment



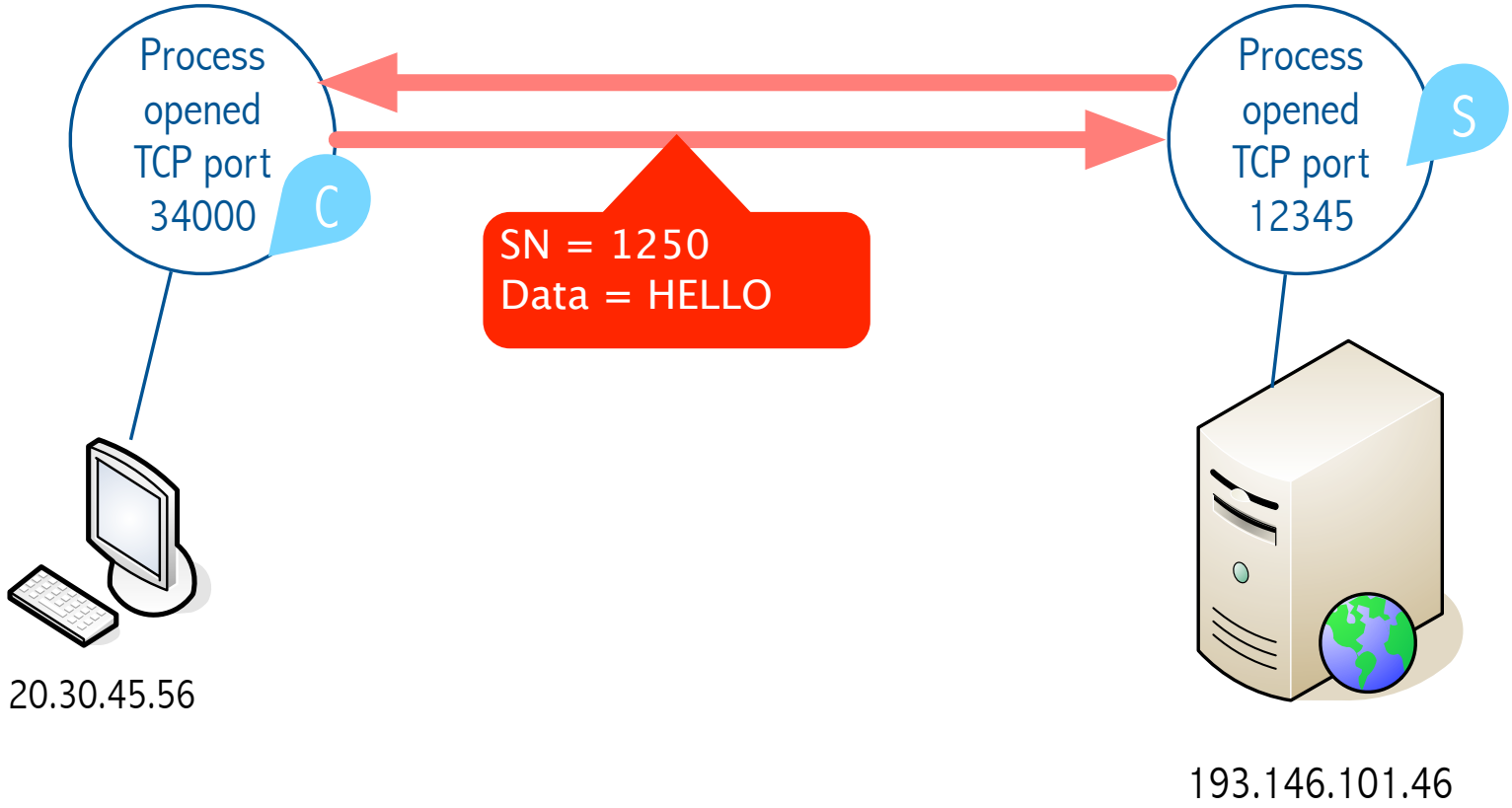
Essential dynamics of TCP

- When TCP receives a new segment, *soon*, it will ***ack it:***
  - Send a segment acknowledging the received segment



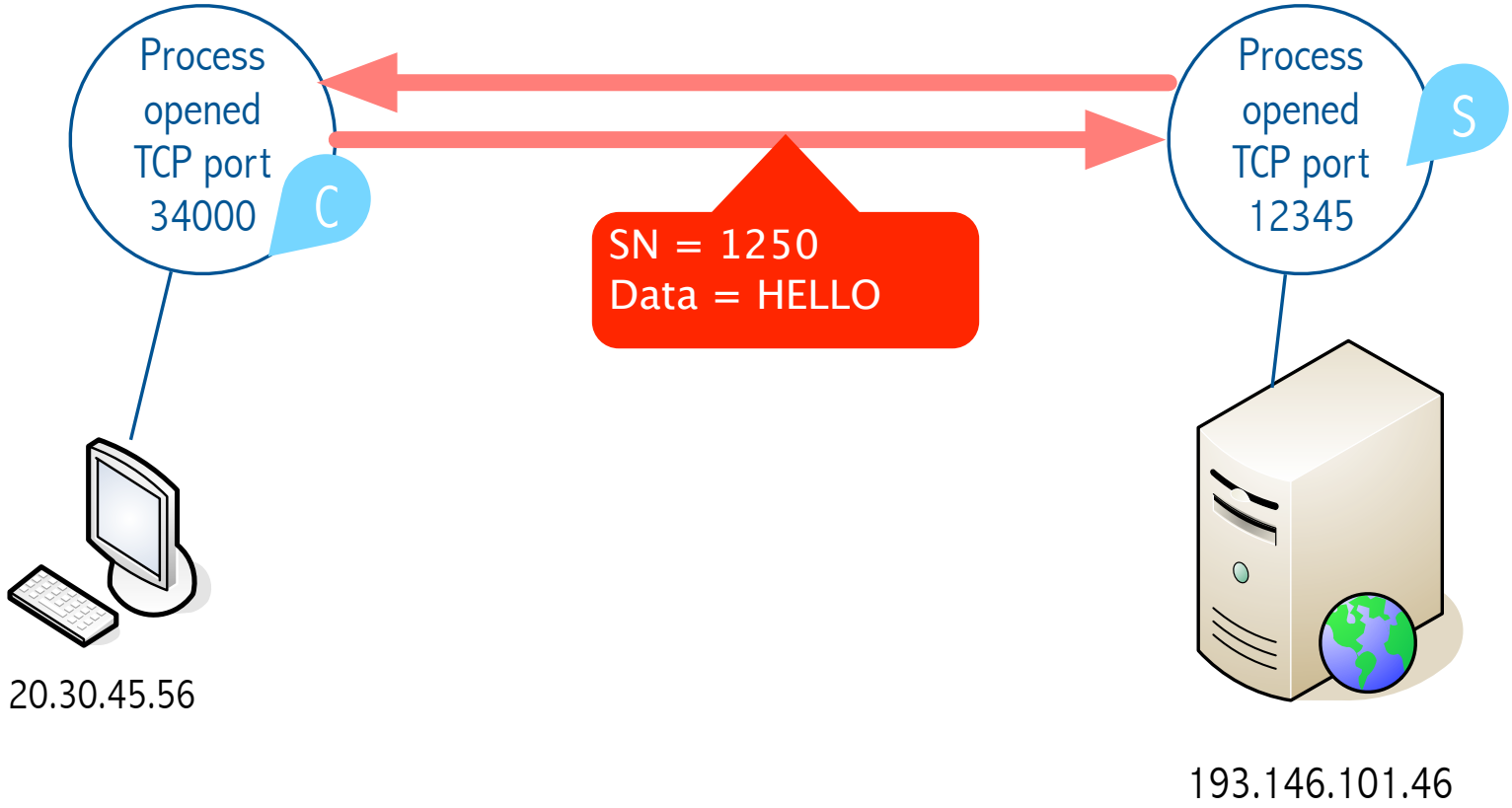
Essential dynamics of TCP

- When TCP receives a new segment, *soon*, it will ***ack it:***
  - Send a segment acknowledging the received segment



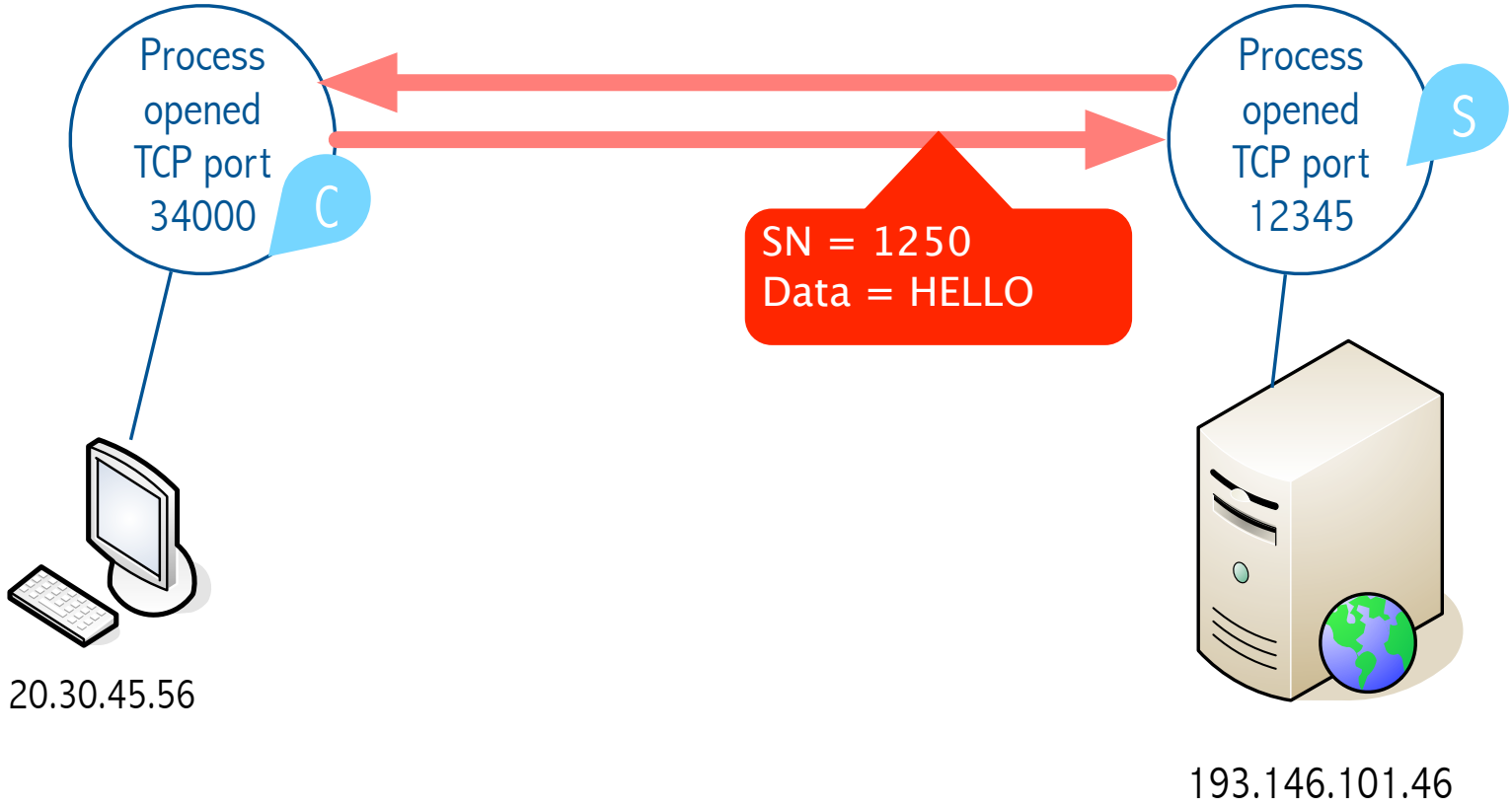
Essential dynamics of TCP

- When TCP receives a new segment, *soon*, it will ***ack it:***
  - Send a segment acknowledging the received segment



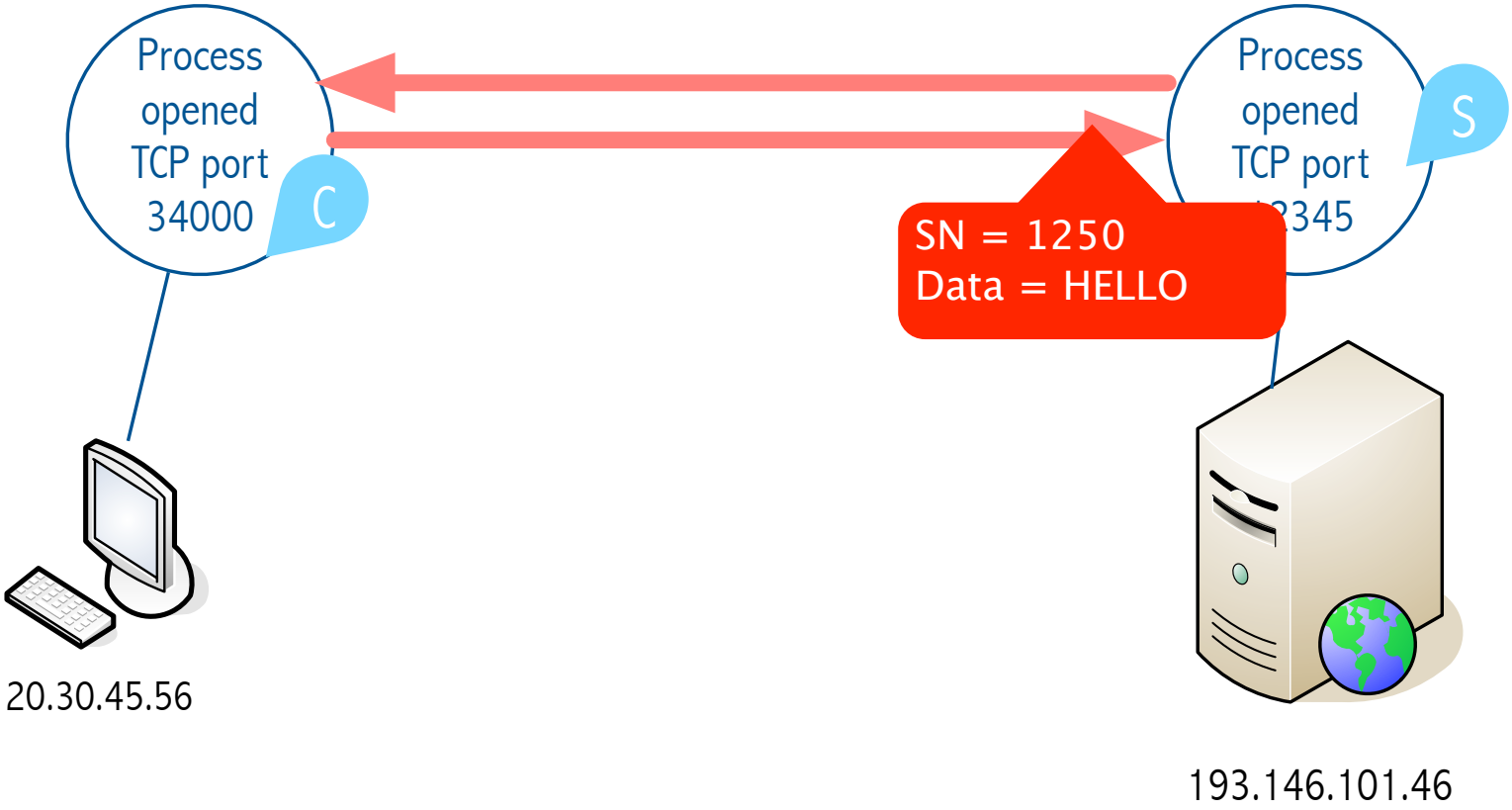
Essential dynamics of TCP

- When TCP receives a new segment, *soon*, it will ***ack it:***
  - Send a segment acknowledging the received segment



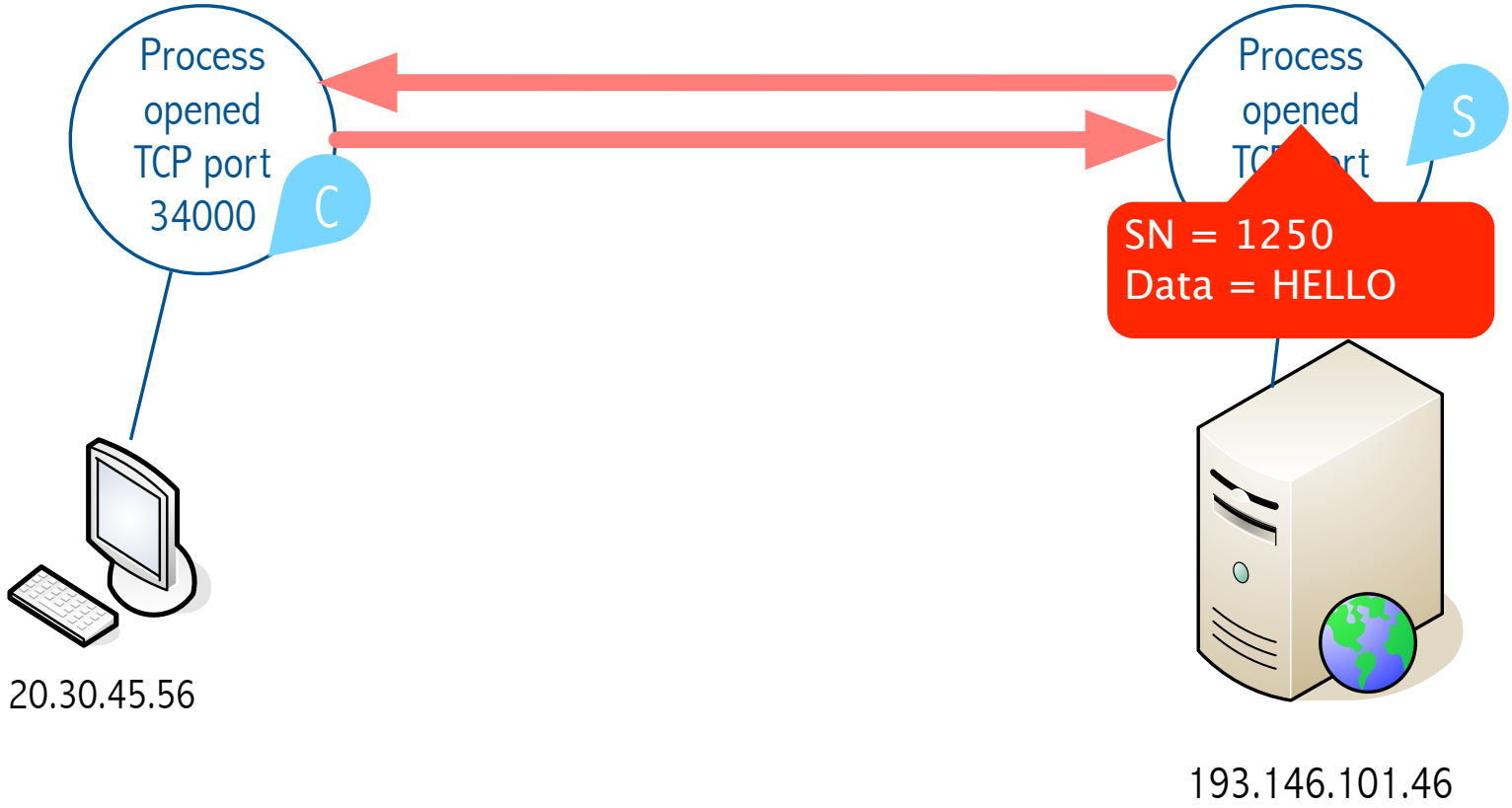
Essential dynamics of TCP

- When TCP receives a new segment, *soon*, it will ***ack it:***
  - Send a segment acknowledging the received segment



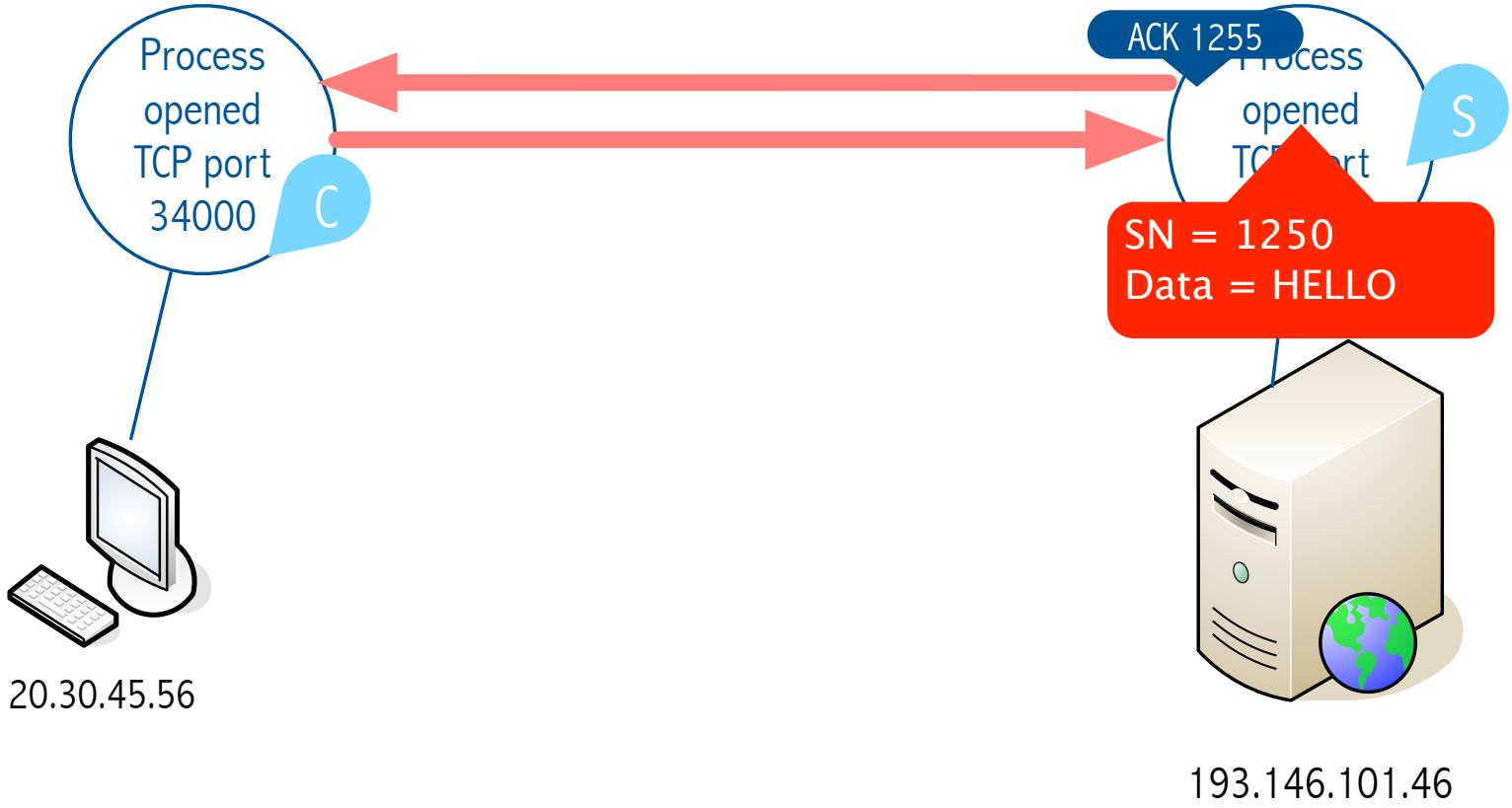
Essential dynamics of TCP

- When TCP receives a new segment, *soon*, it will ***ack it:***
  - Send a segment acknowledging the received segment



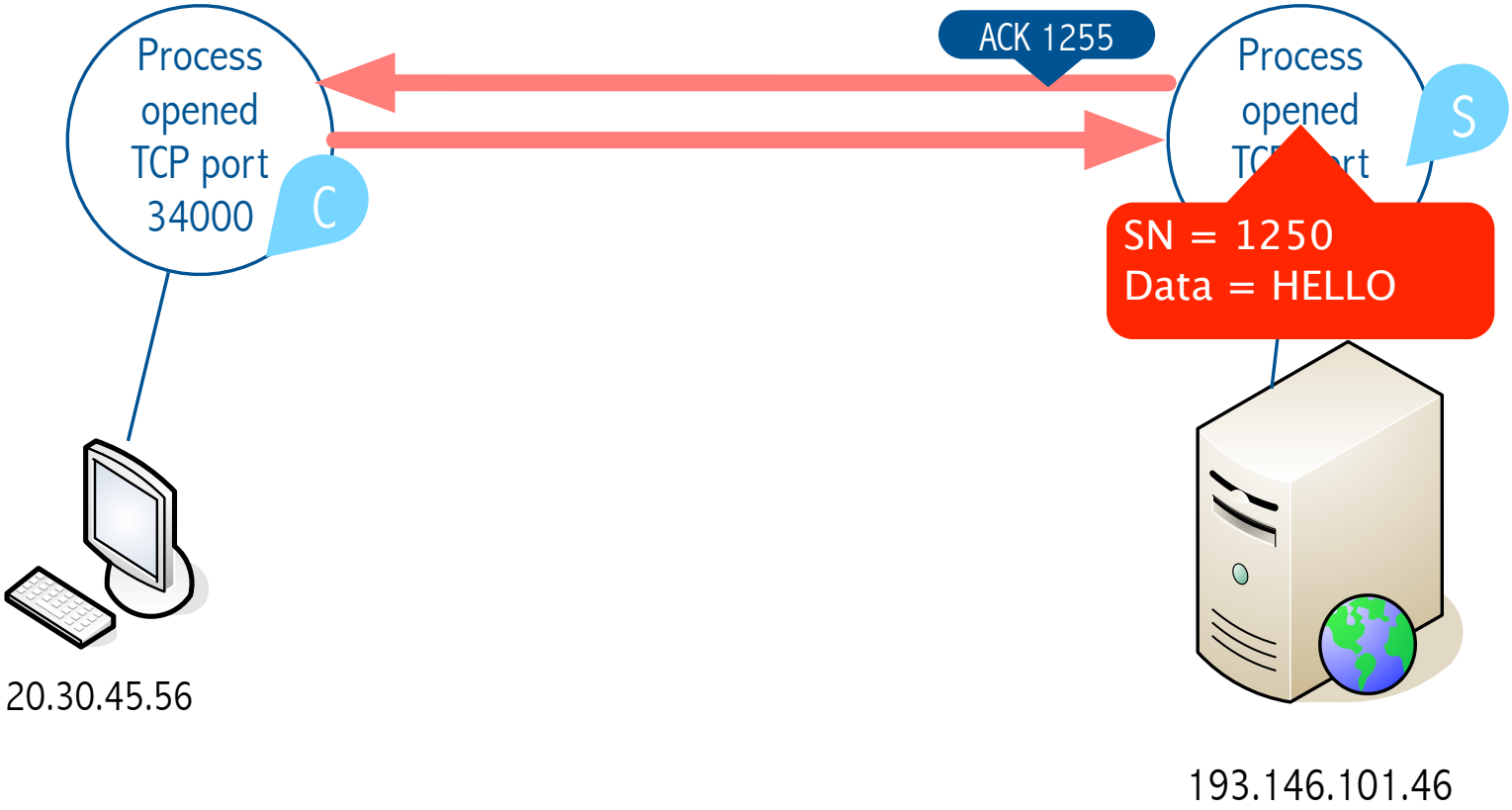
Essential dynamics of TCP

- When TCP receives a new segment, *soon*, it will ***ack it:***
  - Send a segment acknowledging the received segment



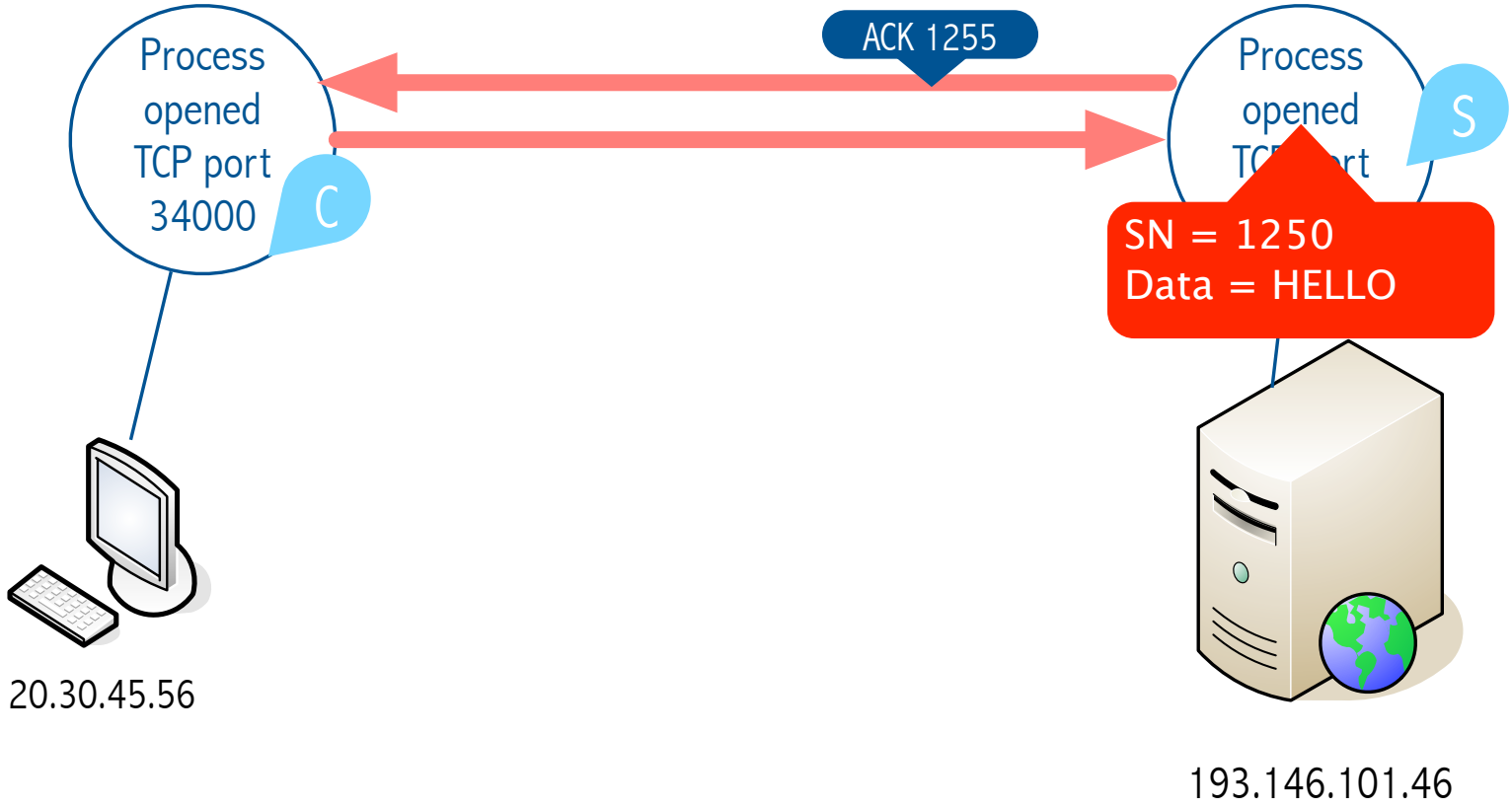
Essential dynamics of TCP

- When TCP receives a new segment, *soon*, it will ***ack it:***
  - Send a segment acknowledging the received segment



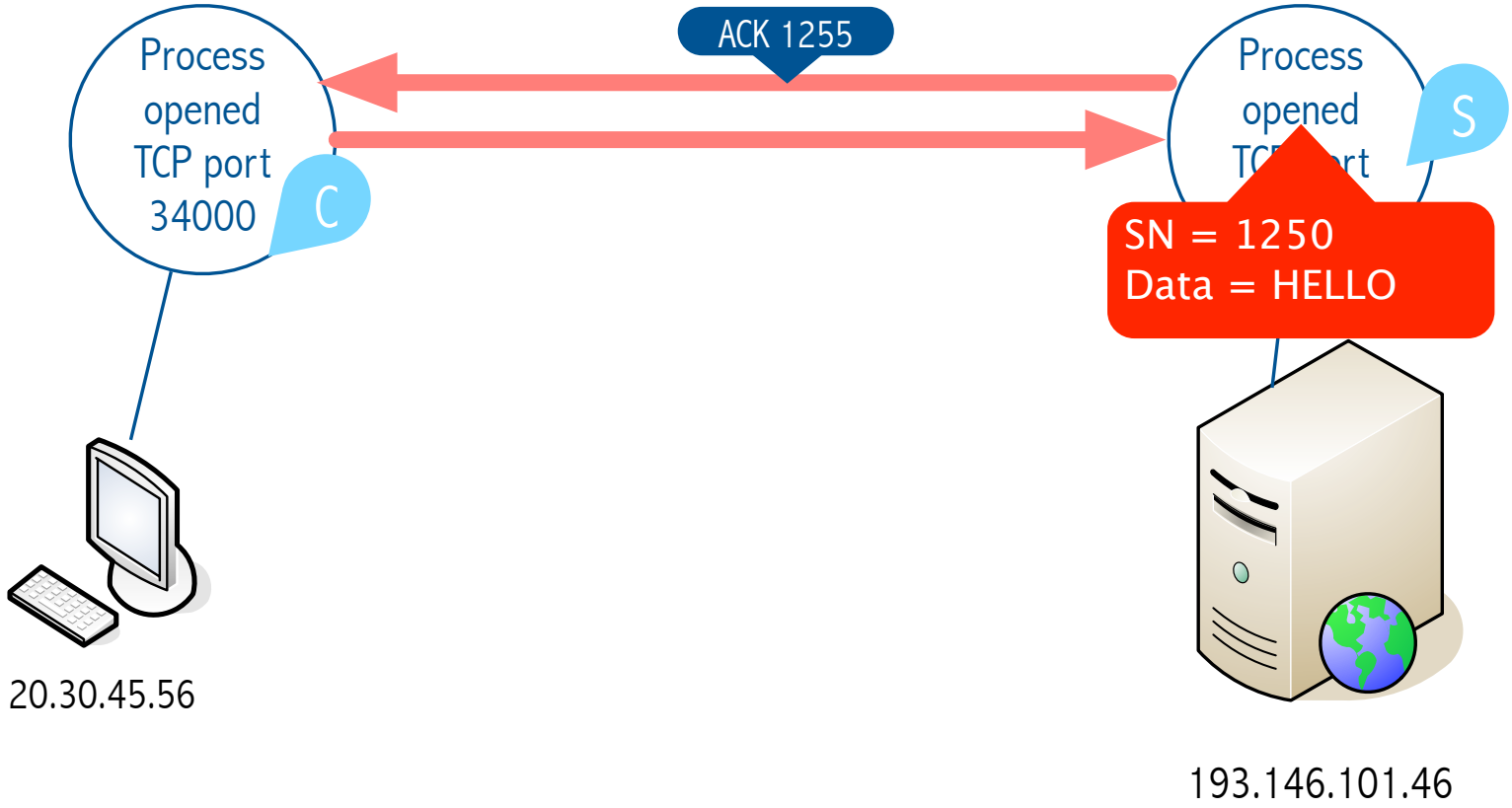
Essential dynamics of TCP

- When TCP receives a new segment, it soon will send an ACK for that segment



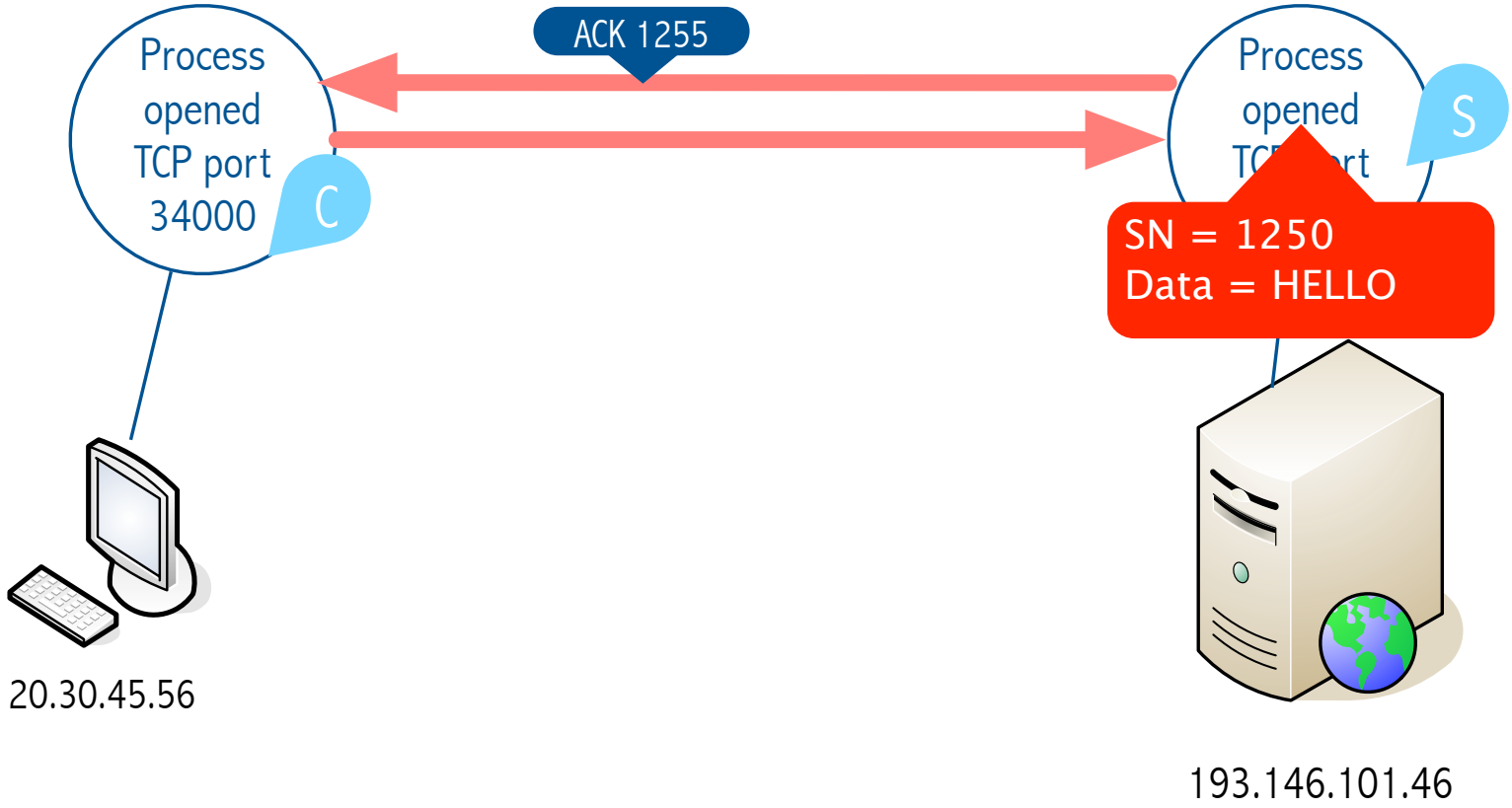
Essential dynamics of TCP

- When TCP receives a new segment, it soon will send an ACK for that segment



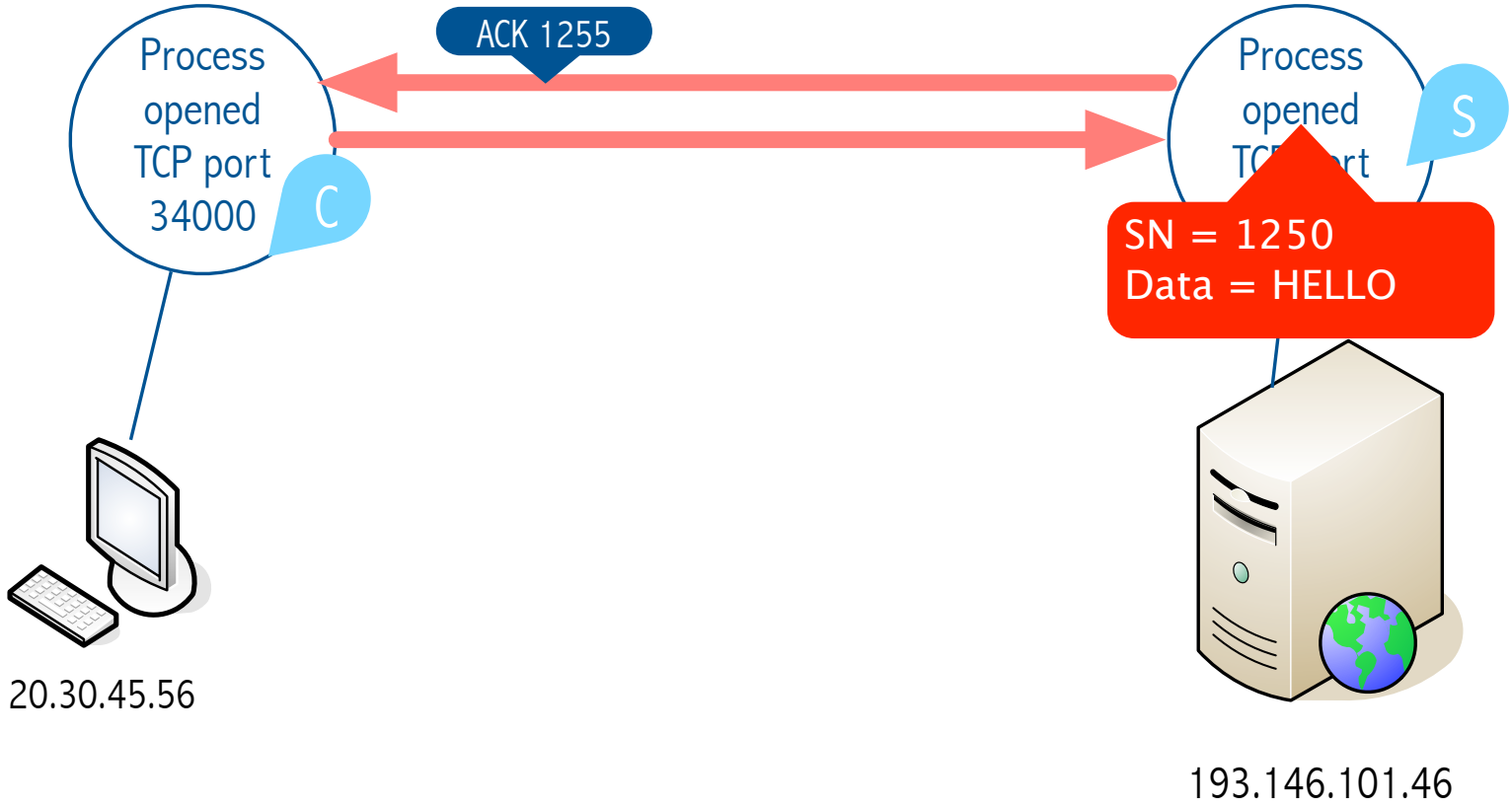
Essential dynamics of TCP

- When TCP receives a new segment, it soon will send an ACK for that segment



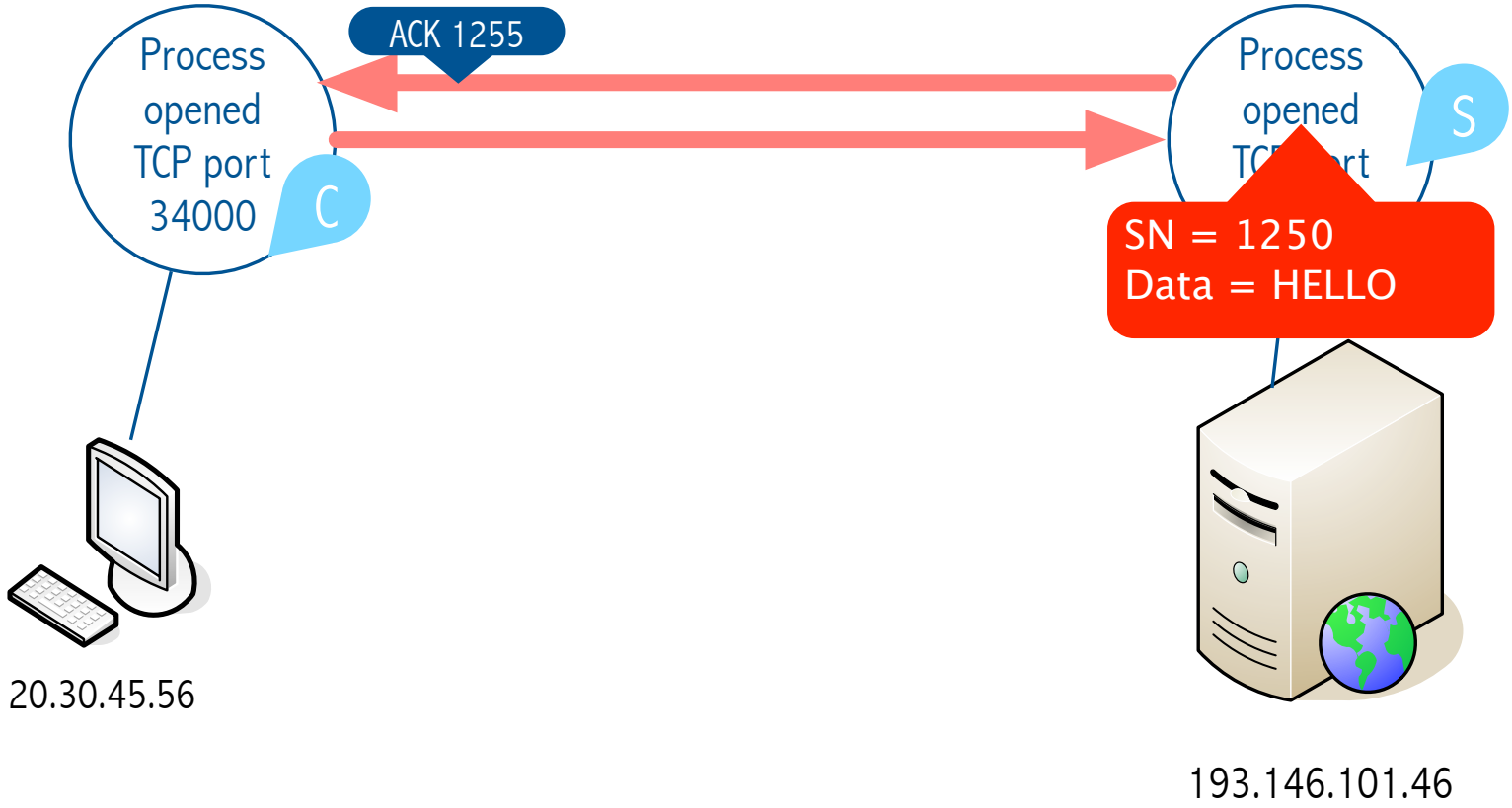
Essential dynamics of TCP

- When TCP receives a new segment, it soon will send an ACK for that segment



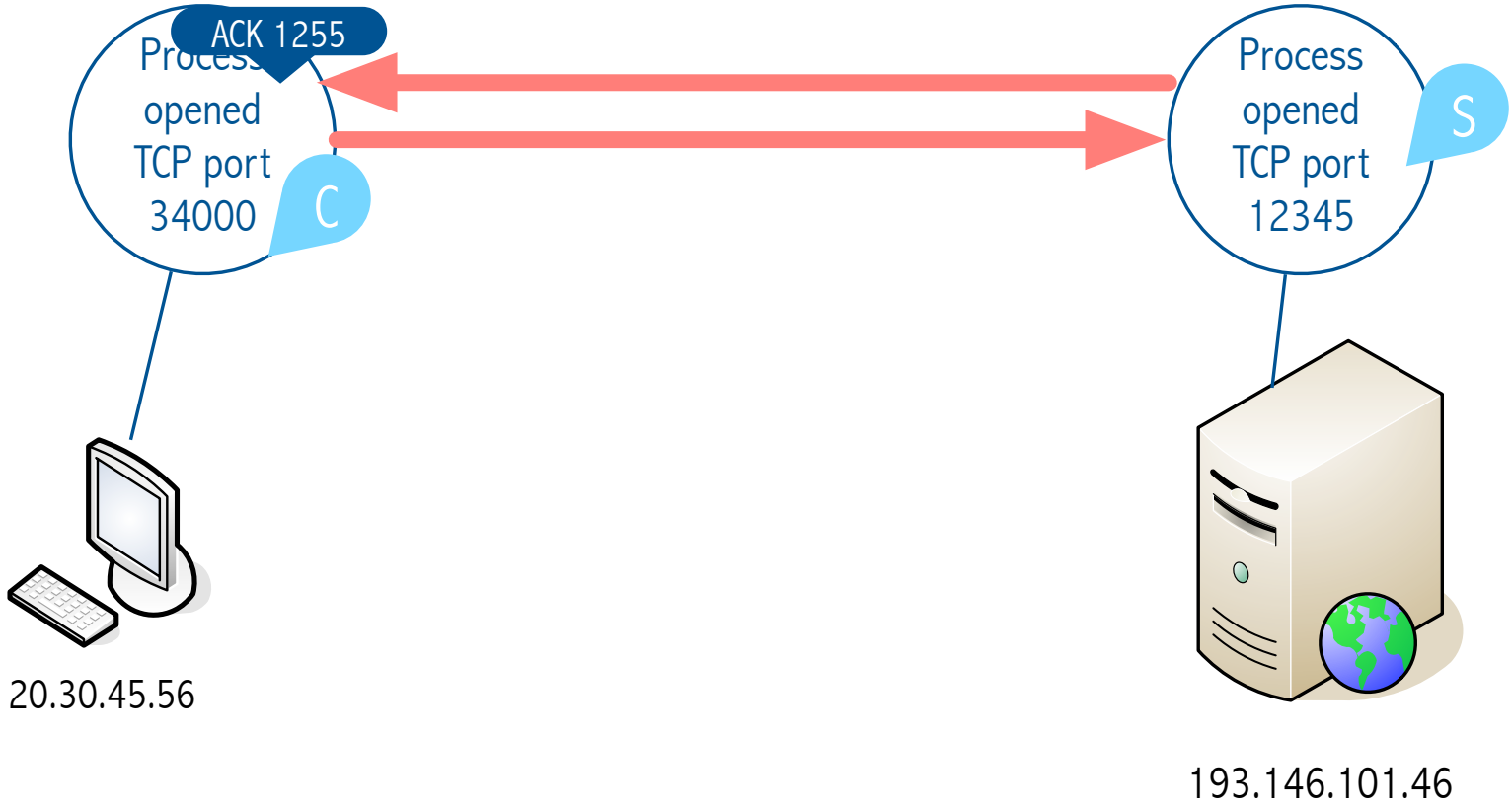
Essential dynamics of TCP

- When TCP receives a new segment, it soon will send an ACK for that segment



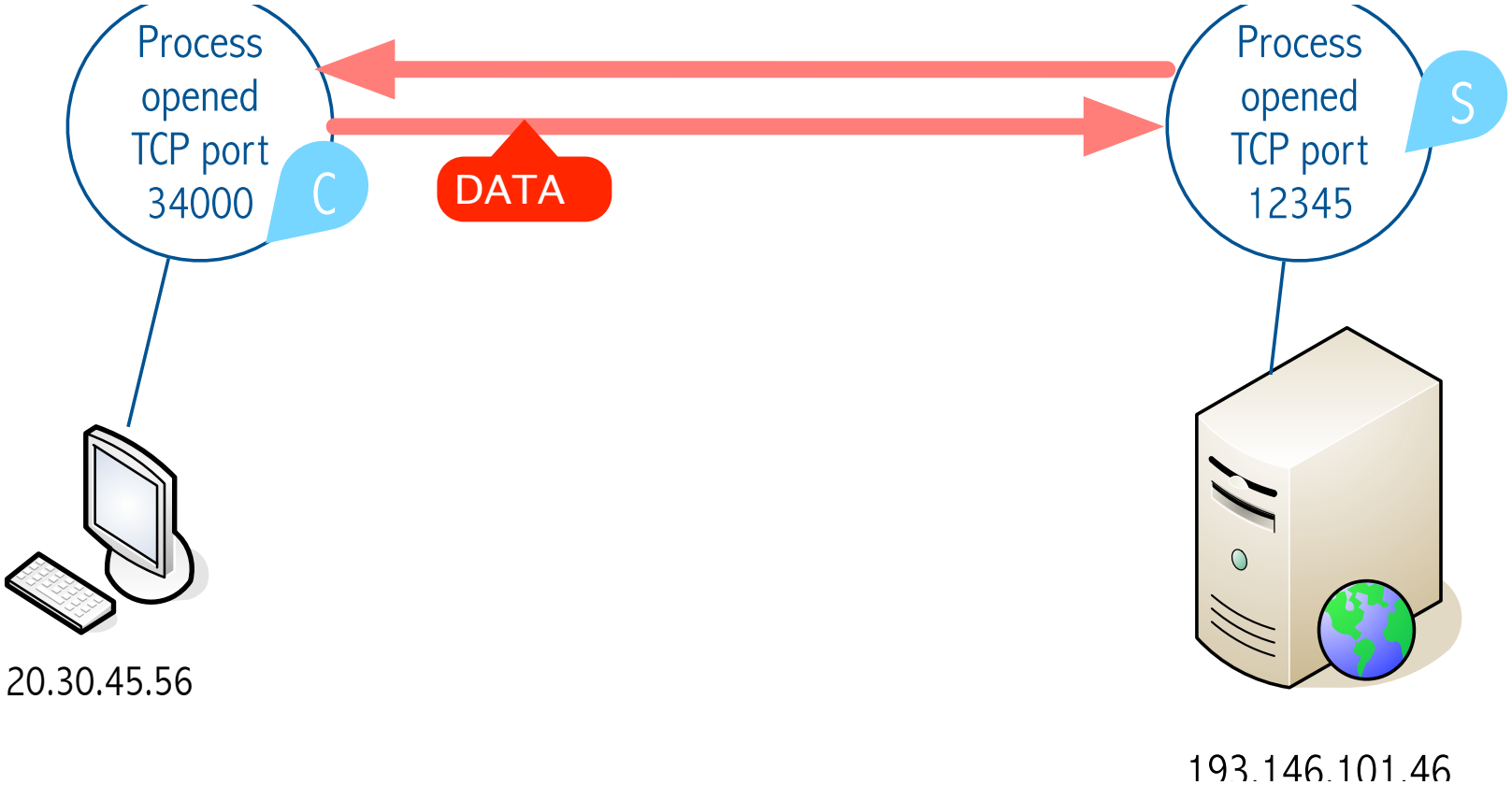
Essential dynamics of TCP

- When TCP receives a new segment, it soon will send an ACK for that segment



Essential dynamics of TCP

- When TCP receives a new segment, it soon will send an ACK for that segment

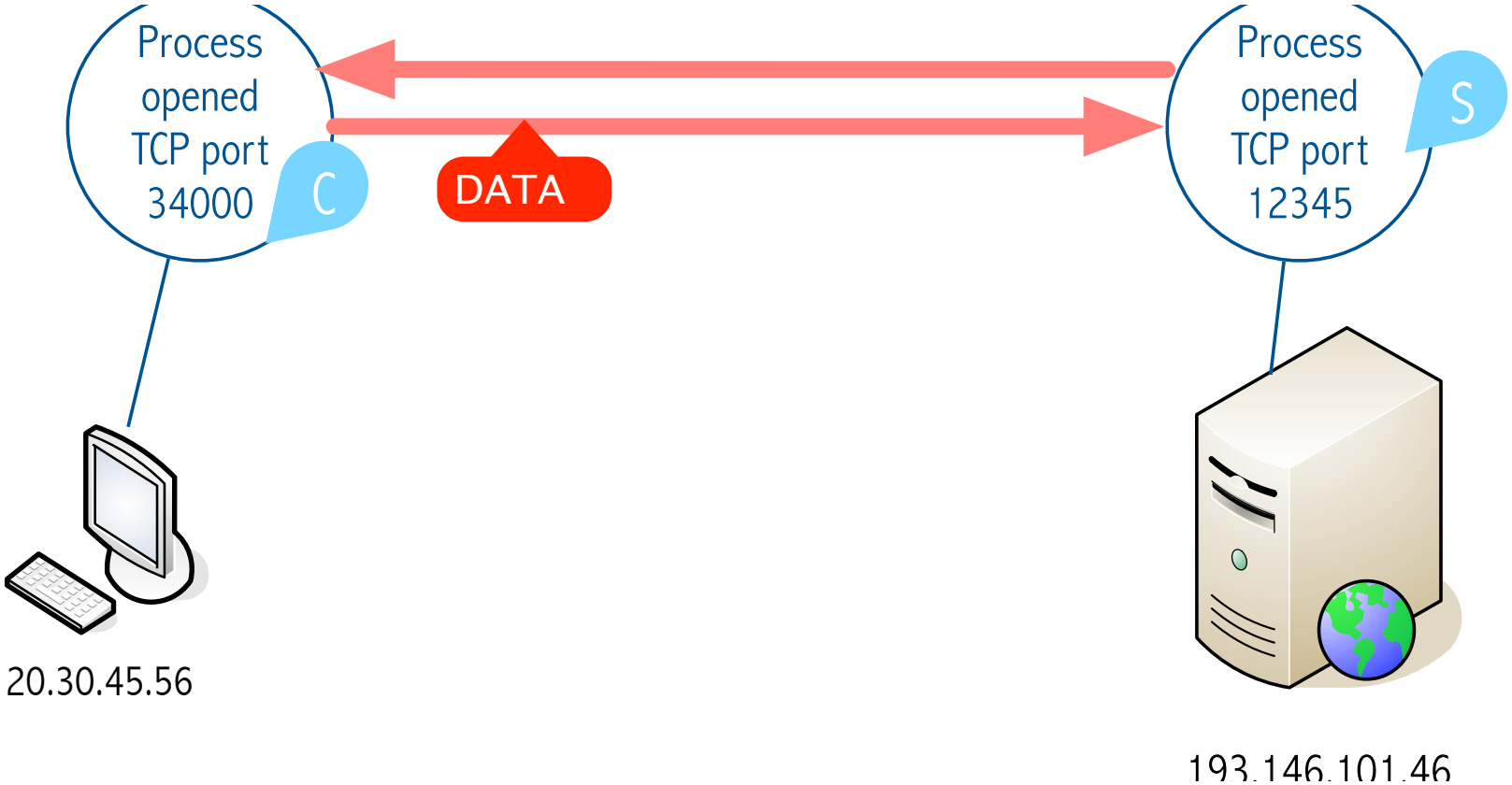


Essential dynamics of TCP

- When TCP receives a new segment, it soon will send an ACK for that segment

Essential dynamics of TCP

- When TCP receives a new segment, it soon will send an ACK for that segment

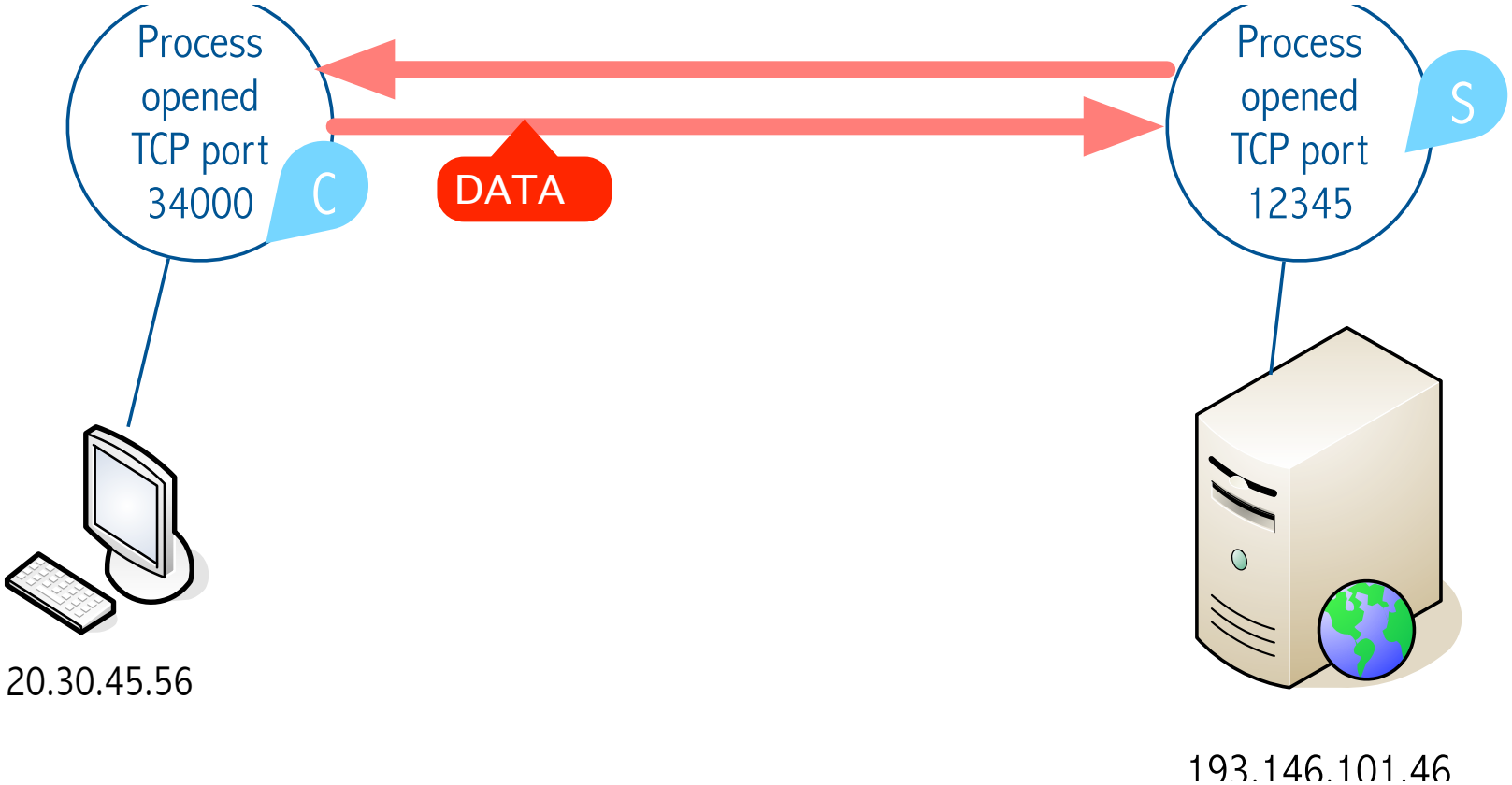


Essential dynamics of TCP

- When TCP receives a new segment, it soon will send an ACK for that segment

Essential dynamics of TCP

- When TCP receives a new segment, it soon will send an ACK for that segment

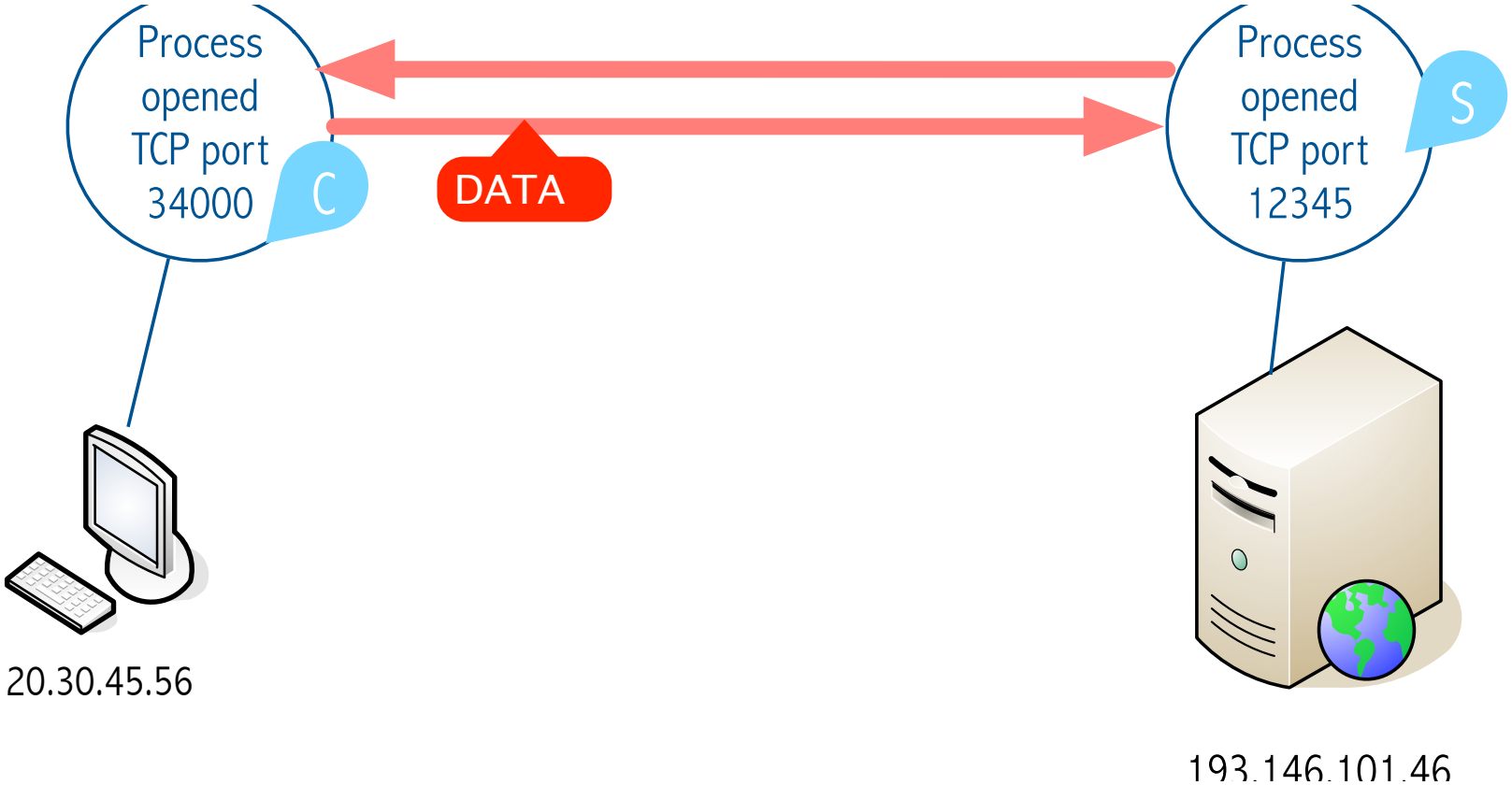


Essential dynamics of TCP

- When TCP receives a new segment, it soon will send an ACK for that segment

Essential dynamics of TCP

- When TCP receives a new segment, it soon will send an ACK for that segment

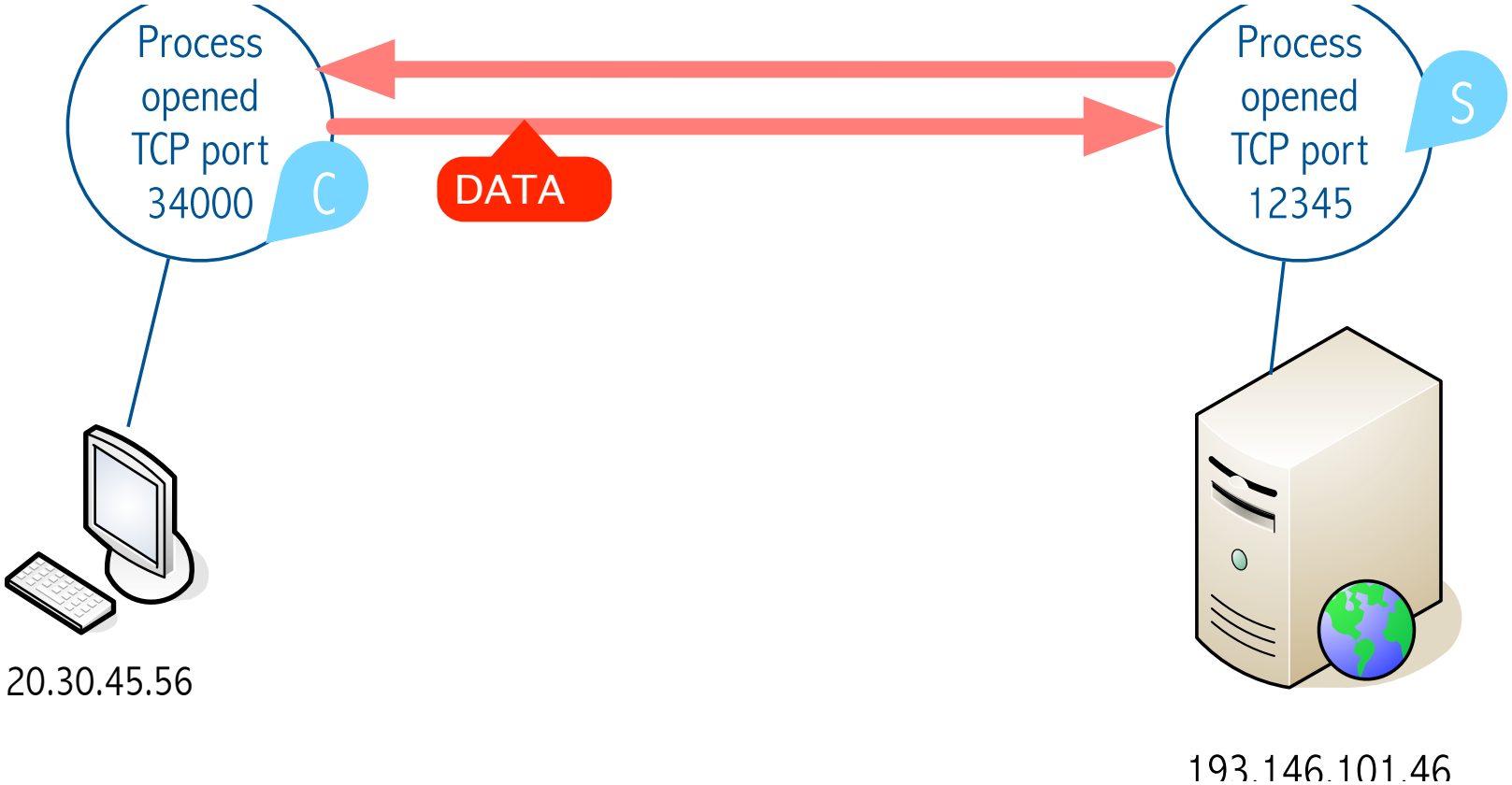


Essential dynamics of TCP

- When TCP receives a new segment, it soon will send an ACK for that segment

Essential dynamics of TCP

- When TCP receives a new segment, it soon will send an ACK for that segment

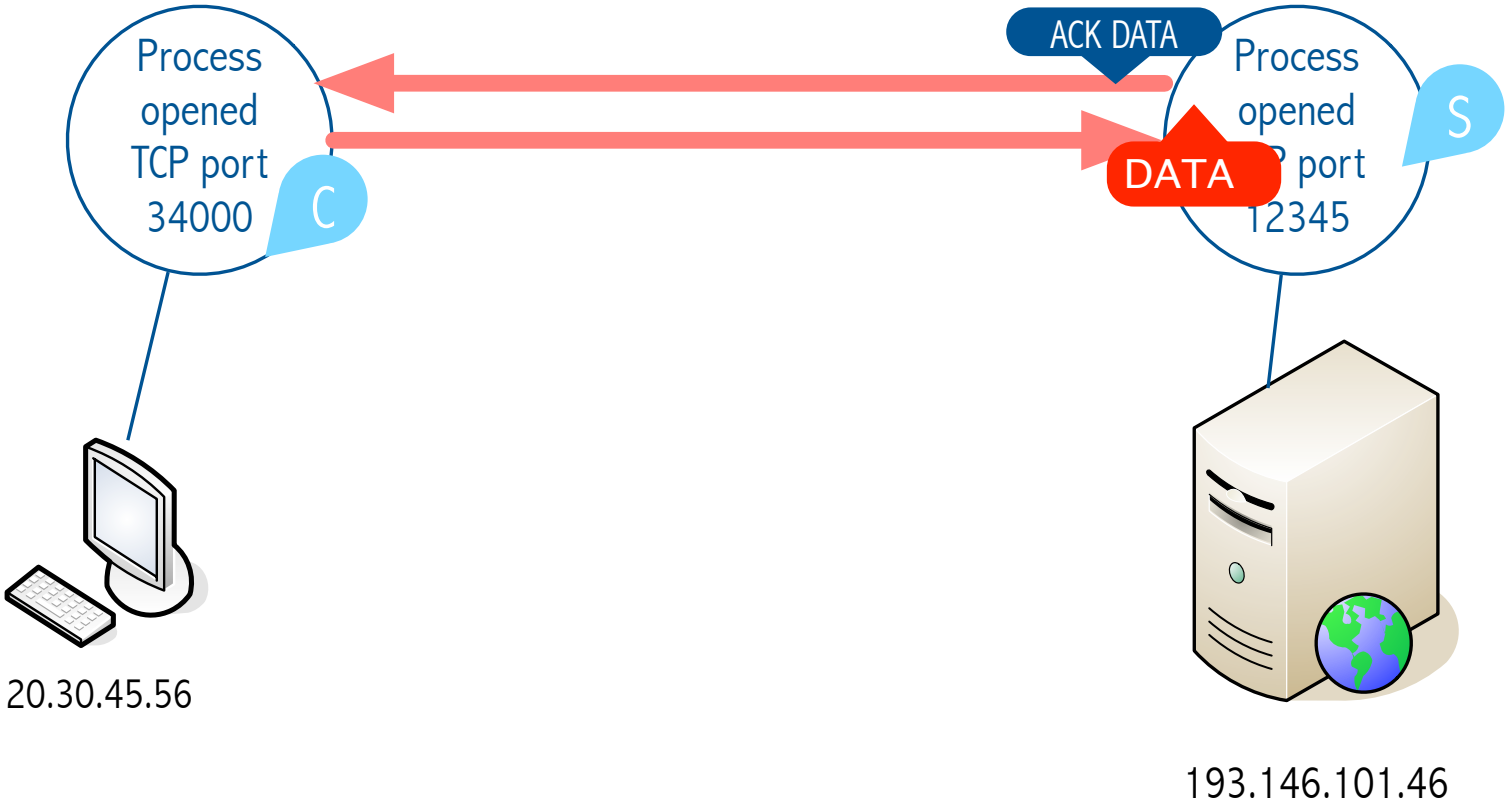


Essential dynamics of TCP

- When TCP receives a new segment, it soon will send an ACK for that segment

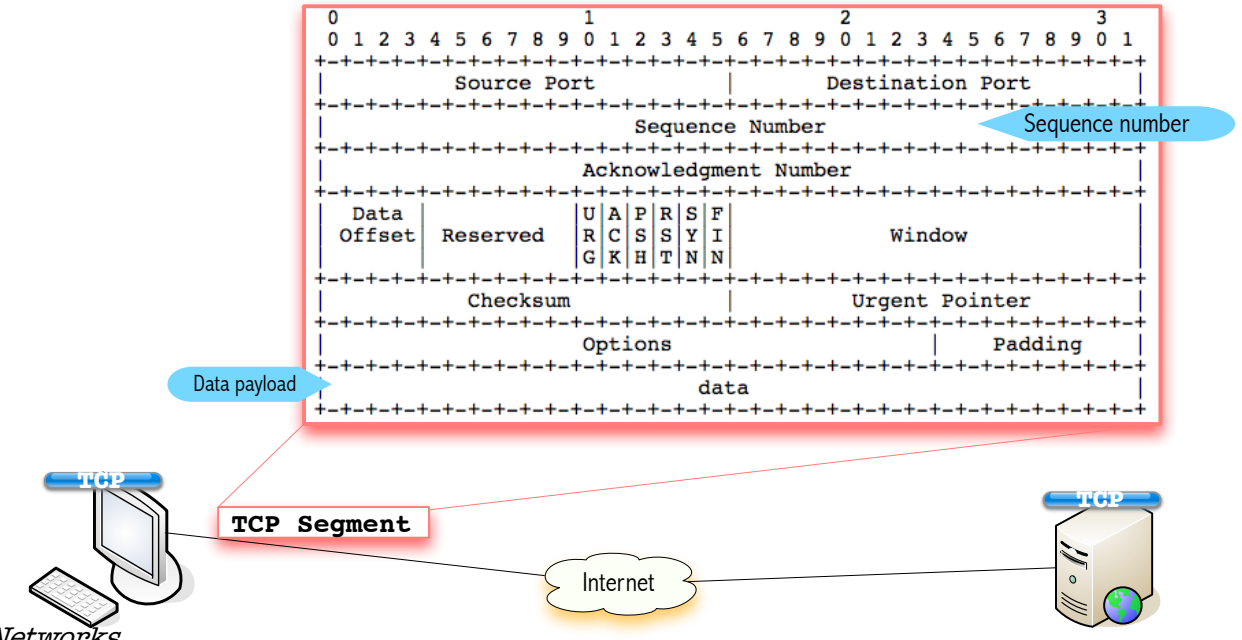
Essential dynamics of TCP

- When TCP receives a new segment, it soon will send an ACK for that segment



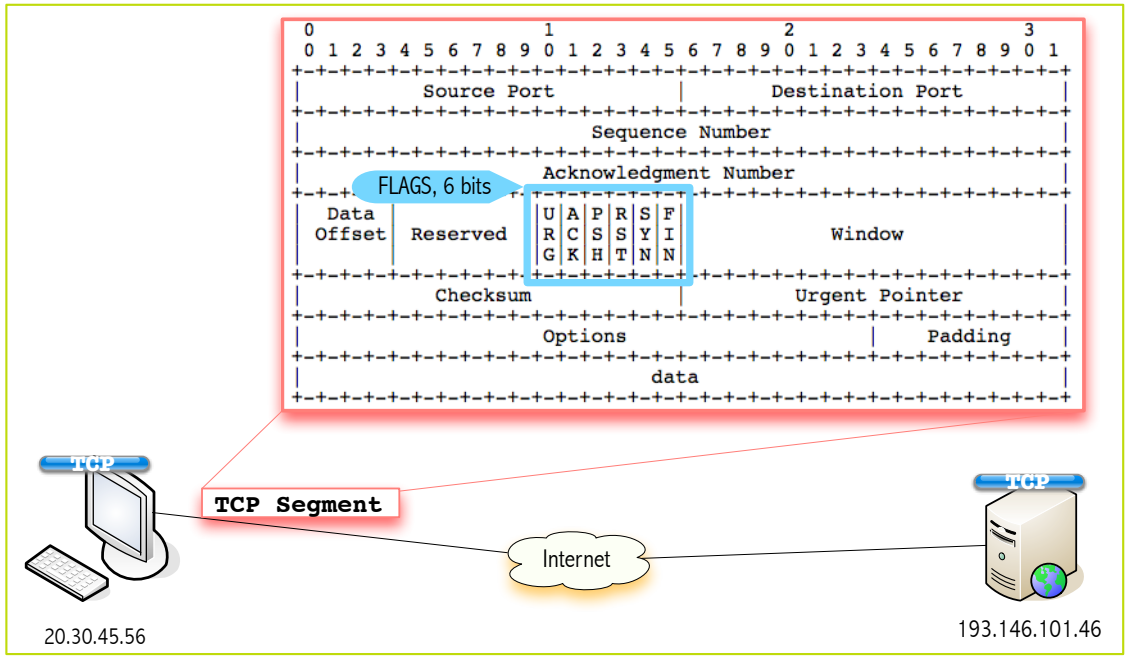
TCP Segment Header

- Sequence numbers
- Because TCP is a byte-oriented protocol, each byte of data has an *implicit* sequence number
- SequenceNum field contains the sequence number for the first byte of data carried in that segment.



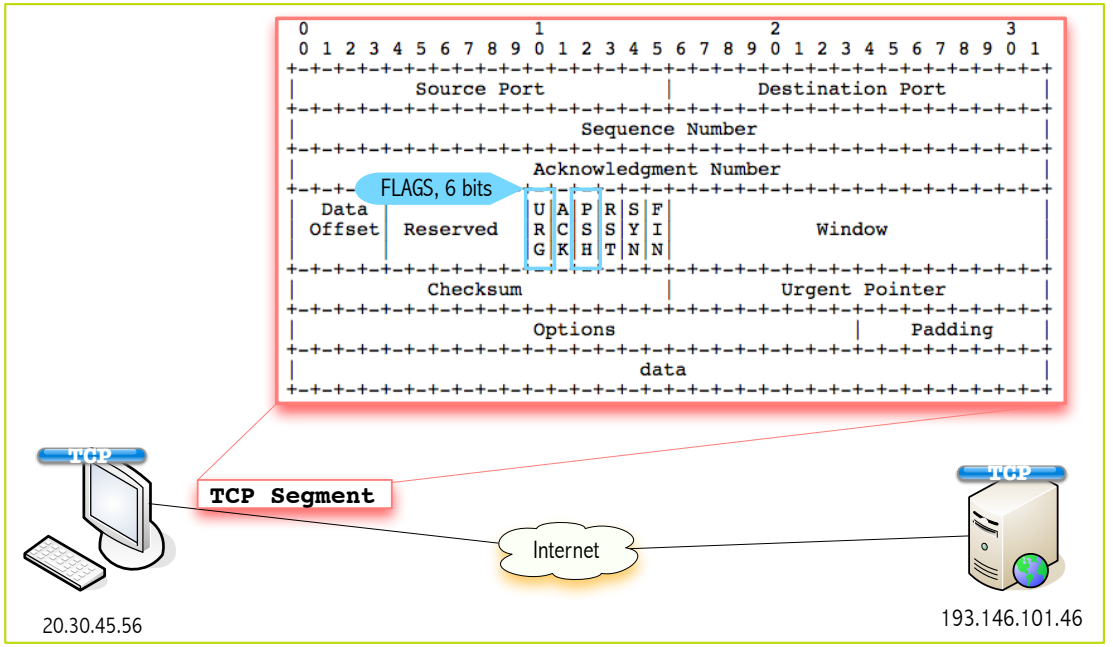
TCP Segment Header

- 6 bits of control flags
  - SYN, FIN, RESET, PUSH, URG, and ACK.
- SYN and FIN
  - Used for establishing and terminating a TCP connection
- ACK flag:
  - Set whenever the Acknowledgment number field is valid



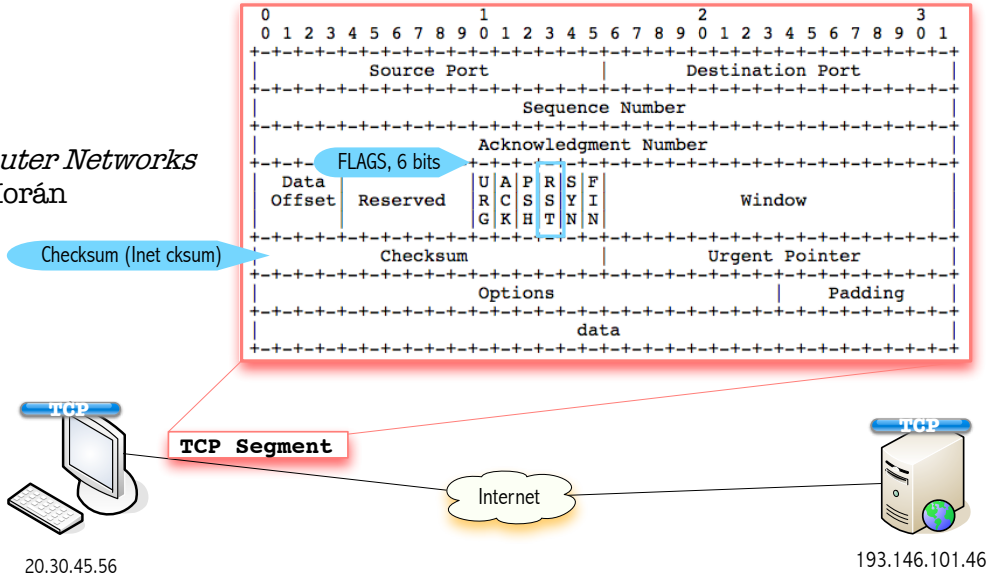
TCP Segment Header

- URG:
  - This segment contains urgent data
  - The urgent data is located at the beginning of the payload
  - UrgPtr points to the *nonurgent* data block
- PUSH:
  - Sender invoked the Socket API PUSH operation
  - The TCP receive side should notify the receiving thread of this fact



TCP Segment Header

- Finally, the RESET flag signifies that the receiver has become confused, it received a segment it did not expect to receive—and so wants to abort the connection.
- Finally, the Checksum field is used in exactly the same way as for UDP—it is computed over the TCP header, the TCP data, and the pseudoheader, which is made up of the source address, destination address, and length fields from the IP header.

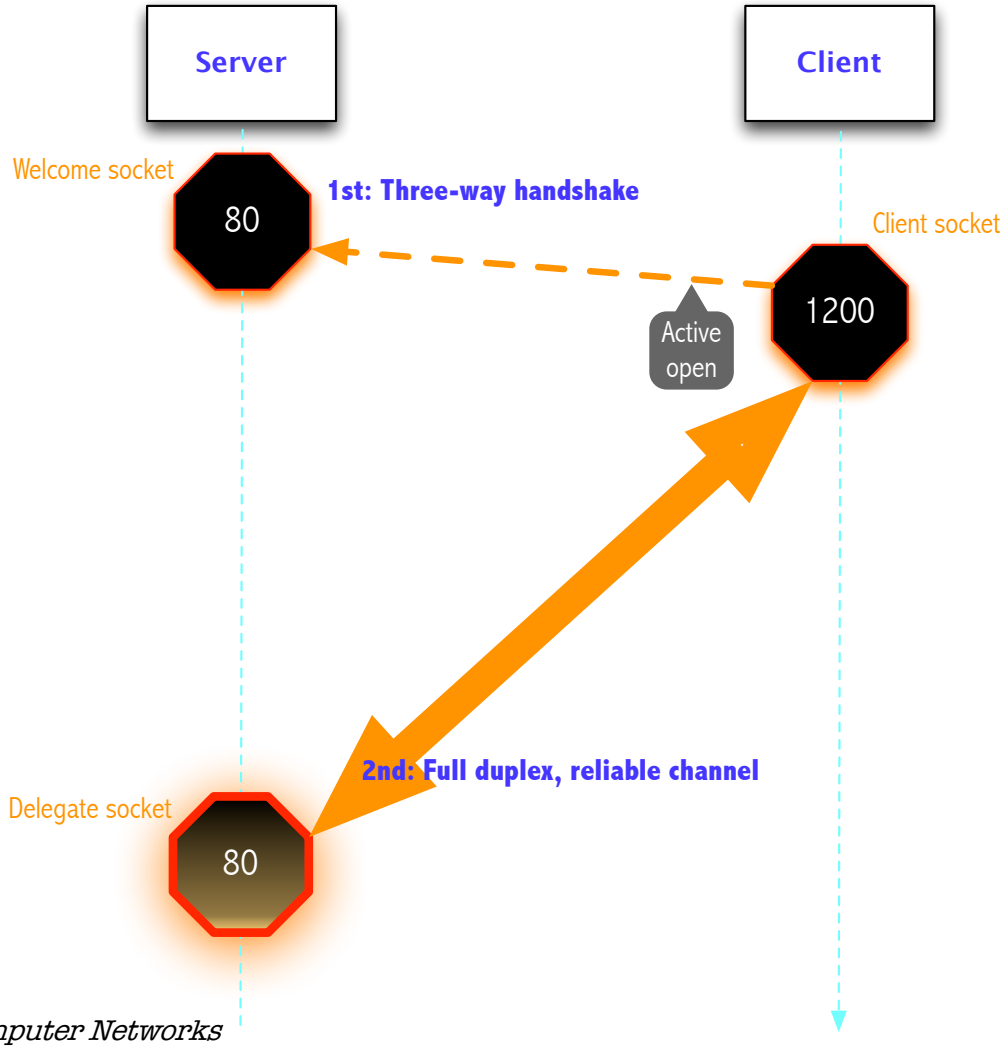


TCP

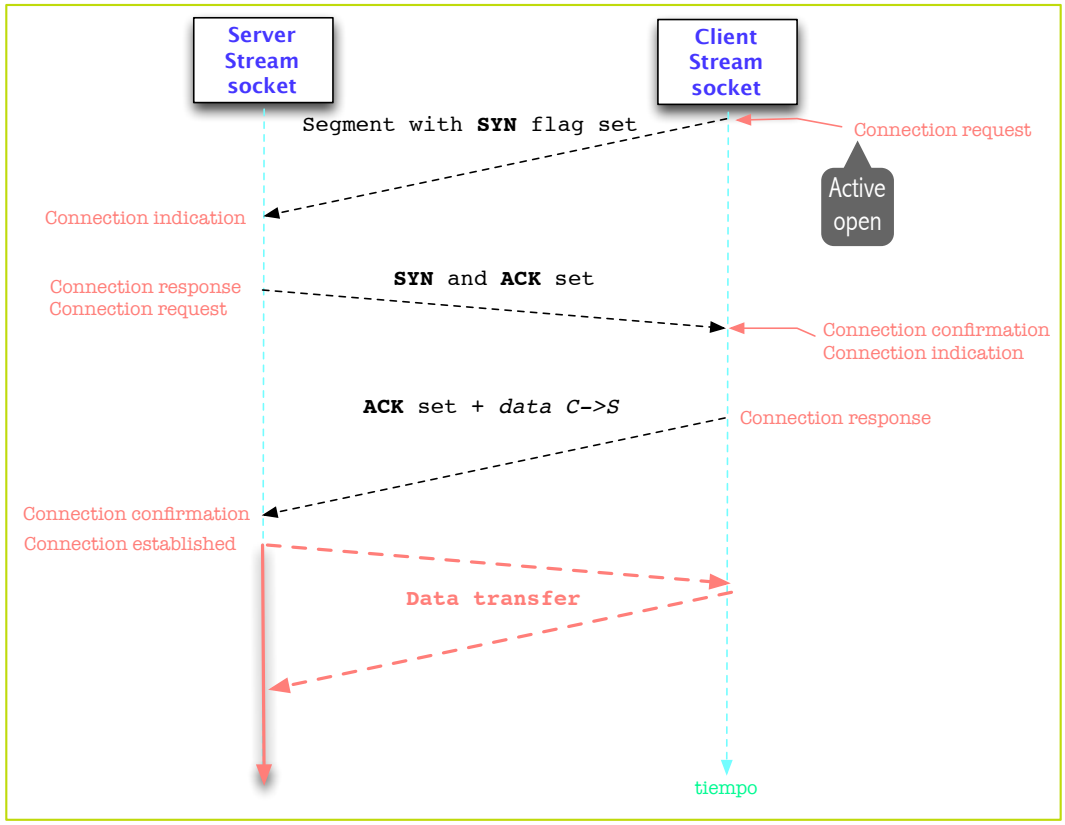
Connection establishment and teardown

Based on textbook Conceptual Computer Networks

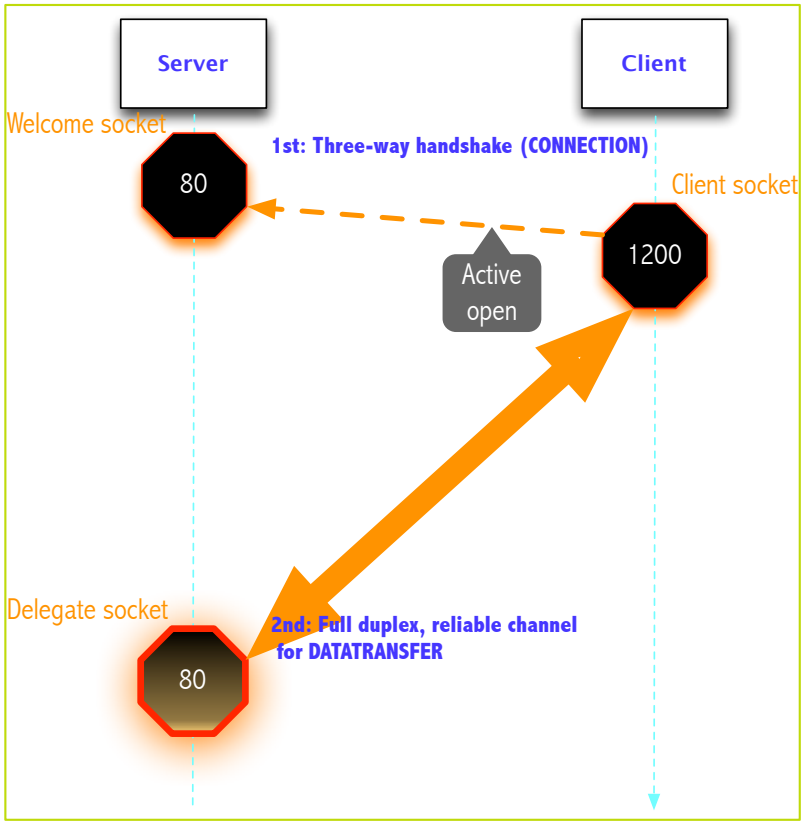
Connection Establishment/Termination in TCP



Connection Establishment/Termination in TCP

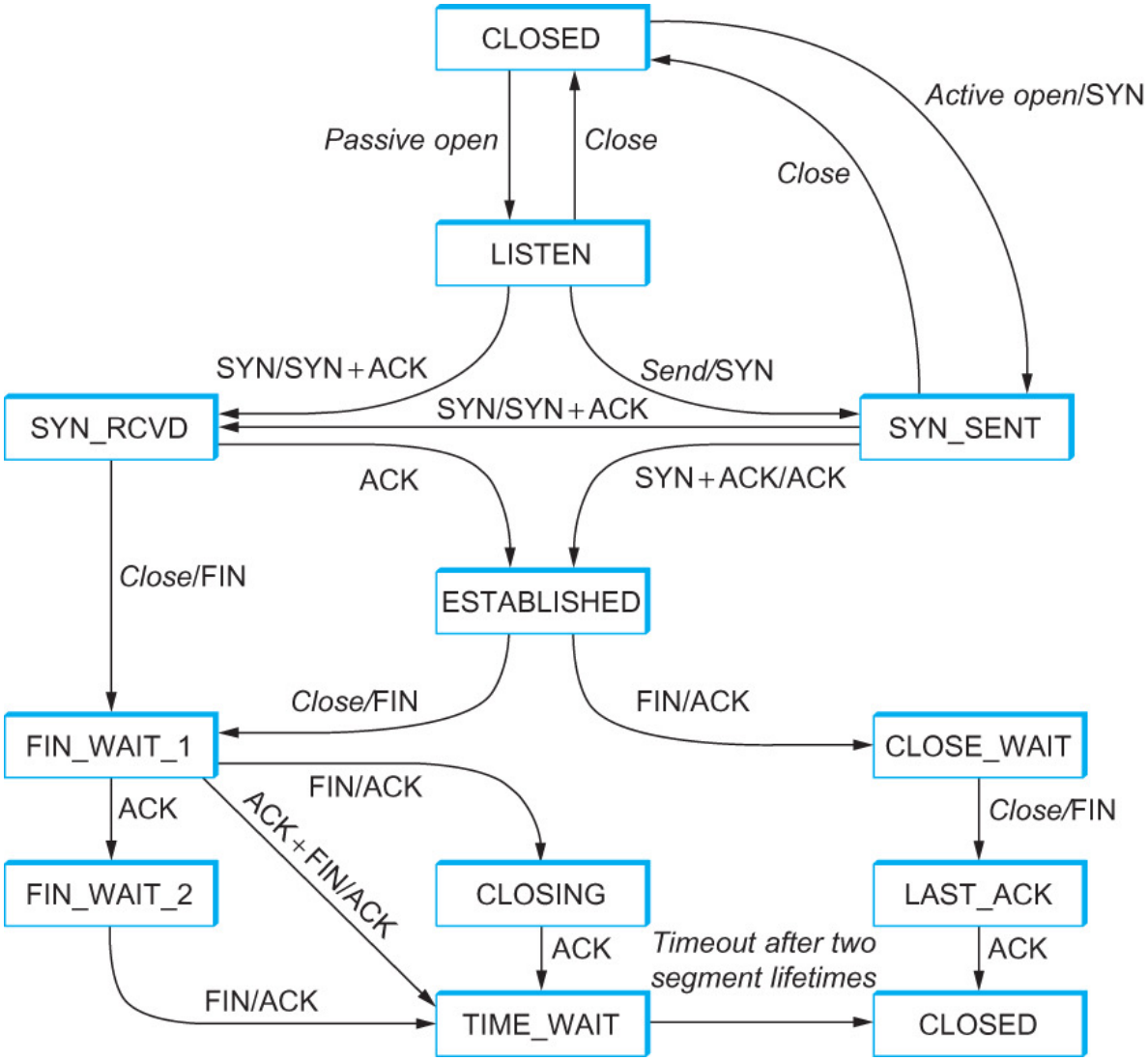


Peer-to-peer interface



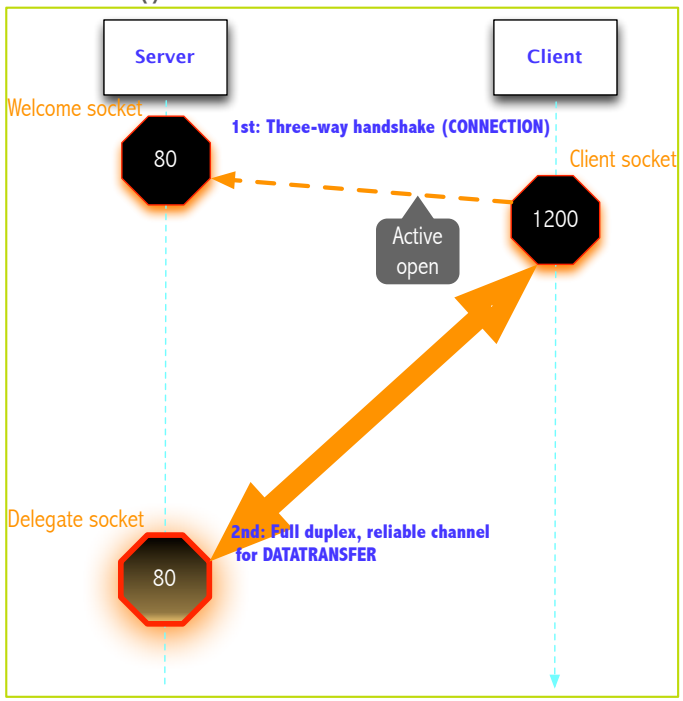
Sockets API/Service interface

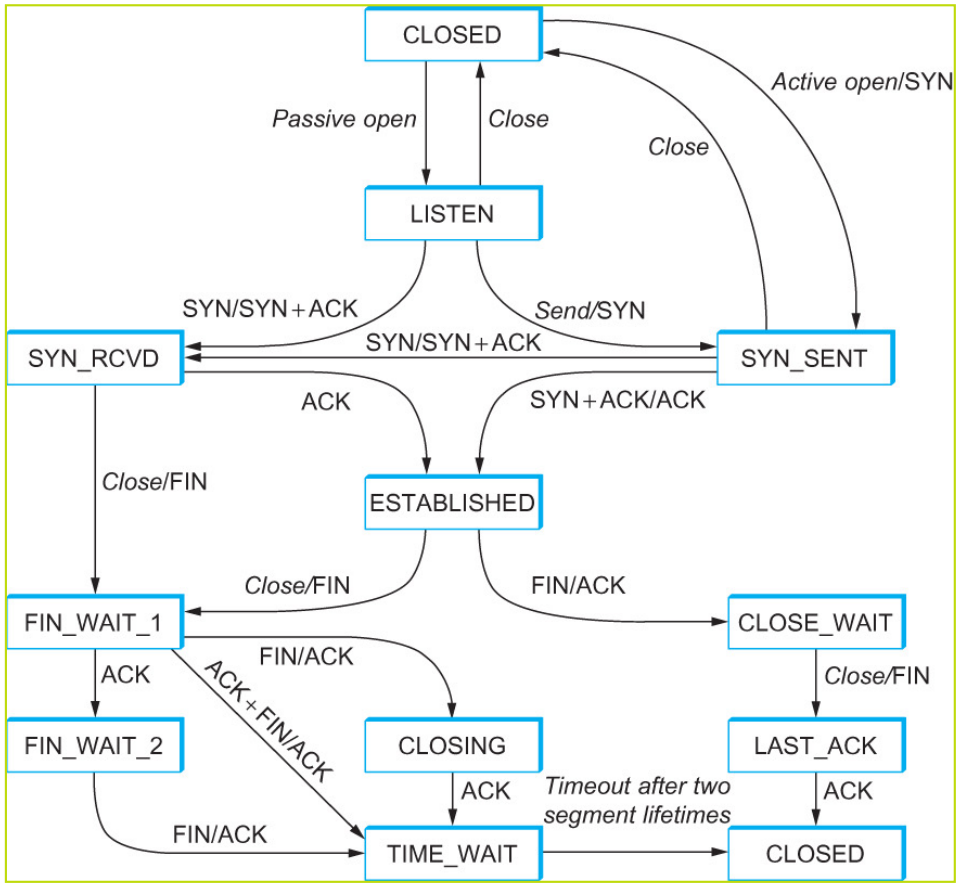
State transitions in TCP



Sockets API

- Passive Open (Server)
  - Welcome Socket is a Java ServerSocket instance
  - In C it is the result of socket()
    - listen()/accept()
- Active Open (Client)
  - Java Socket
  - C socket()





I used the following references in the composition of the present work (In order of importance):

1. Peterson and Davie, Computer Networks, MKP Elsevier 2012
2. TCP/IP Illustrated Vol. 1 R. Stevens Addison-Wesley
3. Relevant RFC's