Lecture 13: Transport Layer Flow and Congestion Control

COMP 332, Fall 2018

Victoria Manfredi

WESLEYAN UNIVERSITY

Acknowledgements: materials adapted from Computer Networking: A Top Down Approach 7th edition: ©1996-2016, J.F Kurose and K.W. Ross, All Rights Reserved as well as from slides by Abraham Matta at Boston University, and some material from Computer Networks by Tannenbaum and Wetherall.

Today

1. Announcements
   - exam wed!
2. Midterm overview
   - exam format
3. TCP
   - reliable data transfer
   - connection management
   - flow control

Midterm

OVERVIEW

Midterm overview

In class on Wednesday, Oct. 17

- closed book, closed notes
- covers material in lectures 1 to 12

Will not ask questions on

- probability

5 questions

- app layer short questions
- HTTP persistent vs. non-persistent connections
- transport layer short questions
- socket coding
- reliable data transport protocol

Problems 1 and 3

App layer and transport layer short questions

- 8 in total
- similar to review questions in book
- should only need to write a few sentences to answer

Problem 2

HTTP persistent vs. non-persistent connections

- review related homework question

Problems 4

Socket coding

- be able to write code to open, use, and close sockets
- differences between client and server code

Problem 5

Design a reliable data transfer protocol

- given channel characteristics design most efficient protocol
- be able to design reliable data transfer protocol like stop-and-wait
- know your timeline diagrams

TCP

RELIABLE DATA TRANSFER

Duplicate ACKs

Time-out period often relatively long

- long delay before resending lost packet

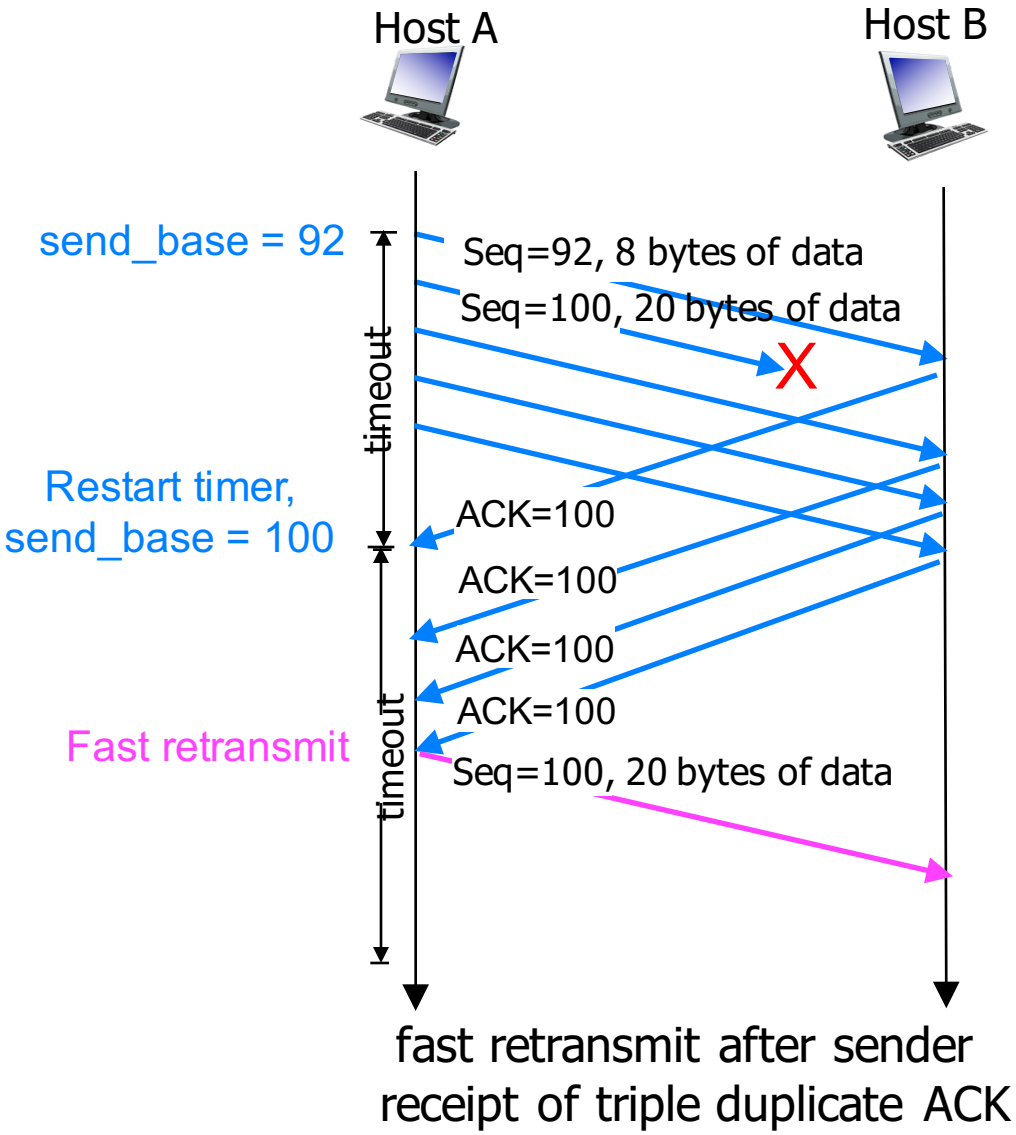
Duplicate ACKs indicate isolated loss

- rather than congestion causing many losses
  - sender often sends many segments back-to-back
  - if segment is lost, likely many duplicate ACKs
  - ACKs being received indicates some packets received at destination since ACK sent for every packet: so not congestion

TCP fast retransmit

- if sender receives 3 ACKs for same data (triple duplicate ACKs)
  - resend unacked segment with smallest seq #
- Q: why 3?
  - pkts may just have been reordered otherwise
  - likely that unacked segment lost, so don't wait for timeout

TCP fast retransmit



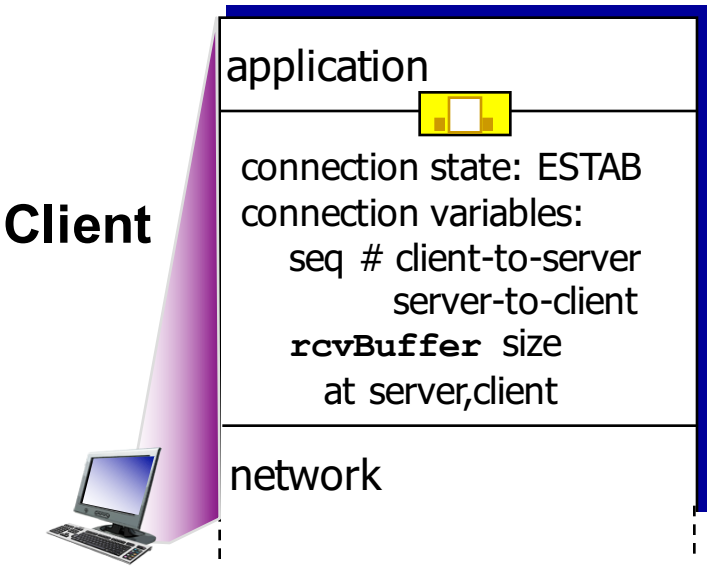
TCP

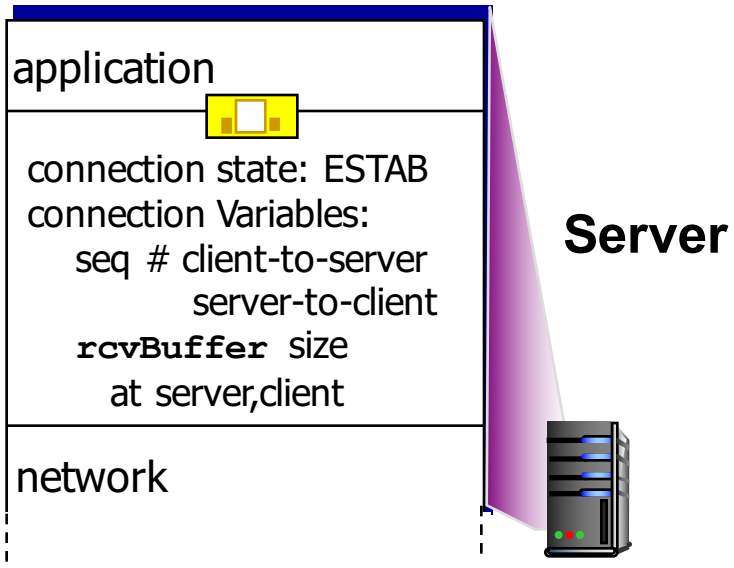
CONNECTION MANAGEMENT

Connection Management

Before exchanging data, sender/receiver handshake

- establish connection and connection parameters
- tear down connection when done



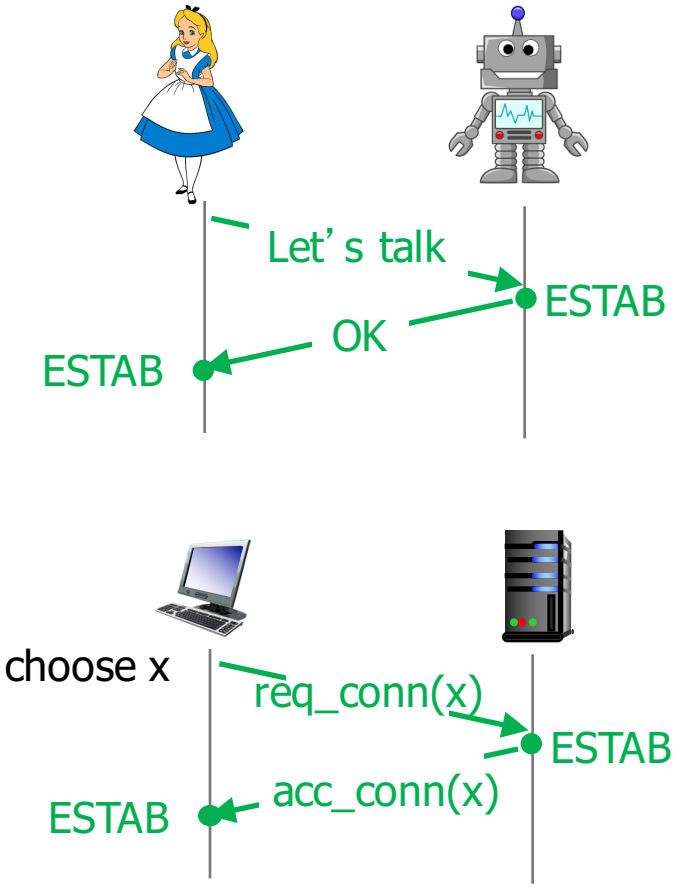


sock = sock.connect((host, port))

conn, addr = server_sock.accept()

Agreeing to establish a connection

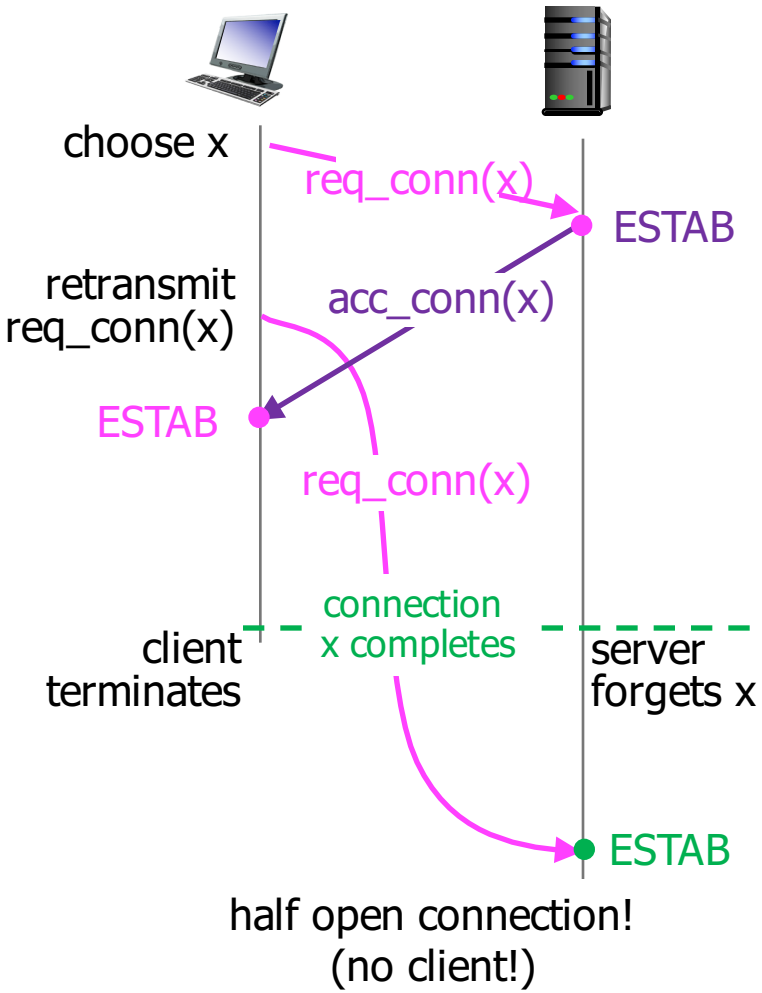
2-way handshake:

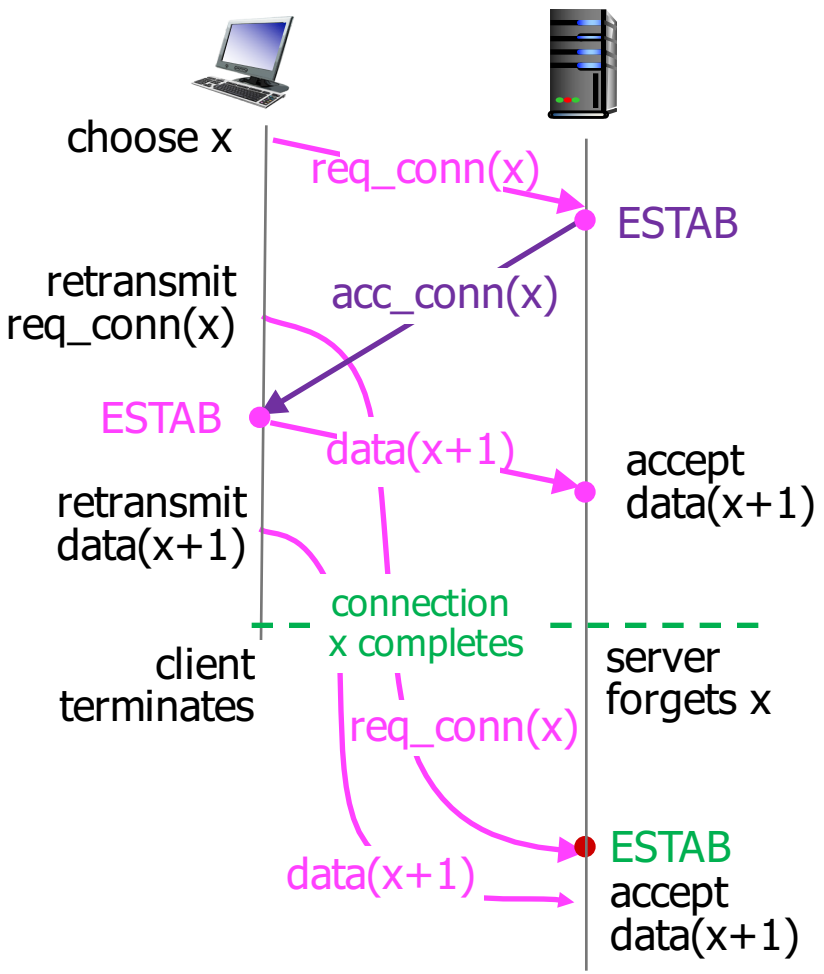


Q: will 2-way handshake always work in network?

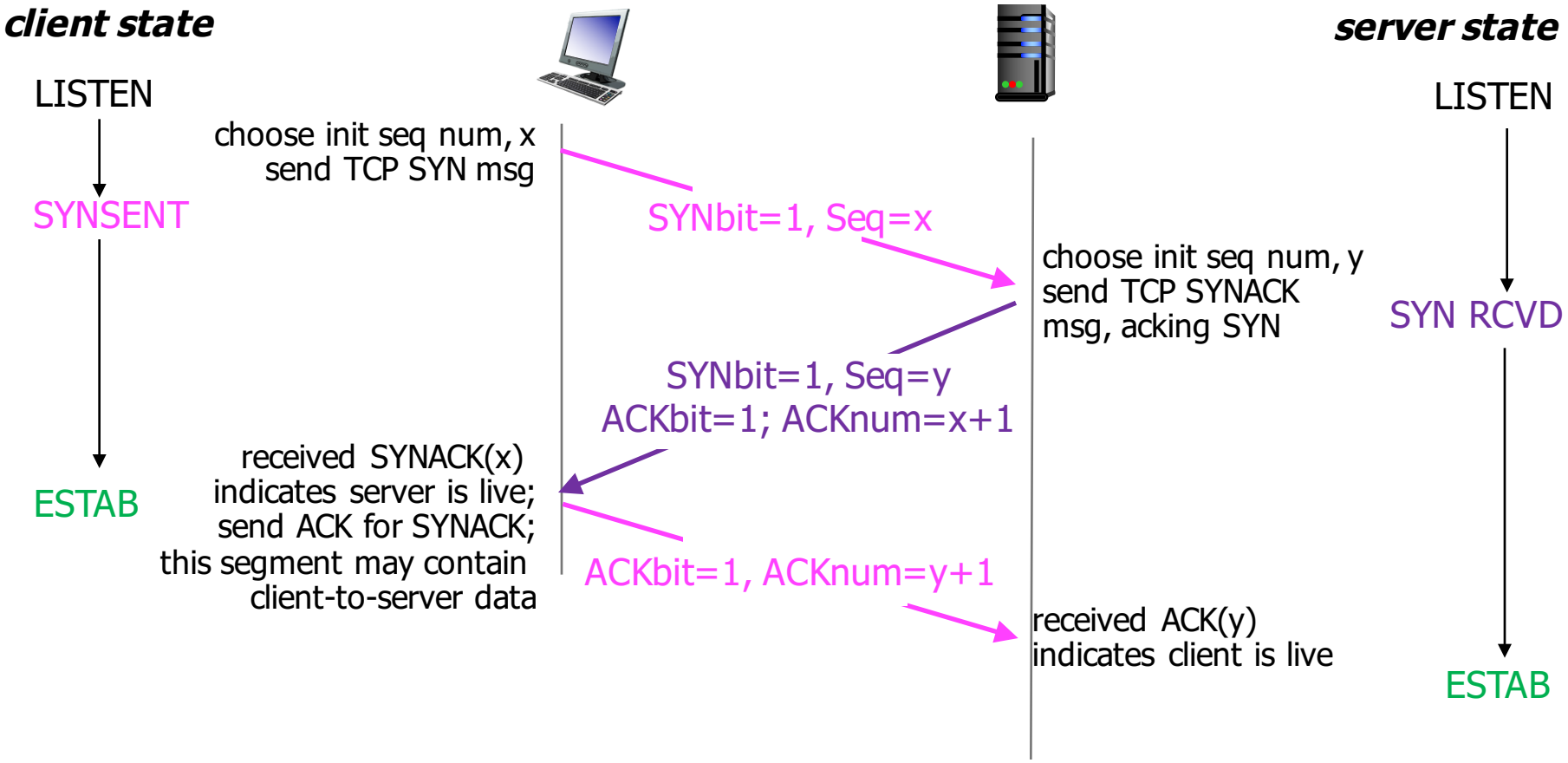
- variable delays
- retransmitted messages
  - e.g. req_conn(x)) due to message loss
- message reordering
- can't see other side

2-way handshake failure scenarios

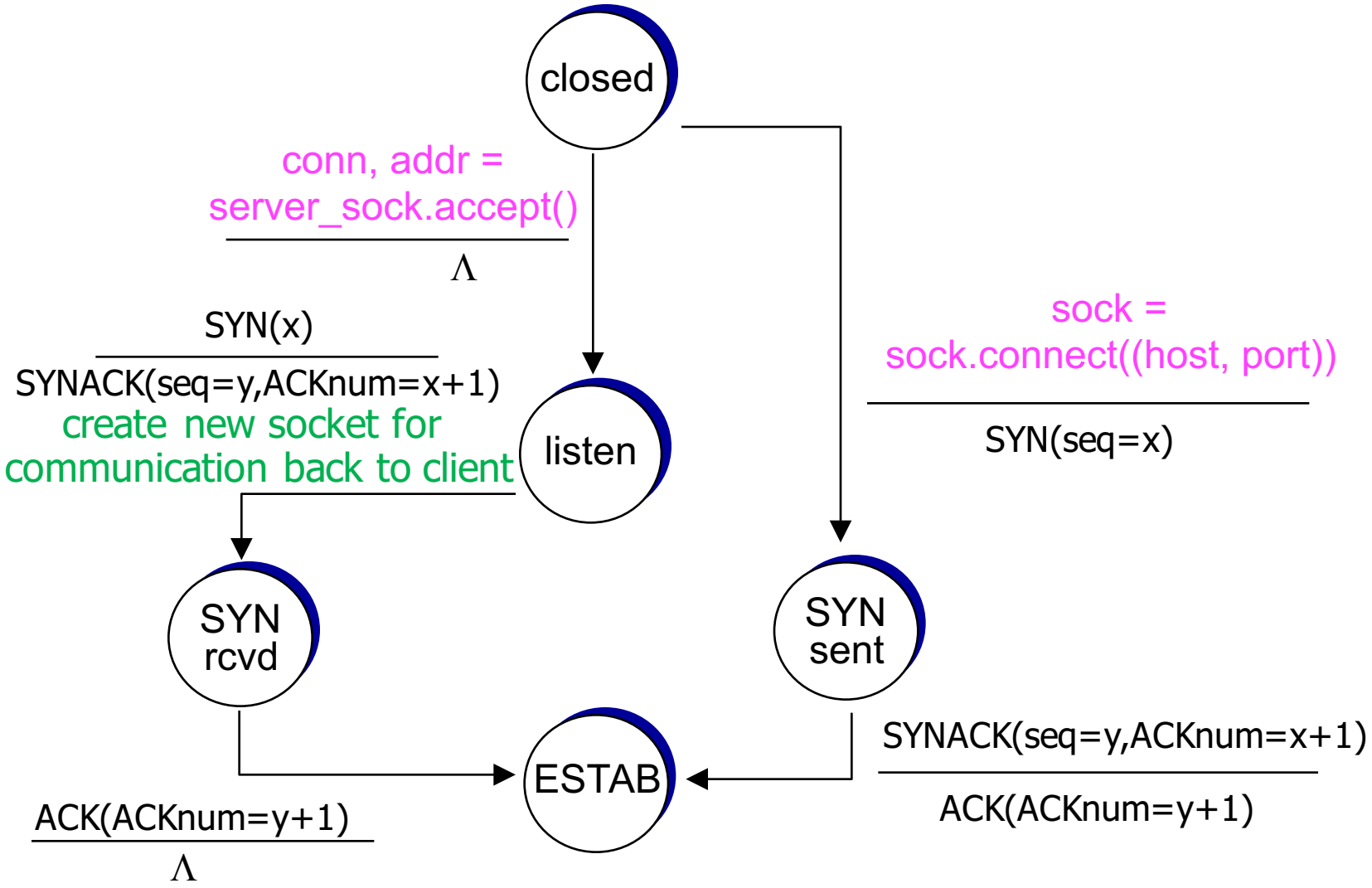




TCP 3-way handshake



TCP 3-way handshake: FSM



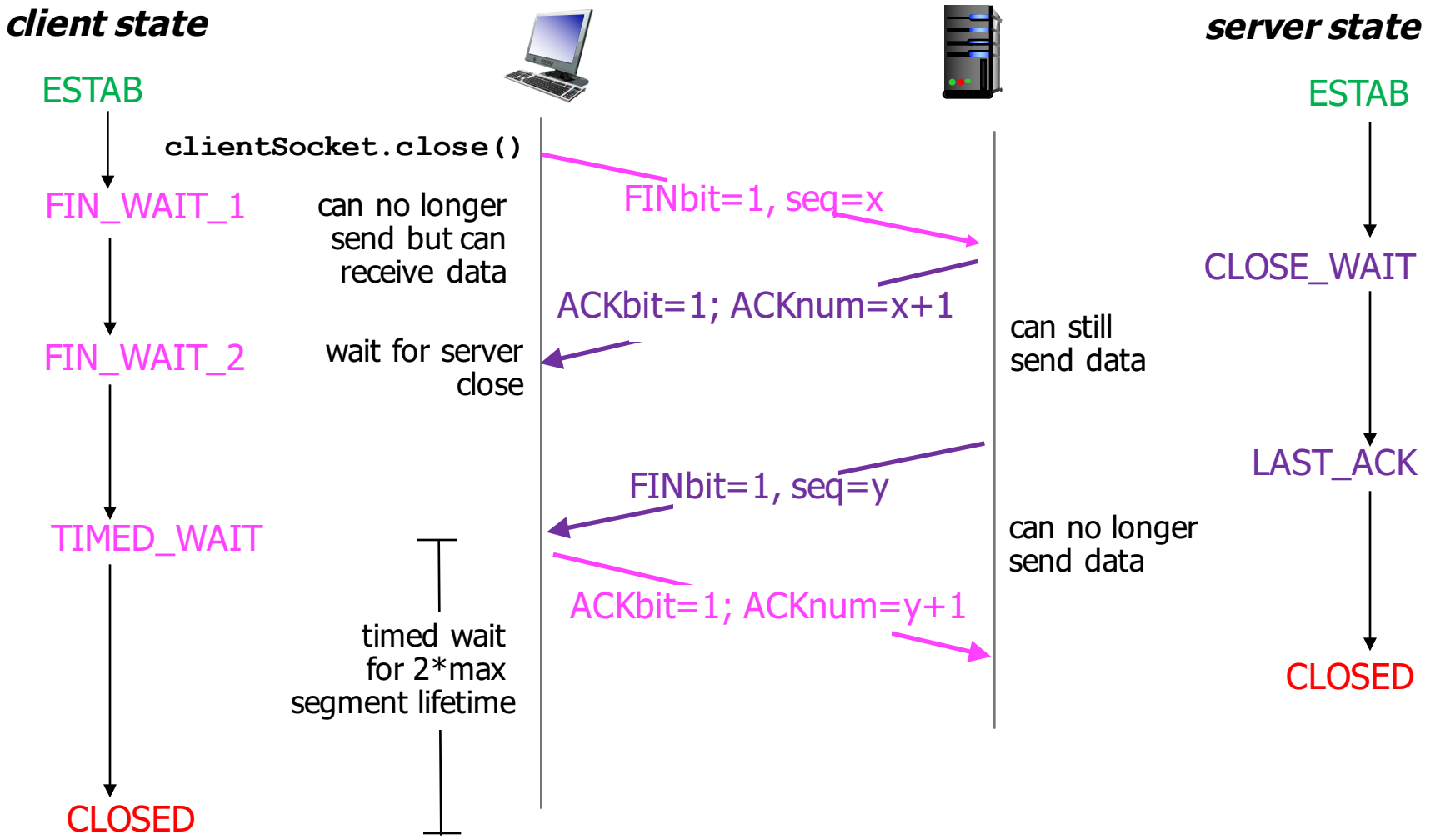
Look at the state of tcp connections

```
> netstat -ta
Active Internet connections (including servers)
Proto Recv-Q Send-Q  Local Address          Foreign Address        (state)
tcp4       0      0  vmanfredismbp2.w.55777 lga25s60-in-f5.1.https ESTABLISHED
tcp4      31      0  vmanfredismbp2.w.55736 162.125.34.6.https     CLOSE_WAIT
tcp4       0      0  vmanfredismbp2.w.55717 a104-110-151-148.https ESTABLISHED
tcp4       0      0  vmanfredismbp2.w.55716 a104-110-151-148.https ESTABLISHED
tcp4       0      0  vmanfredismbp2.w.55715 a104-110-151-148.https ESTABLISHED
tcp4       0      0  vmanfredismbp2.w.55714 a104-110-151-148.https ESTABLISHED
tcp4       0      0  vmanfredismbp2.w.55713 a104-110-151-148.https ESTABLISHED
tcp4       0      0  vmanfredismbp2.w.55668 wesfiles.wesleya.http  CLOSE_WAIT
tcp4       0      0  vmanfredismbp2.w.55486 162.125.18.133.https   ESTABLISHED
tcp4       0      0  vmanfredismbp2.w.55322 162.125.18.133.https   ESTABLISHED
tcp4      31      0  vmanfredismbp2.w.55250 162.125.4.3.https      CLOSE_WAIT
tcp4       0      0  vmanfredismbp2.w.55170 ec2-52-20-75-192.https CLOSE_WAIT
tcp4       0      0  vmanfredismbp2.w.55072 85.97.201.35.bc..https ESTABLISHED
tcp4       0      0  localhost.ipp          *.*                    LISTEN
tcp6       0      0  localhost.ipp          *.*                    LISTEN
tcp4       0      0  vmanfredismbp2.w.53453 6.97.a86c.ip4.st.https ESTABLISHED
```

TCP: politely closing a connection

Client, server each sends TCP segment with FIN bit = 1

- respond to received FIN with ACK (ACK can be combined with own FIN)



FIN segment in Wireshark