Networking

Slides by Andrew DeOrio

Agenda

- Motivation
- IP: Internet Protocol
- TCP: Transmission Control Protocol
  - Flow control and congestion control
- UDP: User Datagram Protocol
- Sockets
- Summary

Distributed systems

- Distributed system: Multiple computers cooperating on a task

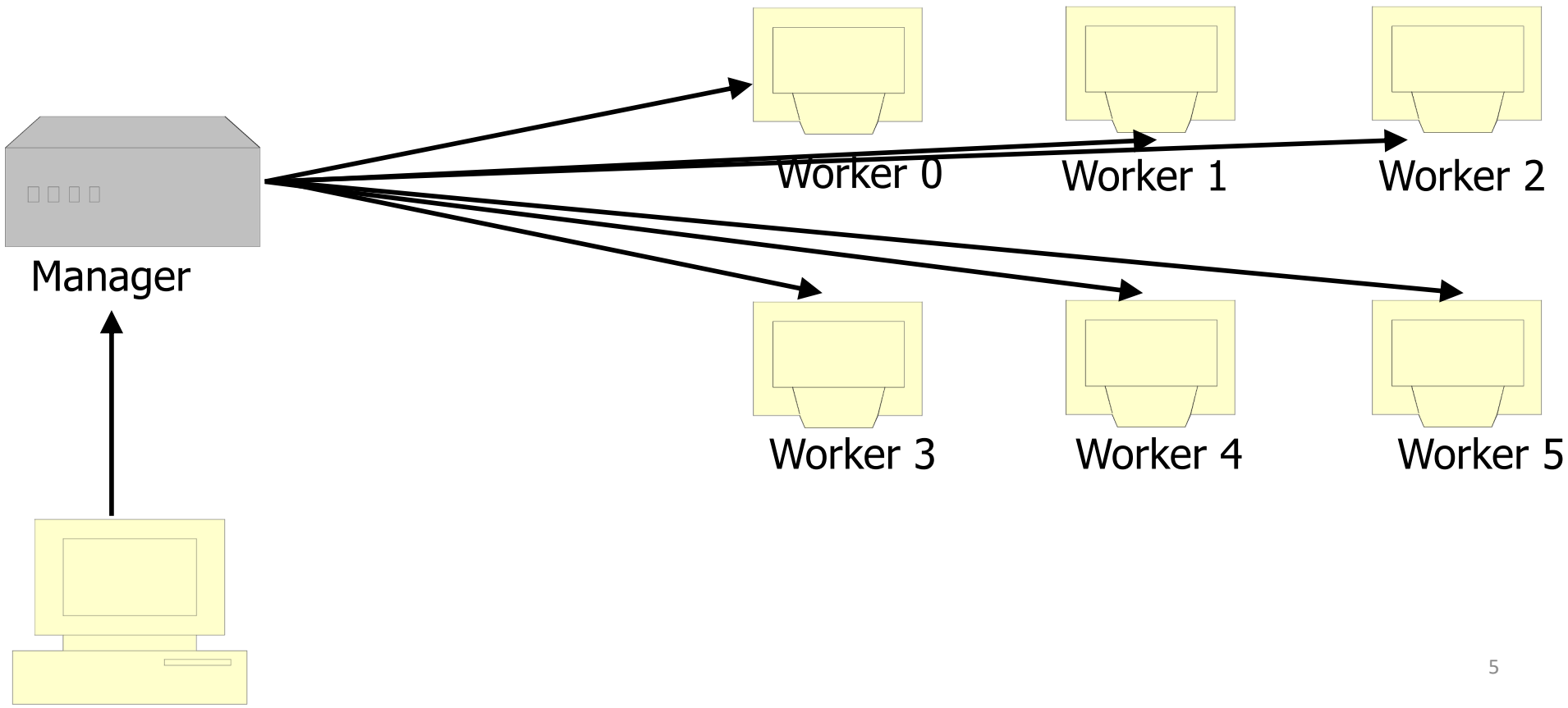
- MapReduce: Distributed system for compute
  - Run a program that would be too slow on one computer
- Google File System: Distributed system for storage
  - Store more data than fits on one computer

Distributed system implementation

- How are distributed systems implemented?
- Threads and processes for parallelization
  - Last time
- Networking for communication
  - Today

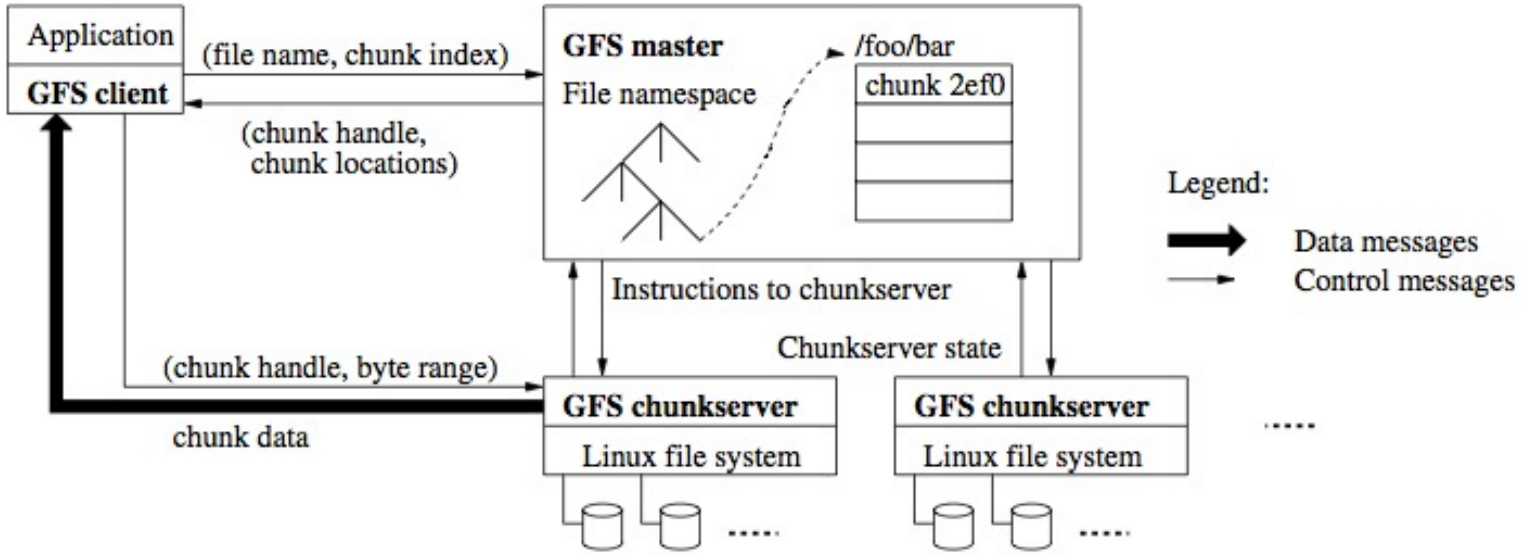
Networking in a MapReduce framework

- How does the MapReduce Manager send a new task to a Worker?
  - Send a message over the network
  - E.g., JSON blob over TCP/IP



Networking in Google File System

- How do chunkservers send a heartbeat message to the Main server to tell it "I'm alive"?
  - Send a message over the network
  - JSON blob over UDP in Project 4 heartbeat implementation



Agenda

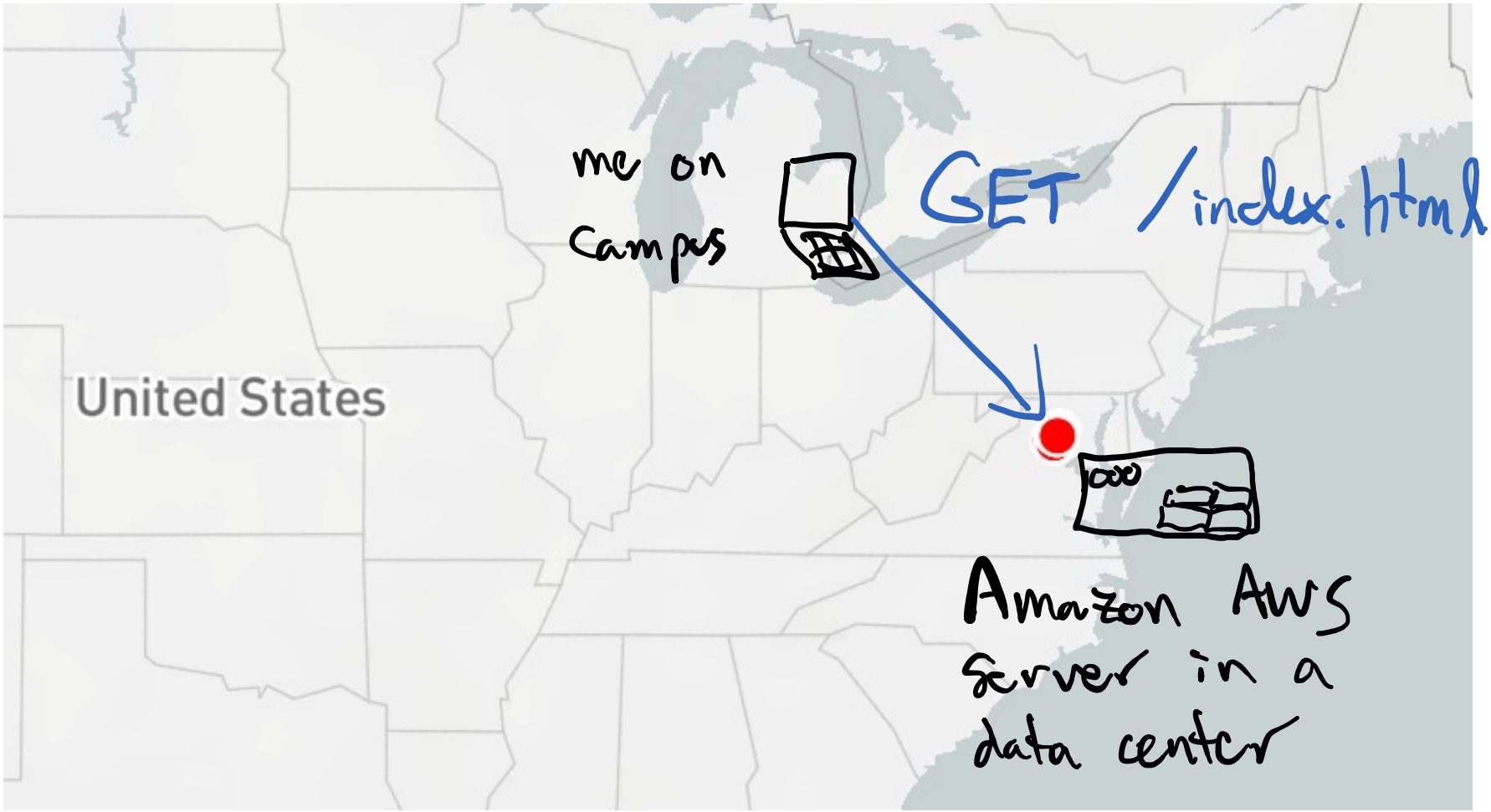
- Motivation
- **IP: Internet Protocol**
- TCP: Transmission Control Protocol
  - Flow control and congestion control
- UDP: User Datagram Protocol
- Sockets
- Summary

IP: Internet Protocol

- How to get a message from one computer to another computer?
- Solution: Internet Protocol (IP)

- Examples of messages
  - MapReduce Manager server assigns task to Worker
  - Google File System chunkserver tells Main server "I'm alive"
  - GET request from a client web browser to an HTTP server

Example message from client to server



IP Address

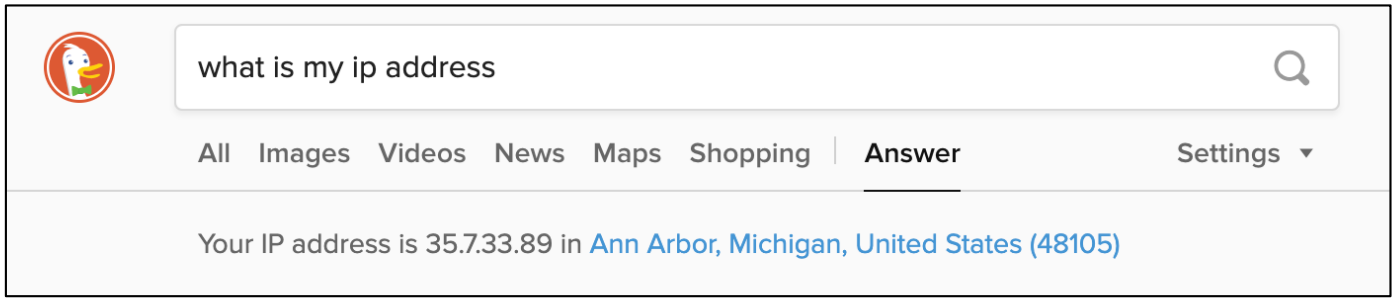
- There are lots of computers connected to the internet
- Problem: How to tell them apart?
- Solution: IP Address

- Every computer on the Internet has an address
  - Google 172.217.5.14
  - Amazon 205.251.242.103
  - My laptop 35.7.33.89

- Newer version: longer addresses
  - IPv4: 32 bits is 4 billion computers
  - IPv6: 64 bits is way more

What is my IP address?

- Your internet service provider gives you an IP address when you connect to the wifi (or wired) network
- Popular search engines provide your externally visible IP address
  - If you're on a home network, this is likely the IP address of your router



- At the command line

```
$ curl ipinfo.io/ip
35.7.33.89
```

What is Amazon's IP address?

- How do you turn aws.amazon.com into an IP address?
- Domain Name Service (DNS)
  - Covered later this semester

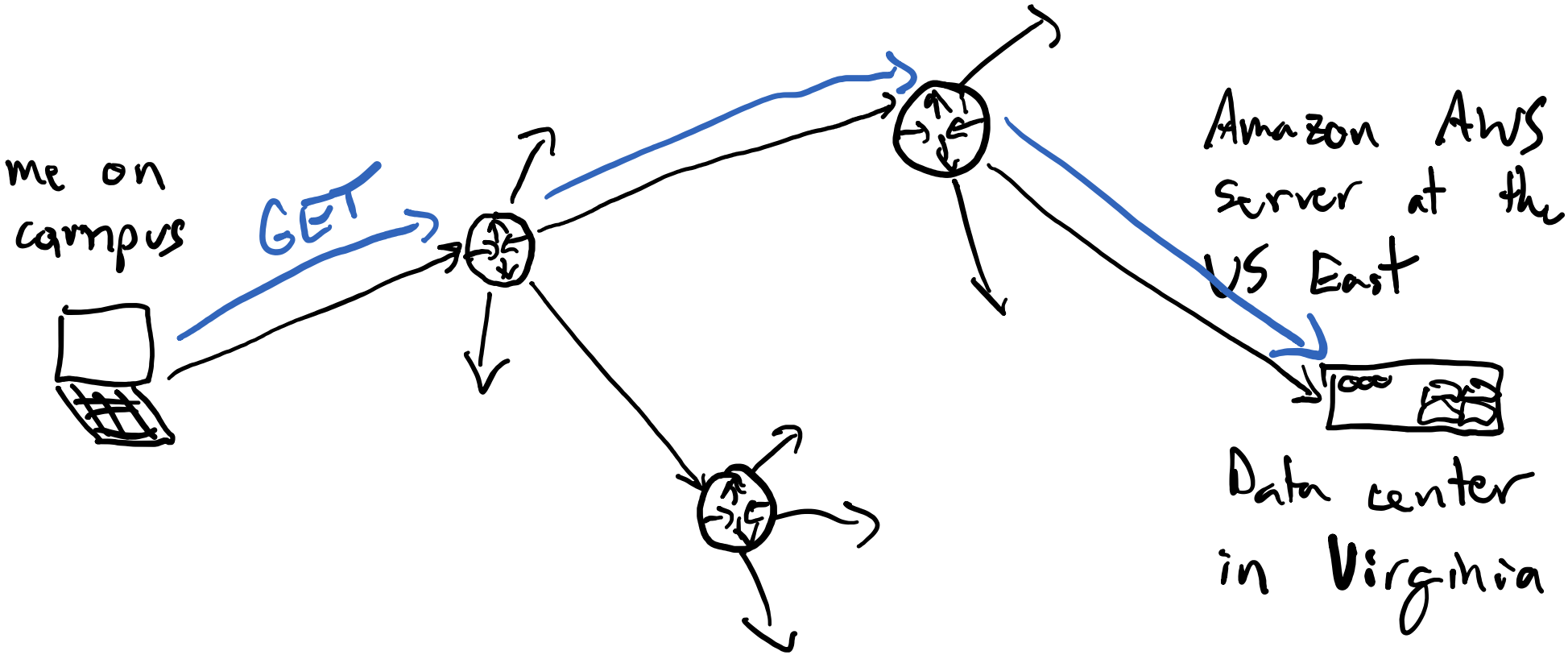
Routers

- How do we connect all the computers on the internet?
- Routers

- A *router* forwards data from one network to the next

- Usually a purpose-built Linux computer with multiple network connections

Routers


me on campus
GET
Amazon AWS server at the US East
Data center in Virginia
routers

Routing

- Problem: A router is connected many other routers. Where to send data?
- Solution: One step closer to the destination
  - AKA "longest prefix matching"

- Each router has a *routing table*
  - Map groups of IP address to destination
  - Destination could be another router

- What new data arrives, look up the first few bits of the destination IP address in the routing table

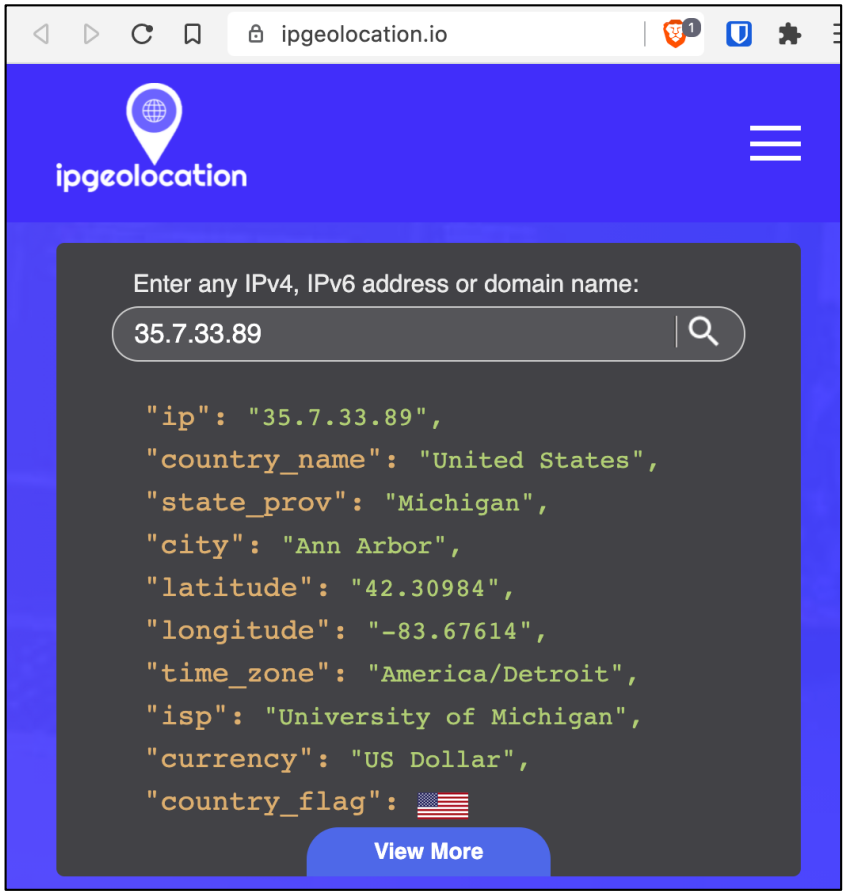
Tracing a route

- See the route from your computer to aws.amazon.com
  - Each step is called a *hop*

```
$ traceroute aws.amazon.com
 1  l3-umvpn2-seb.r-seb.umnet.umich.edu (141.213.172.1)
 2  l3-binseb-seb-ipsvrf.r-bin-seb.umnet.umich.edu (192.12.80.138)
 3  et-1-0-0.2061.rtsw.star.net.internet2.edu (198.71.45.249)
 4  ae-0.4079.rtsw.eqch.net.internet2.edu (162.252.70.110)
 5  99.82.179.50 (99.82.179.50)
 ...
22  server-54-239-175-72.ord53.r.cloudfront.net (54.239.175.72)
```

Internet geolocation

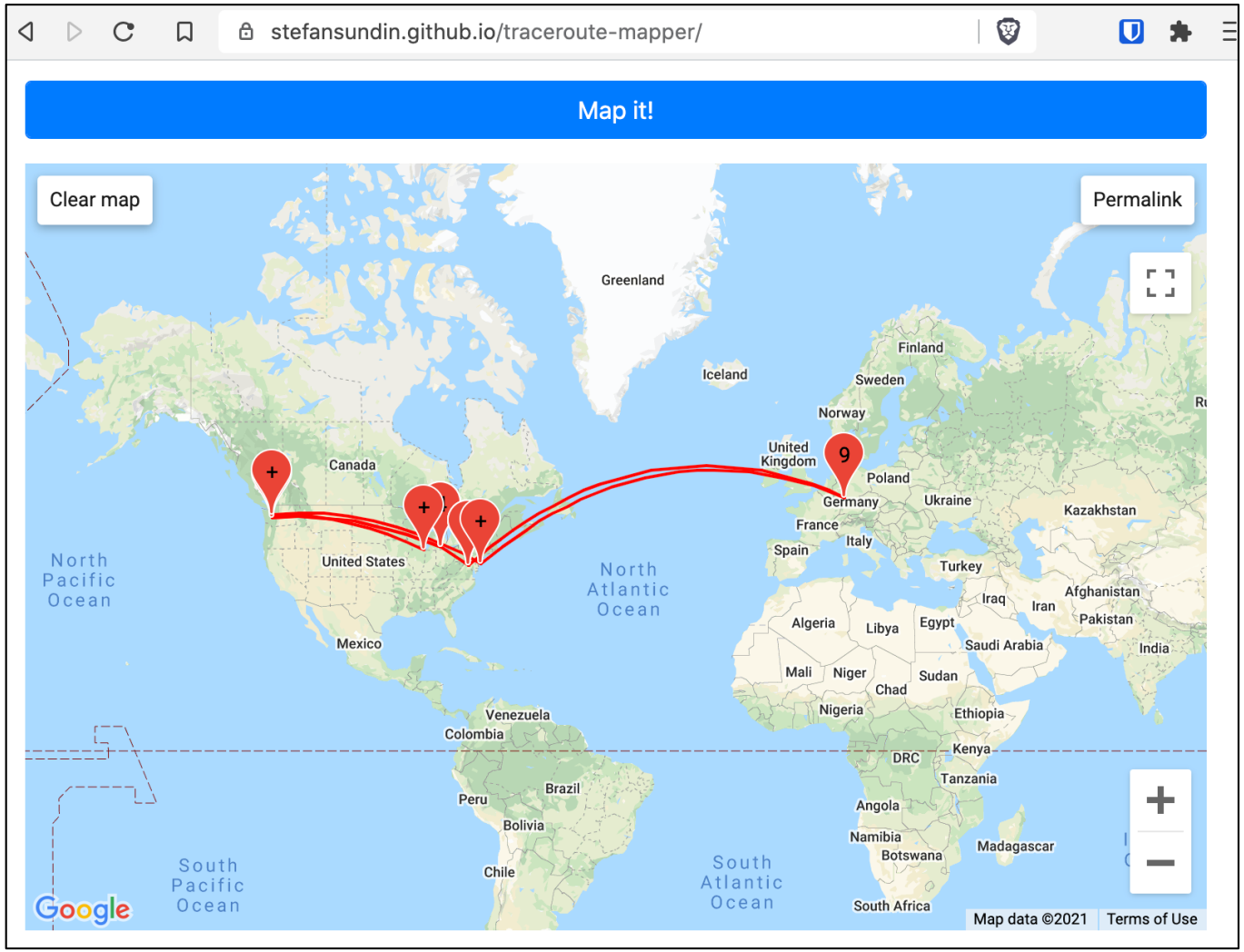
- *Internet geolocation* tells us where a device is located
- Database maps IP address to approximate location



- At the command line
  - Get API_KEY from ipgeolocation.io dashboard

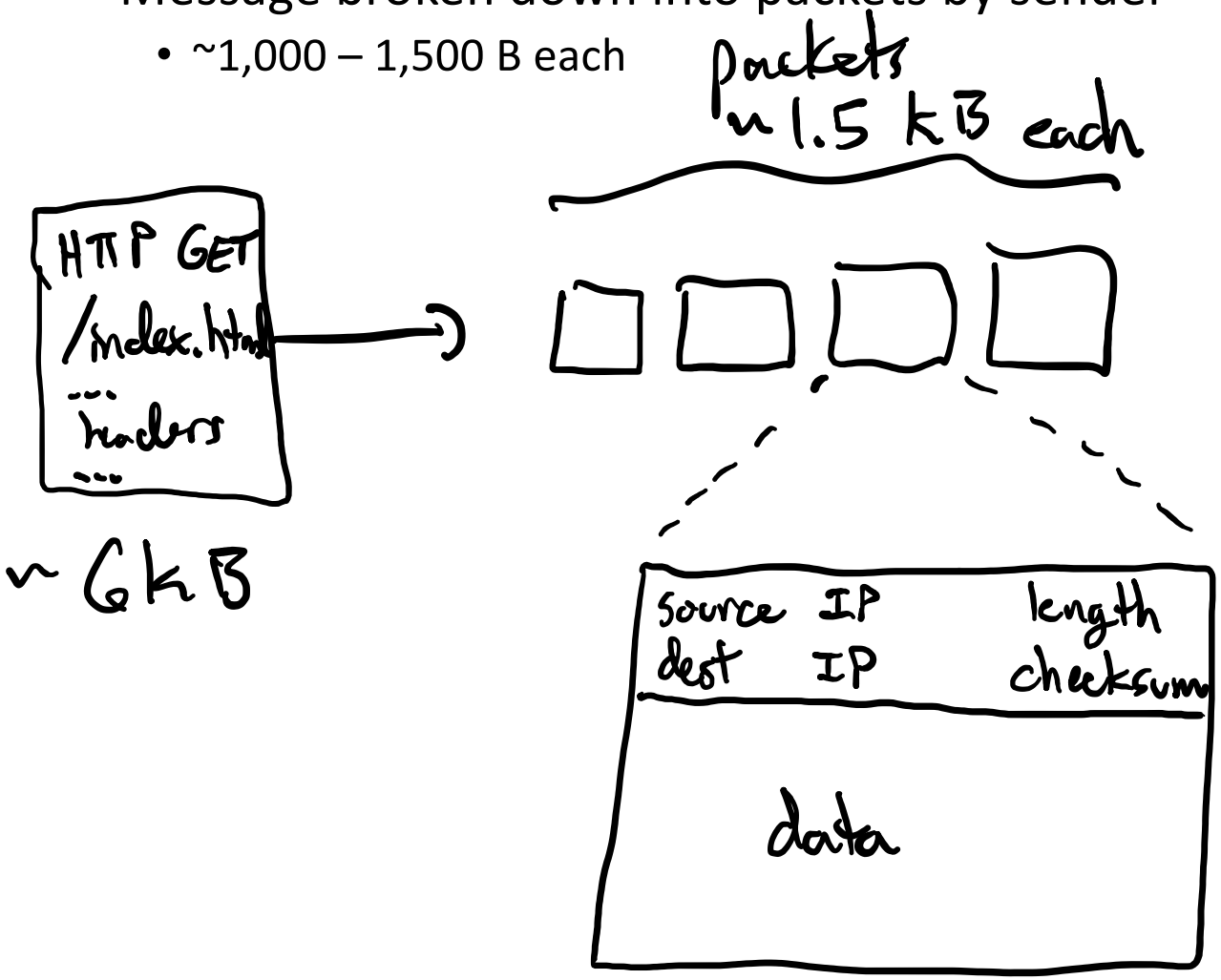
```
$ curl 'https://api.ipgeolocation.io/ipgeo?apiKey=API_KEY'
{
  "ip": "35.7.33.89",
  "city": "Ann Arbor",
  "isp": "University of Michigan",
  ...
}
```

Traceroute + geolocation



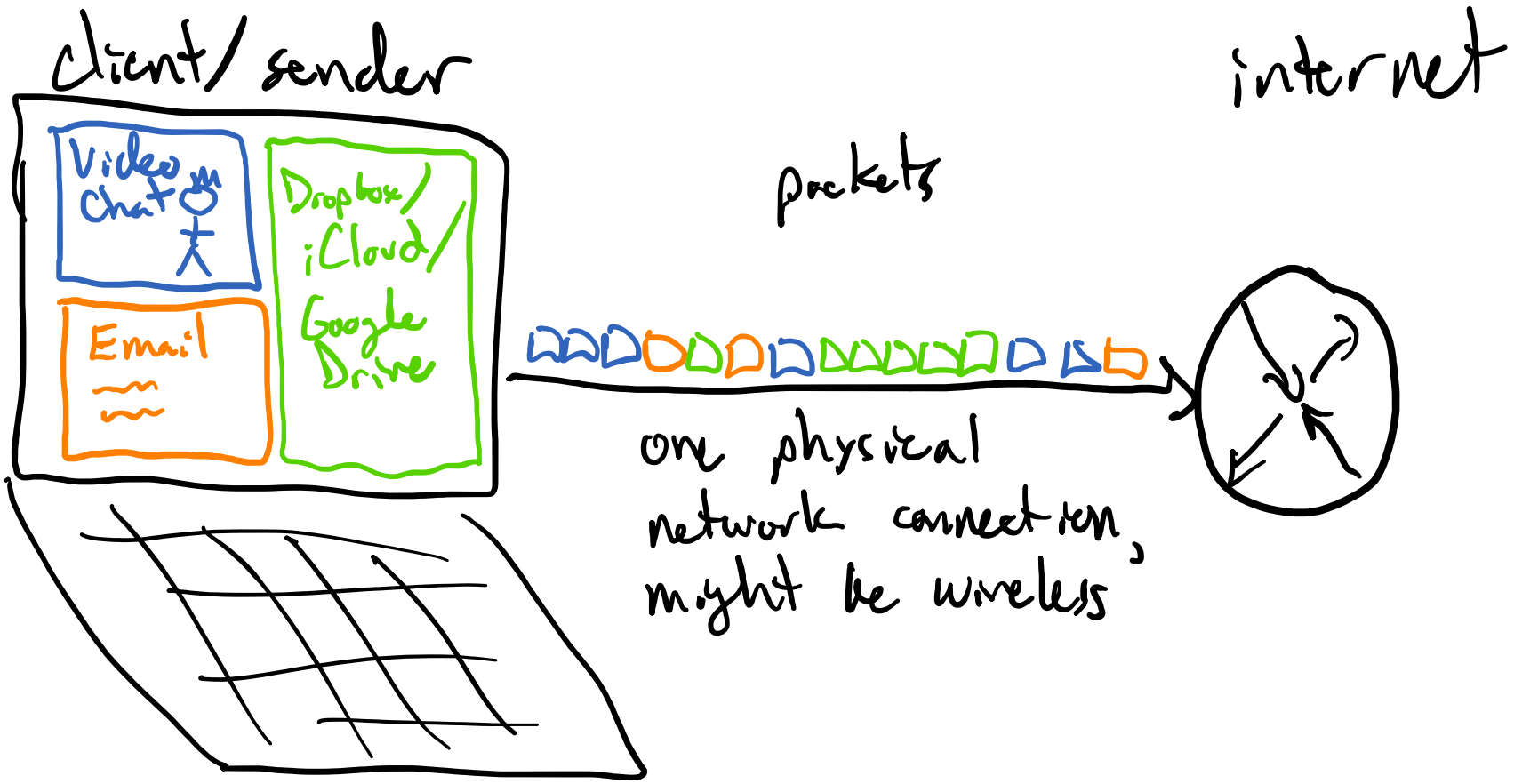
Packets

- Message broken down into packets by sender
  - ~1,000 – 1,500 B each



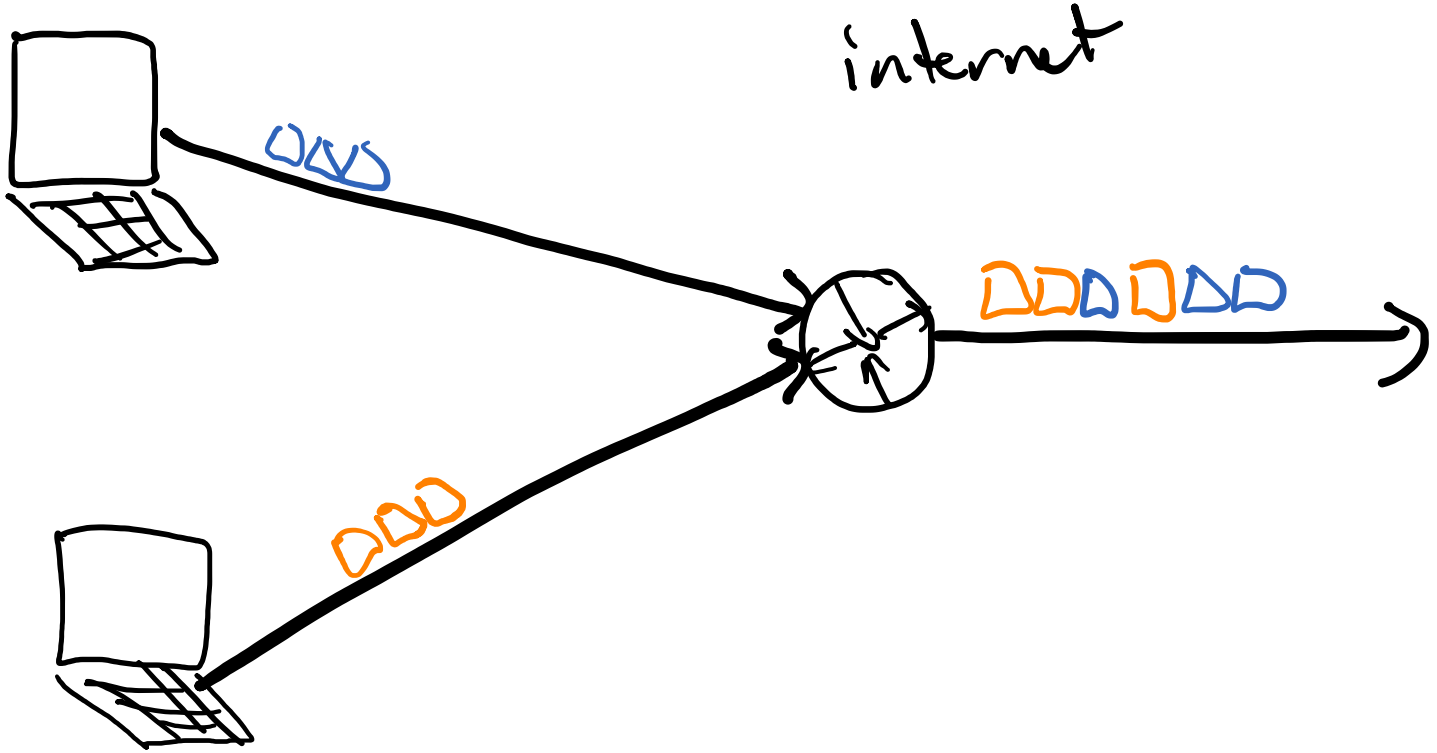
Why packets?

- Multiple programs on one computer can share one connection



Why packets?

- Multiple computers can share one connection



IP summary

- How to get a message from one computer to another computer?
  - Internet Protocol (IP)
- How to tell computers connected to the internet apart?
  - IP Address
- How are computers connected?
  - Routers
- How are long messages broken up? How do multiple computers share one network link?
  - Packets

Agenda

- Motivation
- IP: Internet Protocol
- **TCP: Transmission Control Protocol**
  - Flow control and congestion control
- UDP: User Datagram Protocol
- Sockets
- Summary

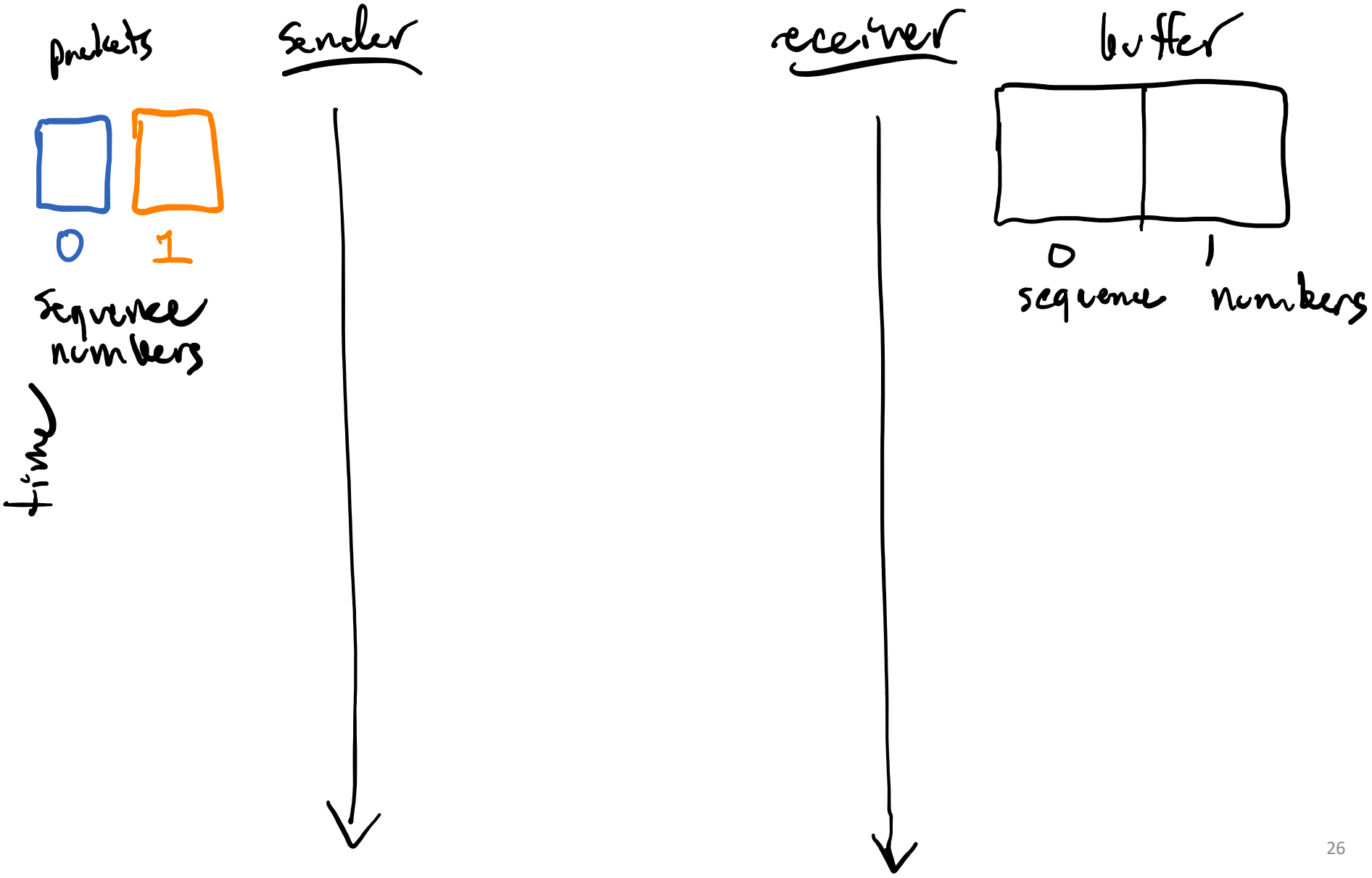
IP is not reliable

- Problems with IP
  - Packets may arrive out of order
  - Packets may disappear
  - Packets may be repeated
- TCP: Abstraction to make it look like these problems don't exist
- TCP/IP: TCP along with IP

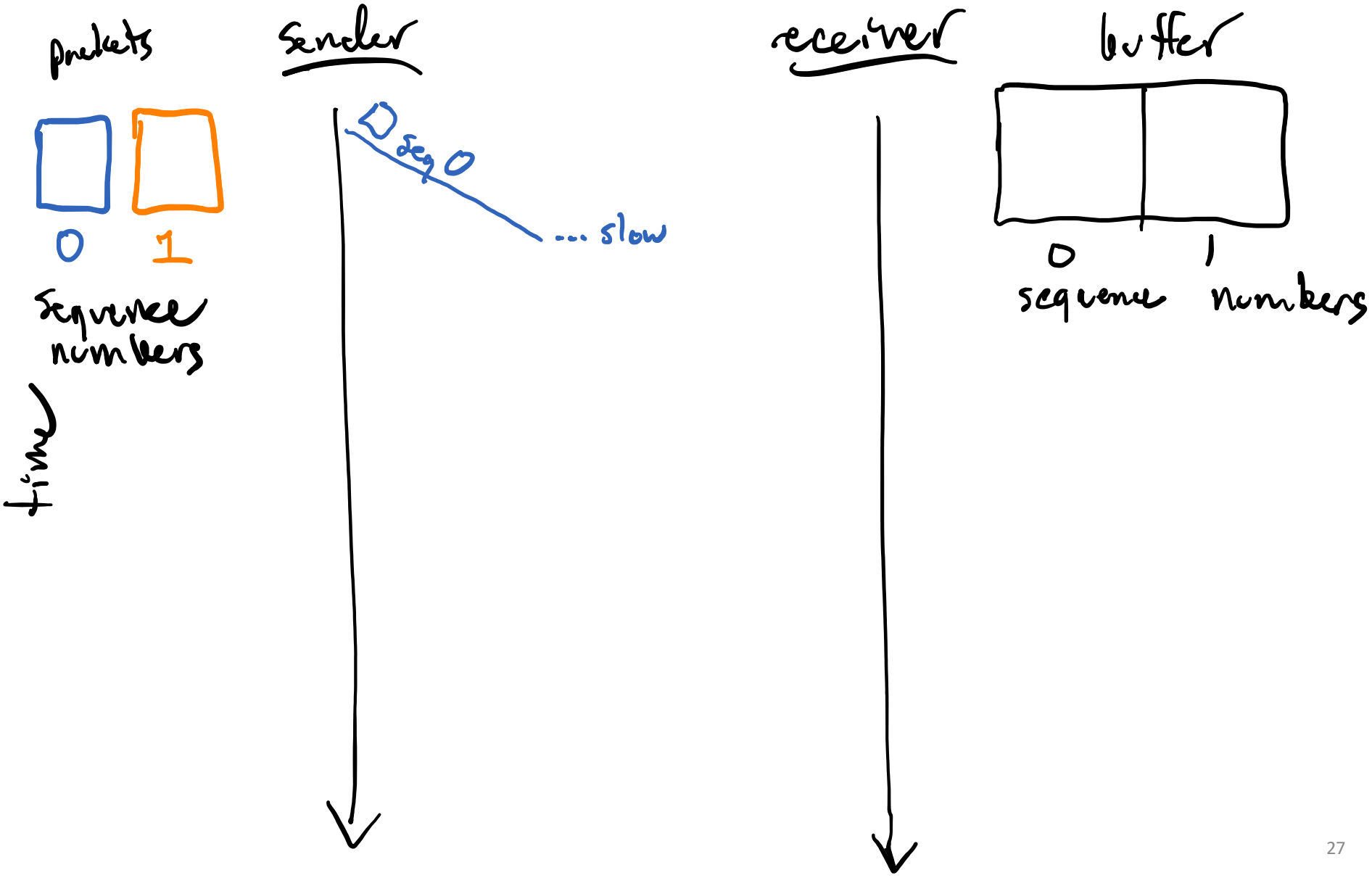
Packets may arrive out of order

- Problem: Packets may arrive out of order
- Solution:
  1. Sender assigns a *sequence number* to each packet
  2. Receiver reassembles packets in order by *sequence number*
  3. Receiver gives data to application (e.g. browser) in order

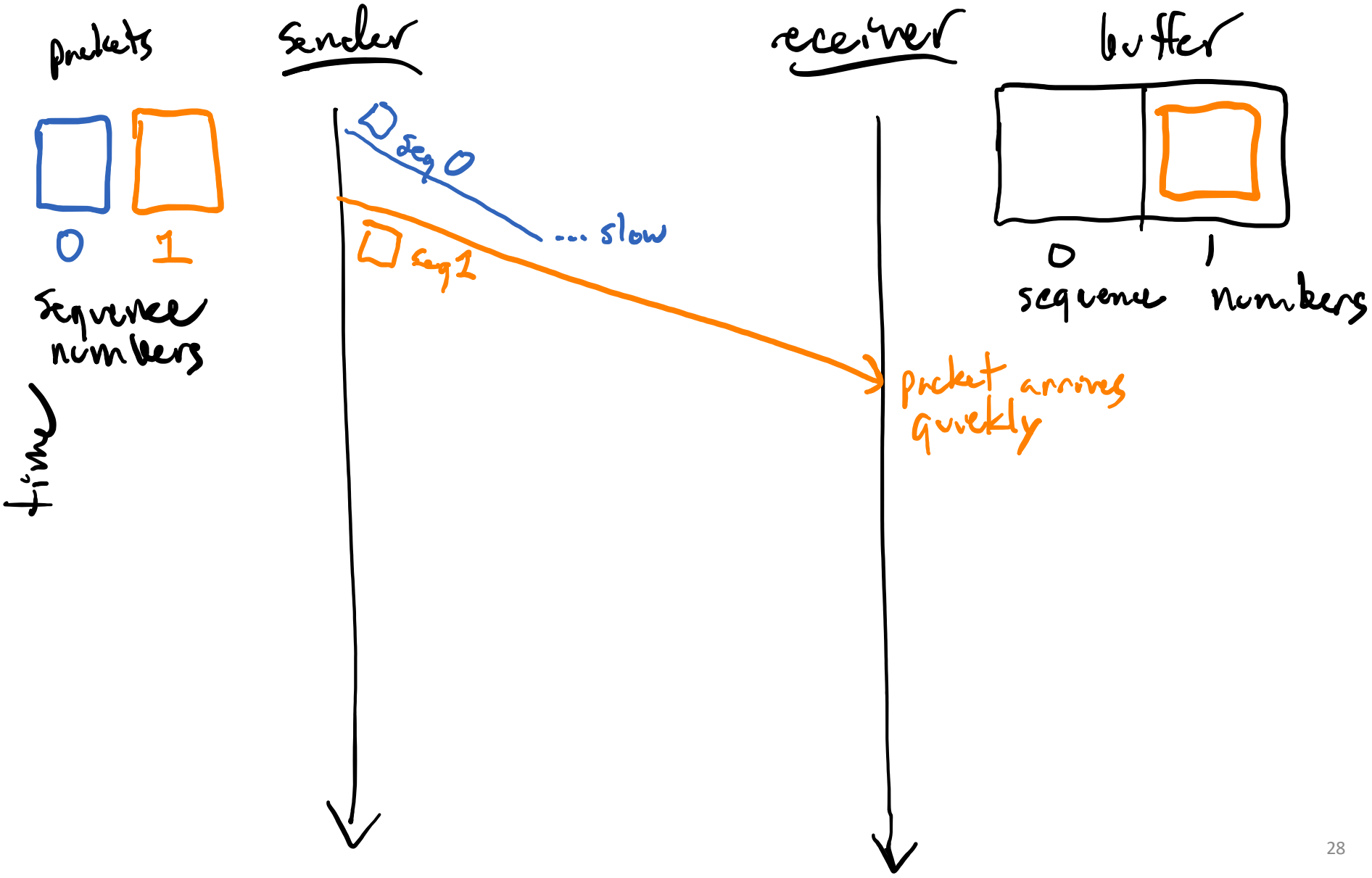
Out of order packets example



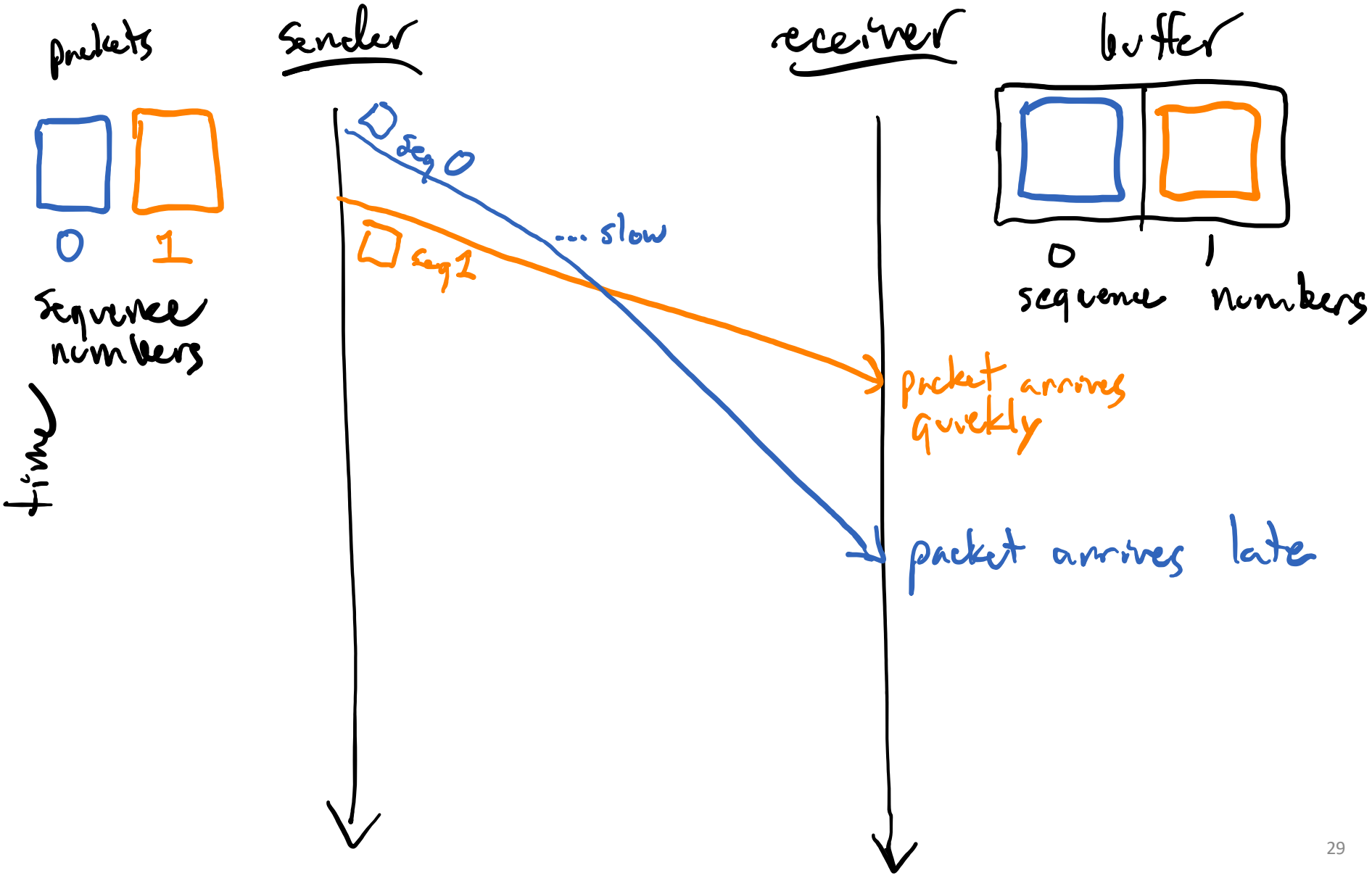
Out of order packets example



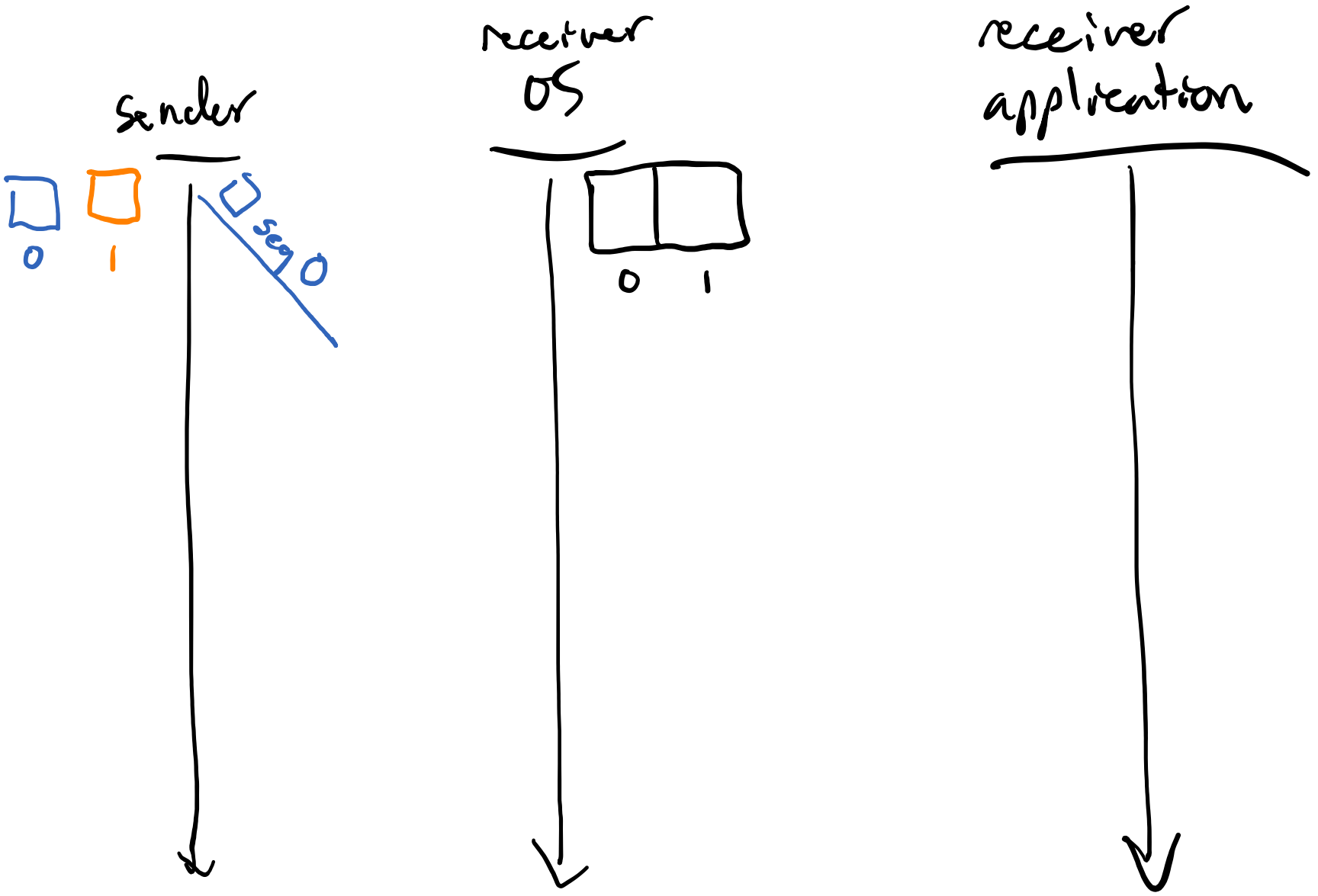
Out of order packets example



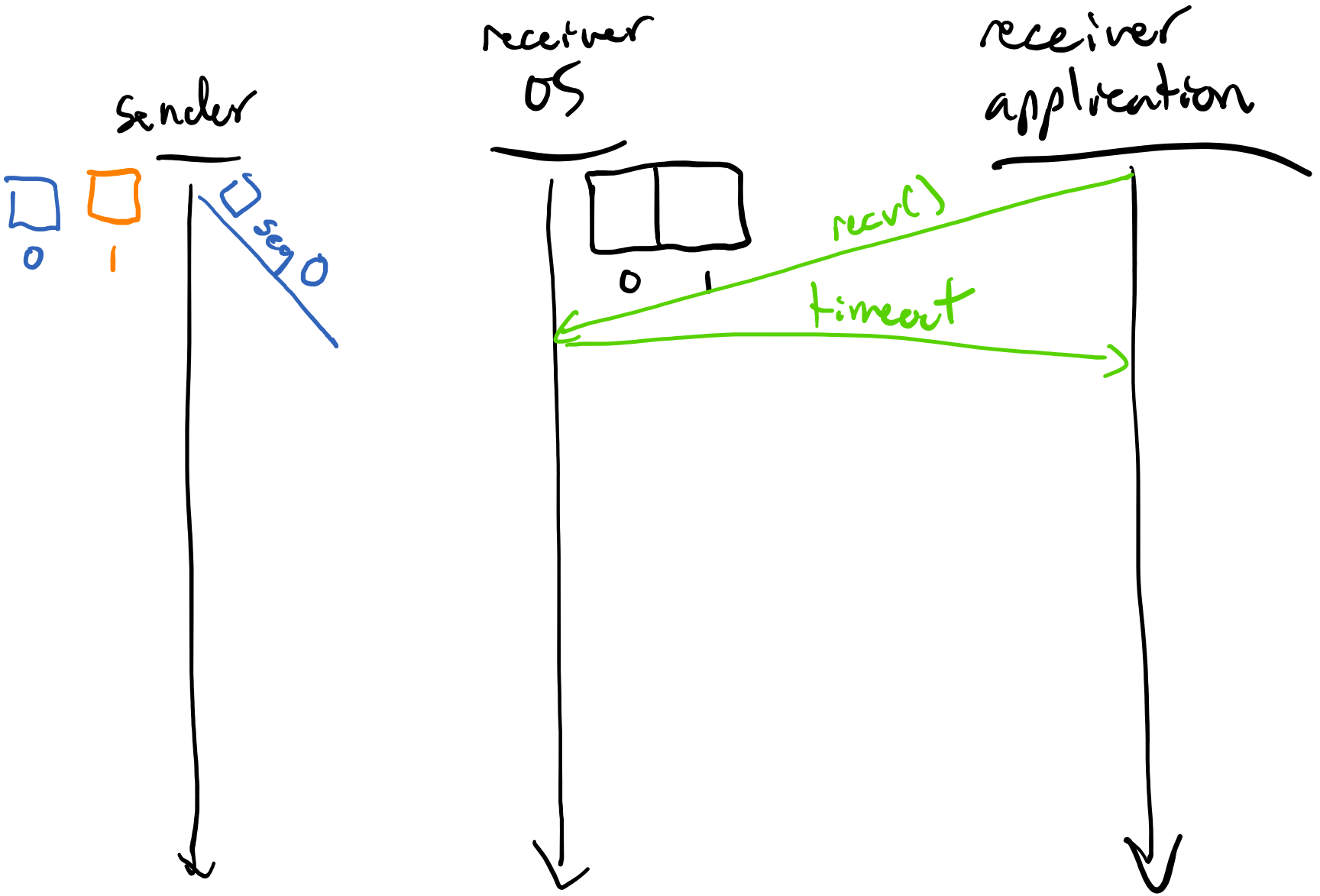
Out of order packets example



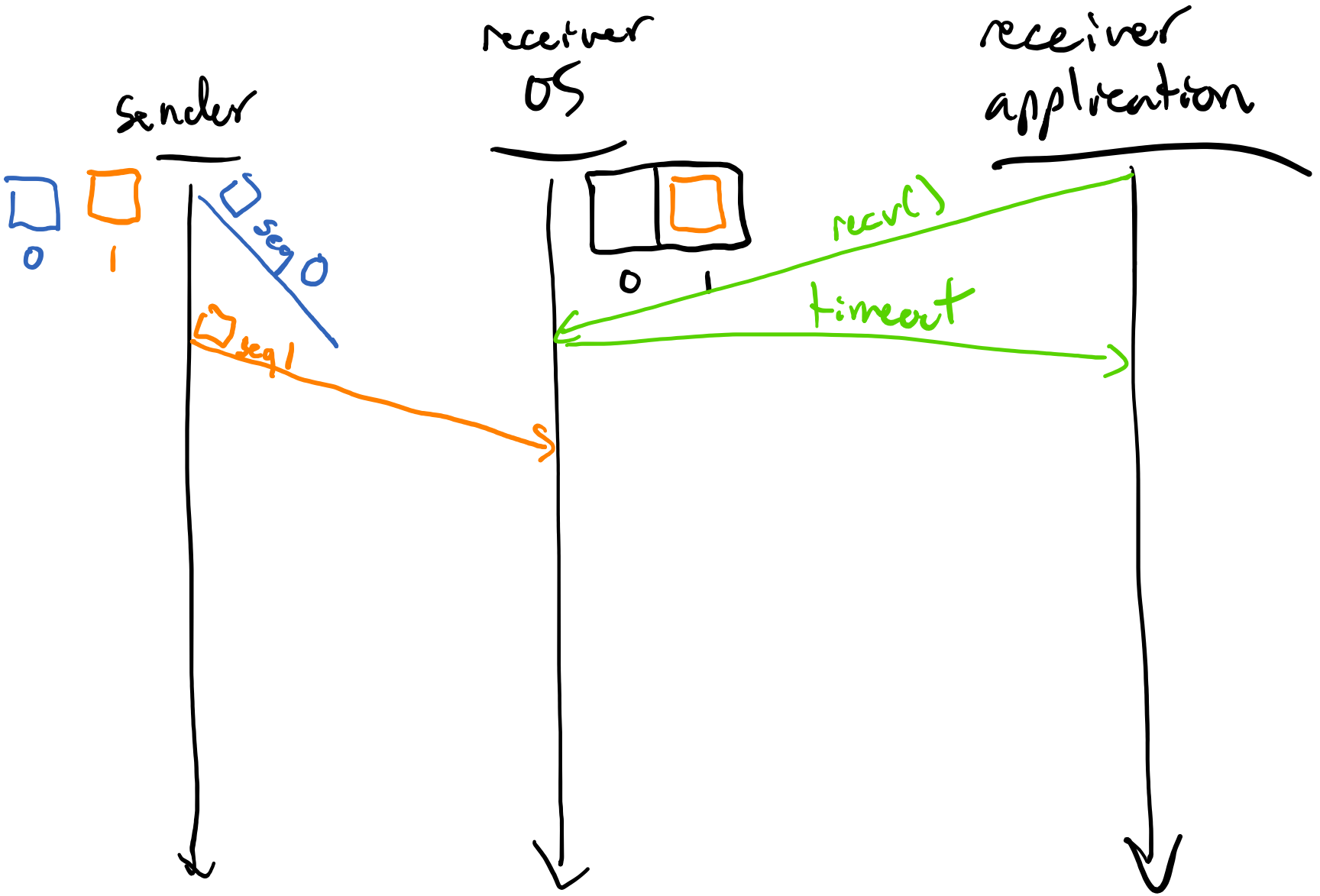
Application receives packets in order



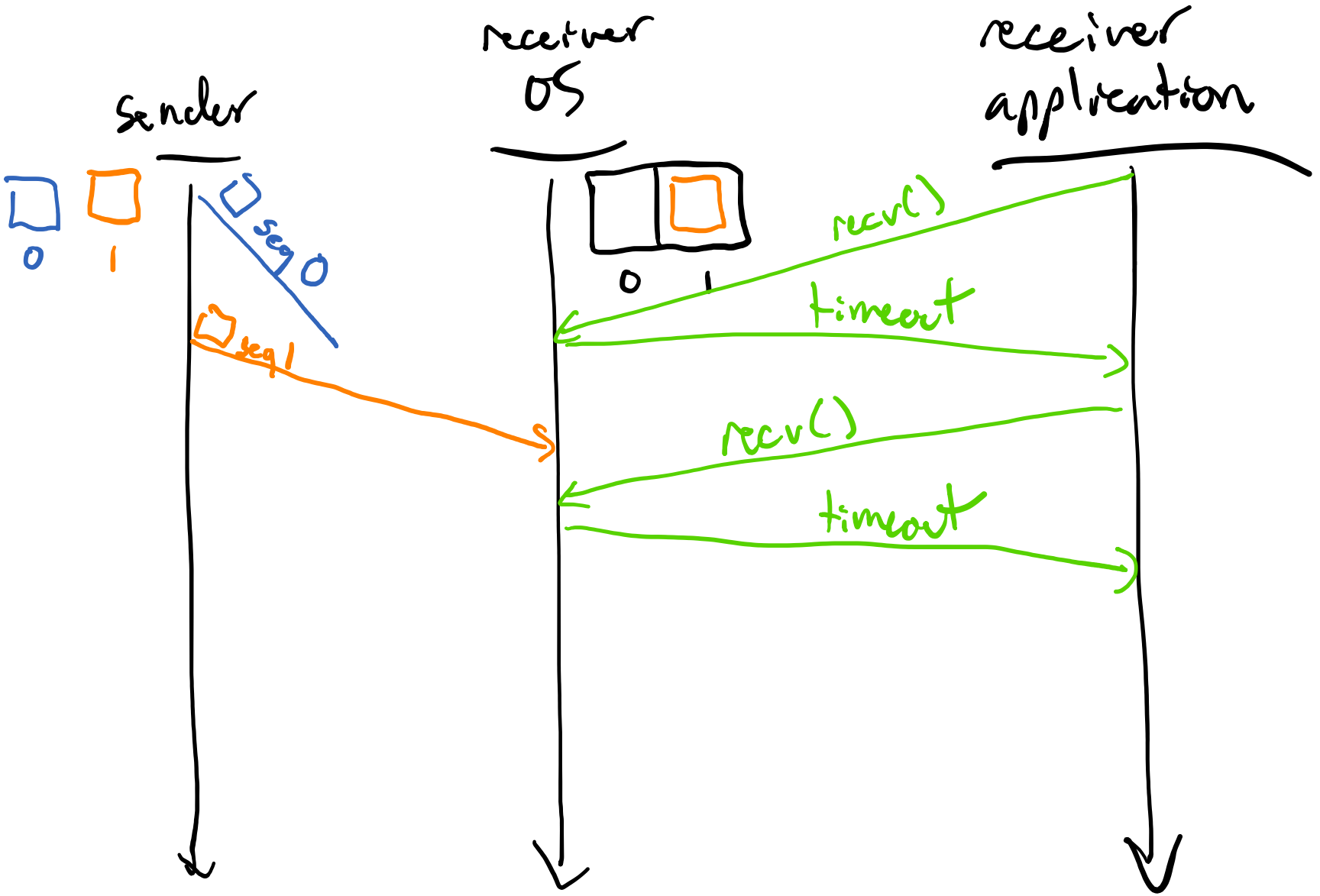
Application receives packets in order



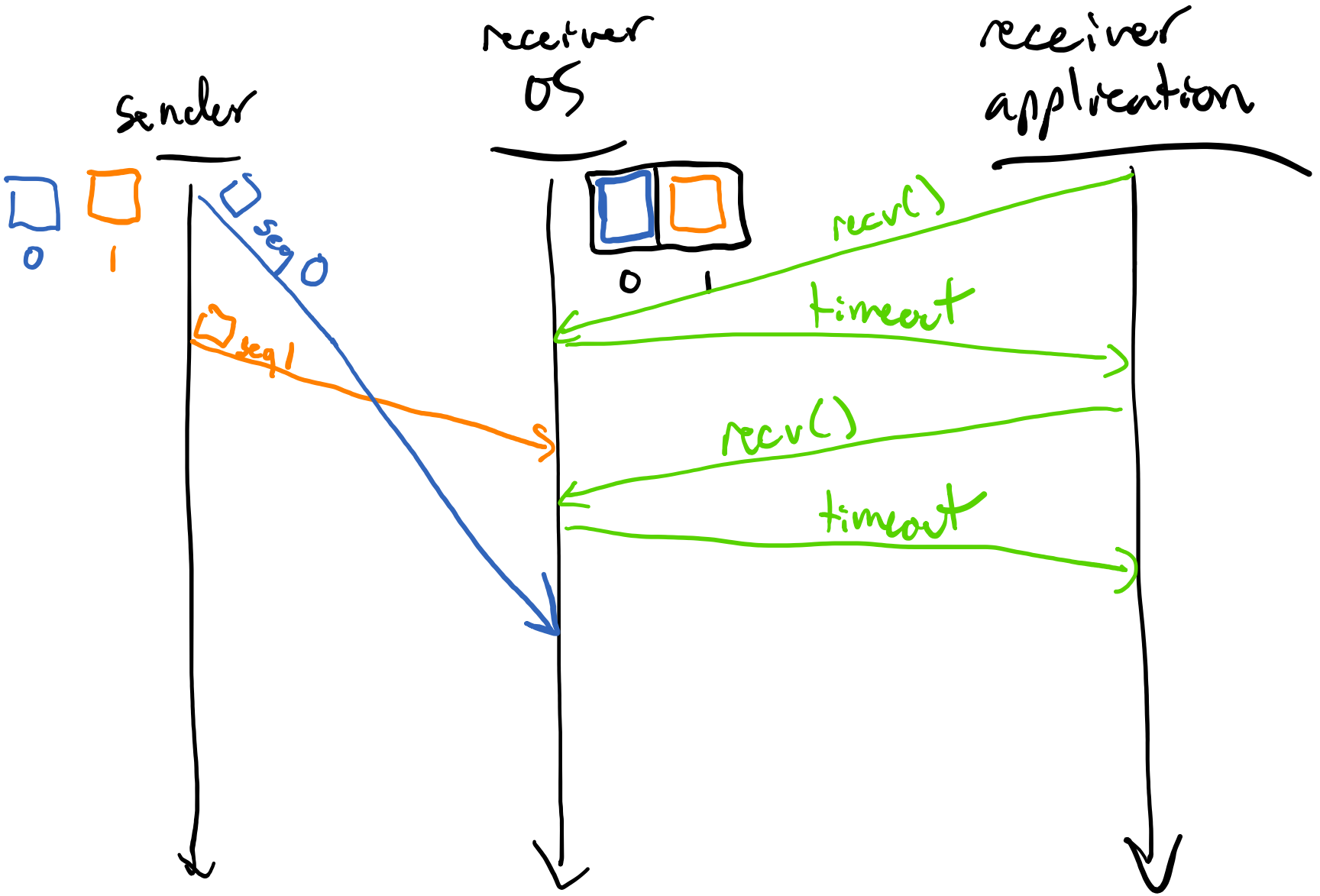
Application receives packets in order



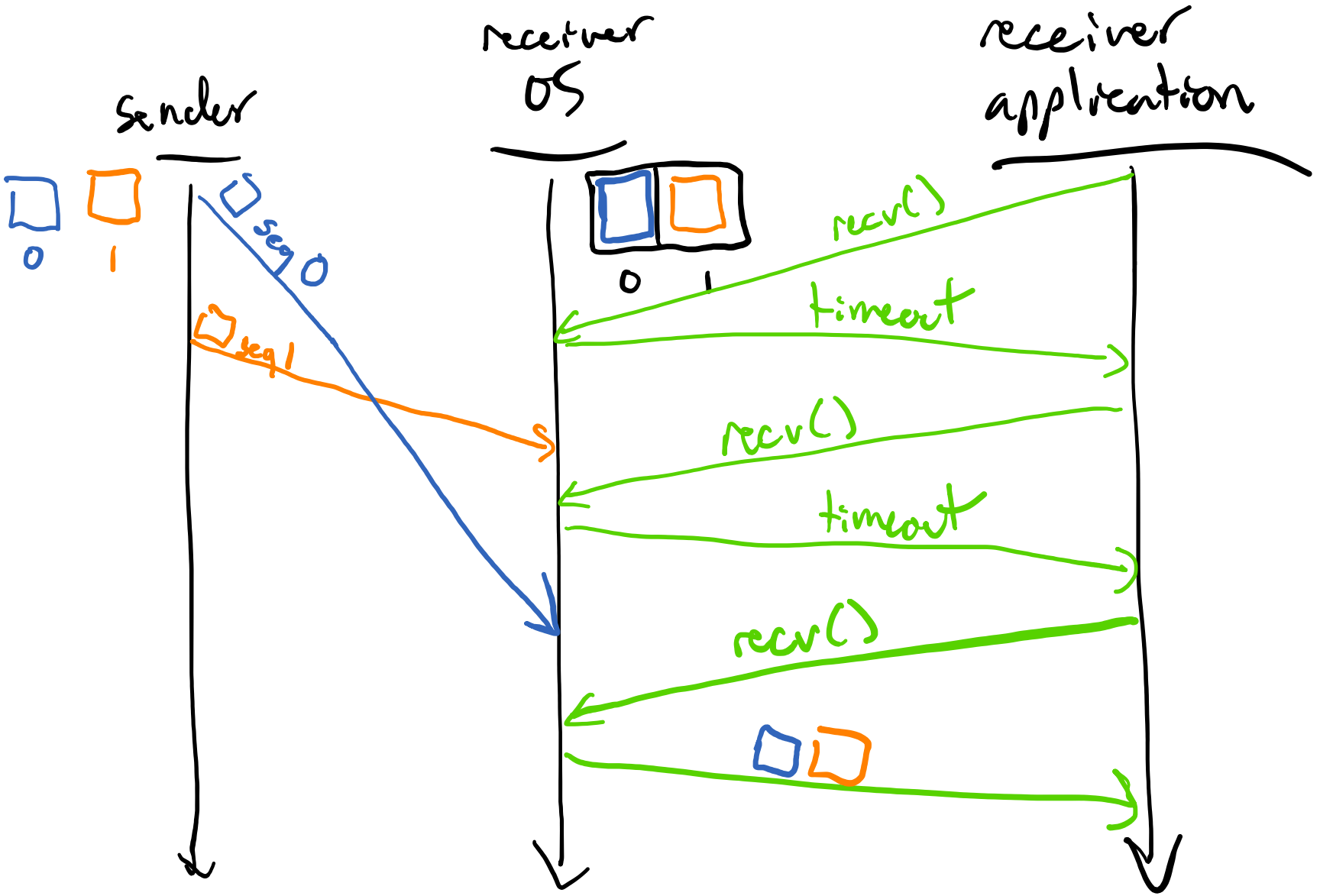
Application receives packets in order



Application receives packets in order



Application receives packets in order

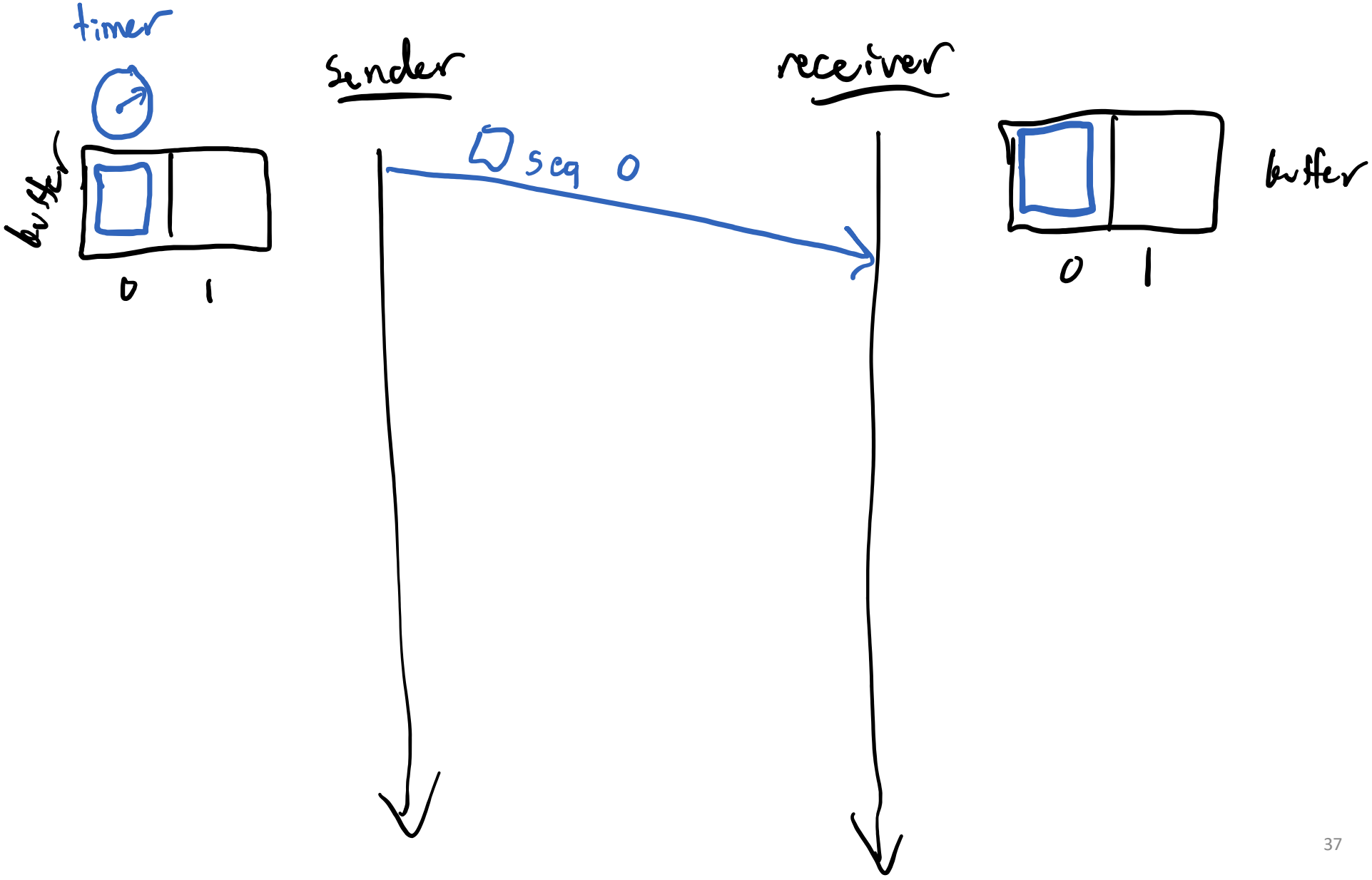


Packets may disappear

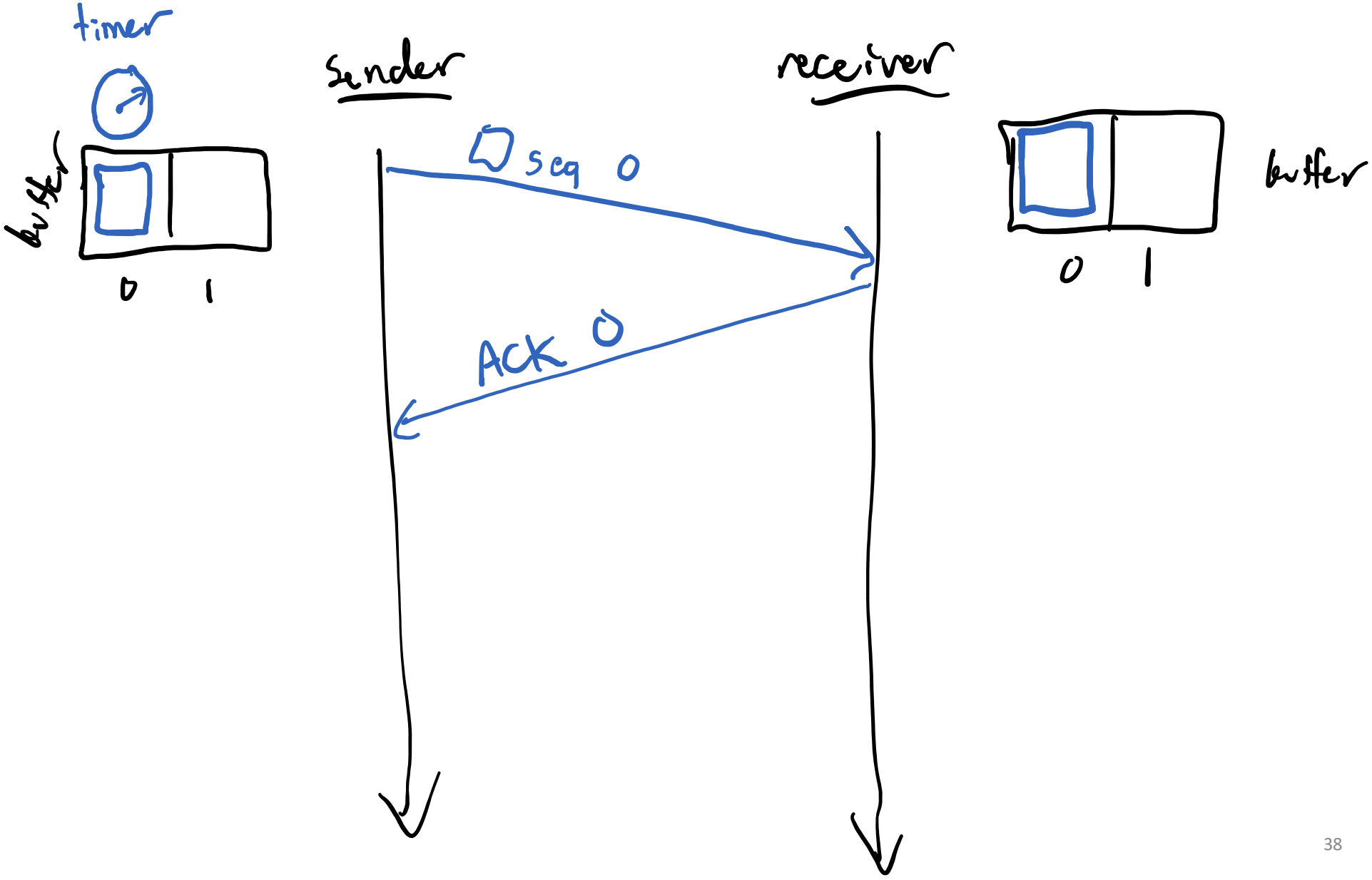
- Problem: Packets may disappear
- Solution: Sender stores data in buffer until receiver ACK

1. Sender stores a copy of each packet
2. Sender sets a timer for each packet when it goes out
3. Receiver sends an acknowledgement (ACK) for each packet
4. Sender resends the packet if the timer expires
    1. Delete the packet upon receiving ACK

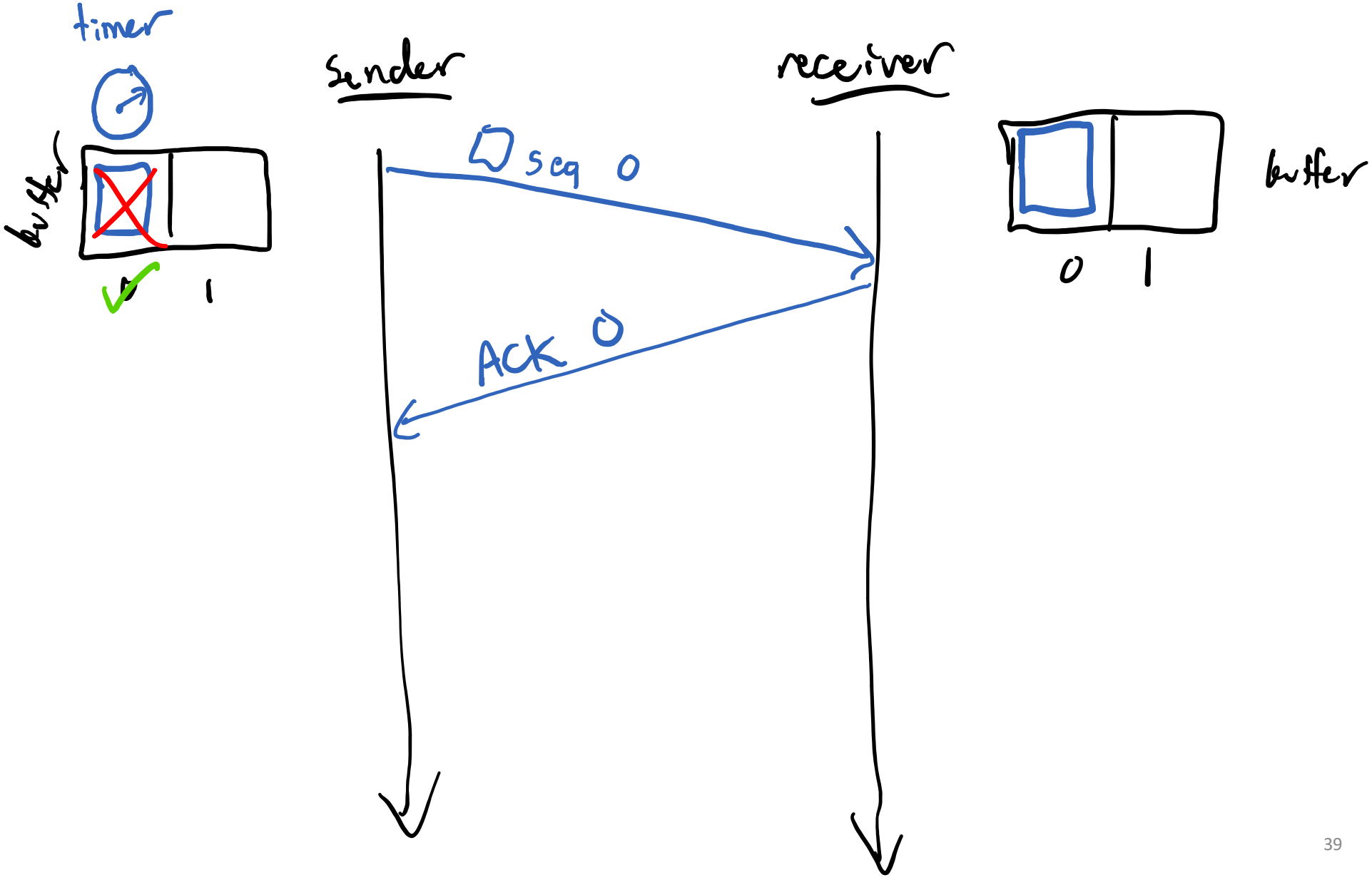
Packet acknowledgement example



Packet acknowledgement example



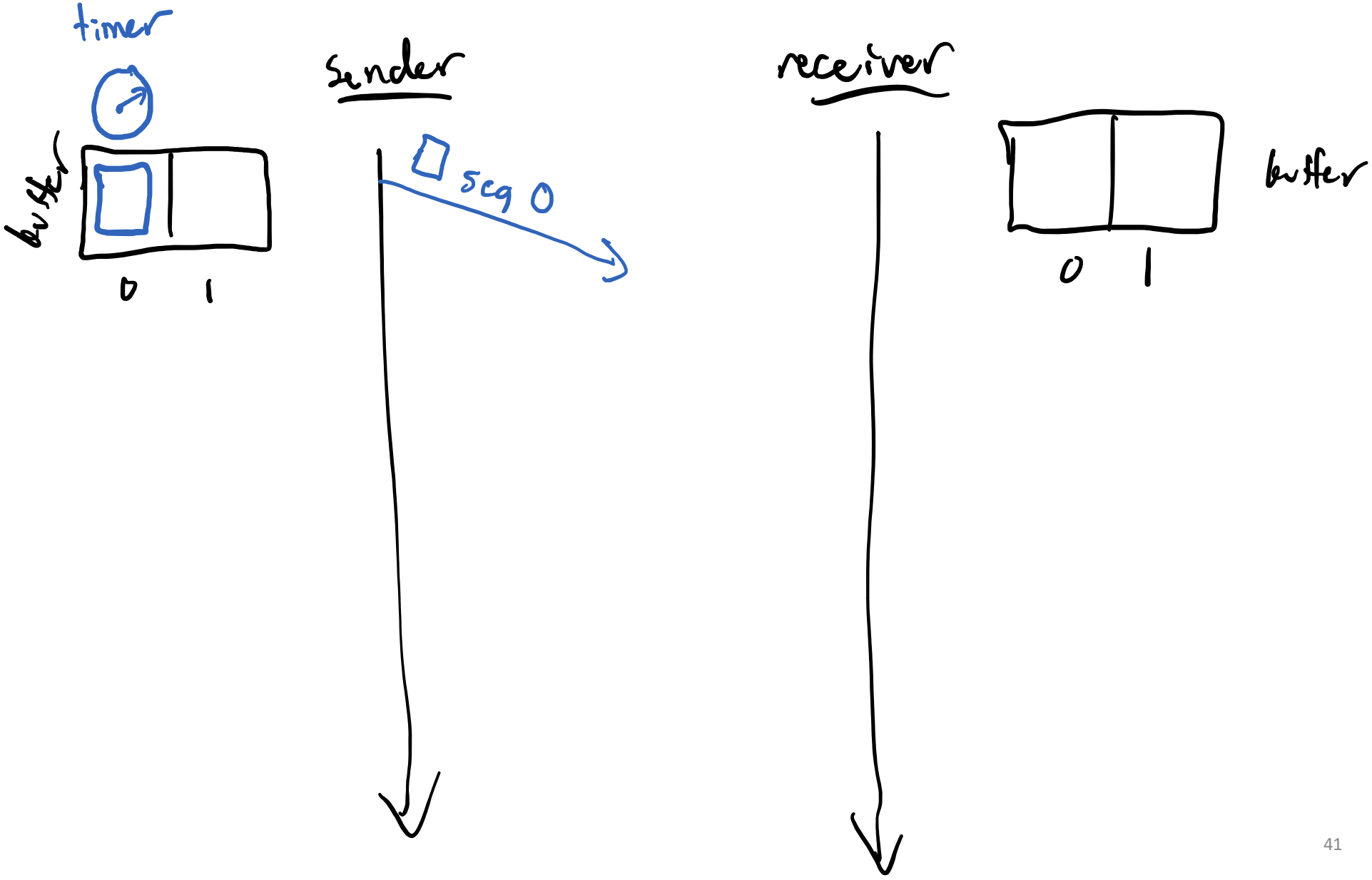
Packet acknowledgement example



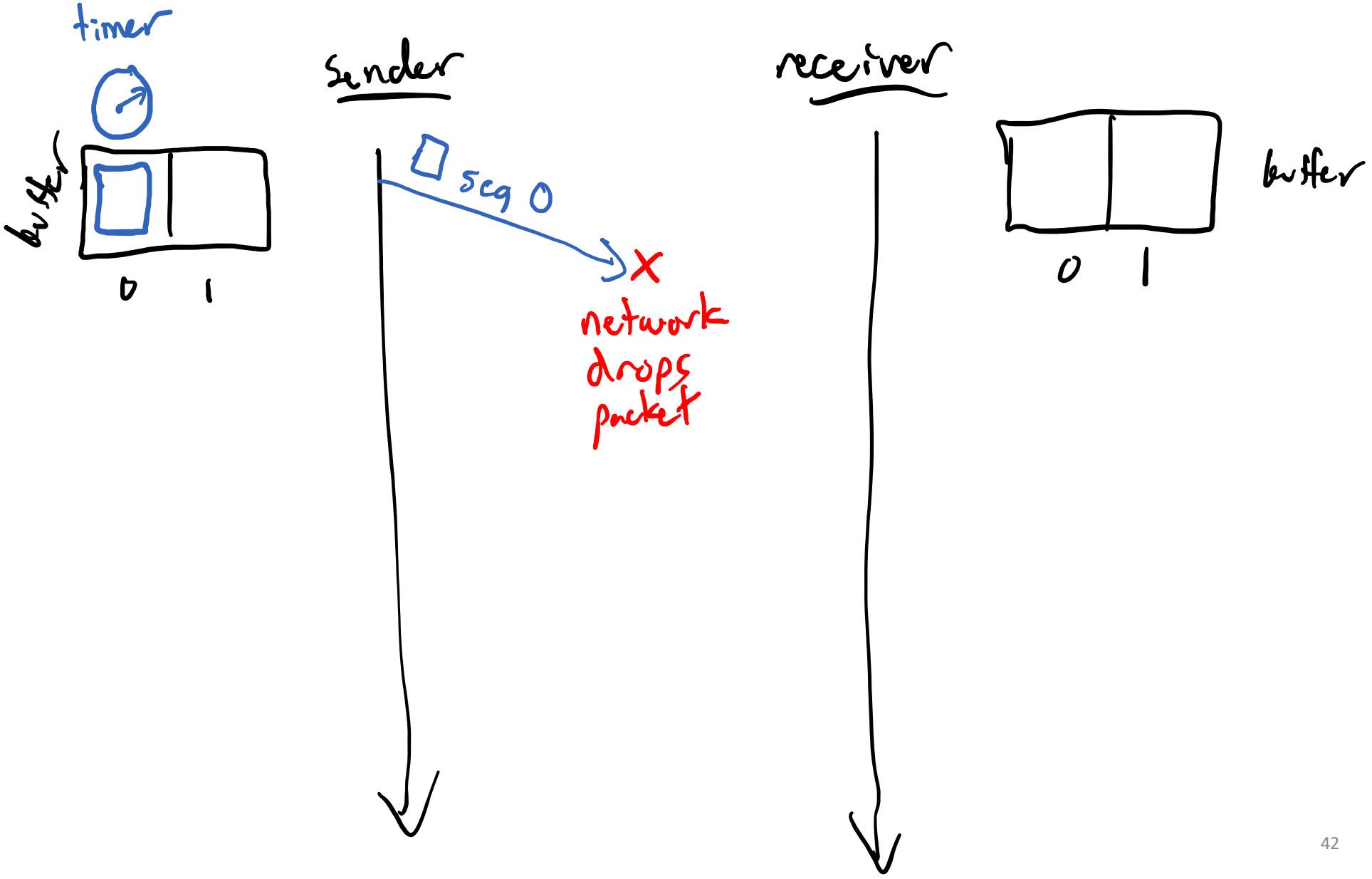
Disappearing packet

- What happens if the packet disappears?
- Sender resends the packet if the timer expires

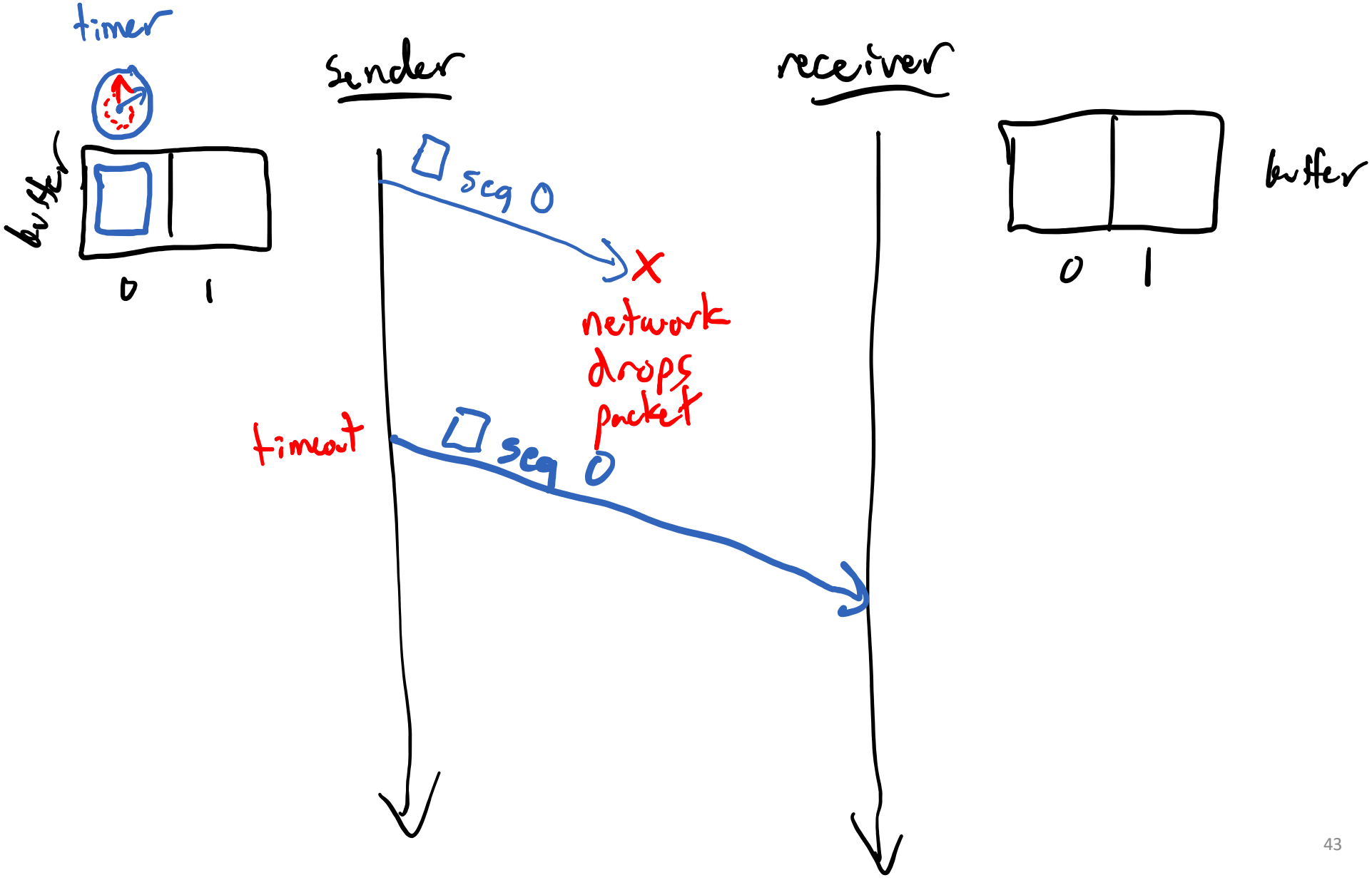
Disappearing packet example



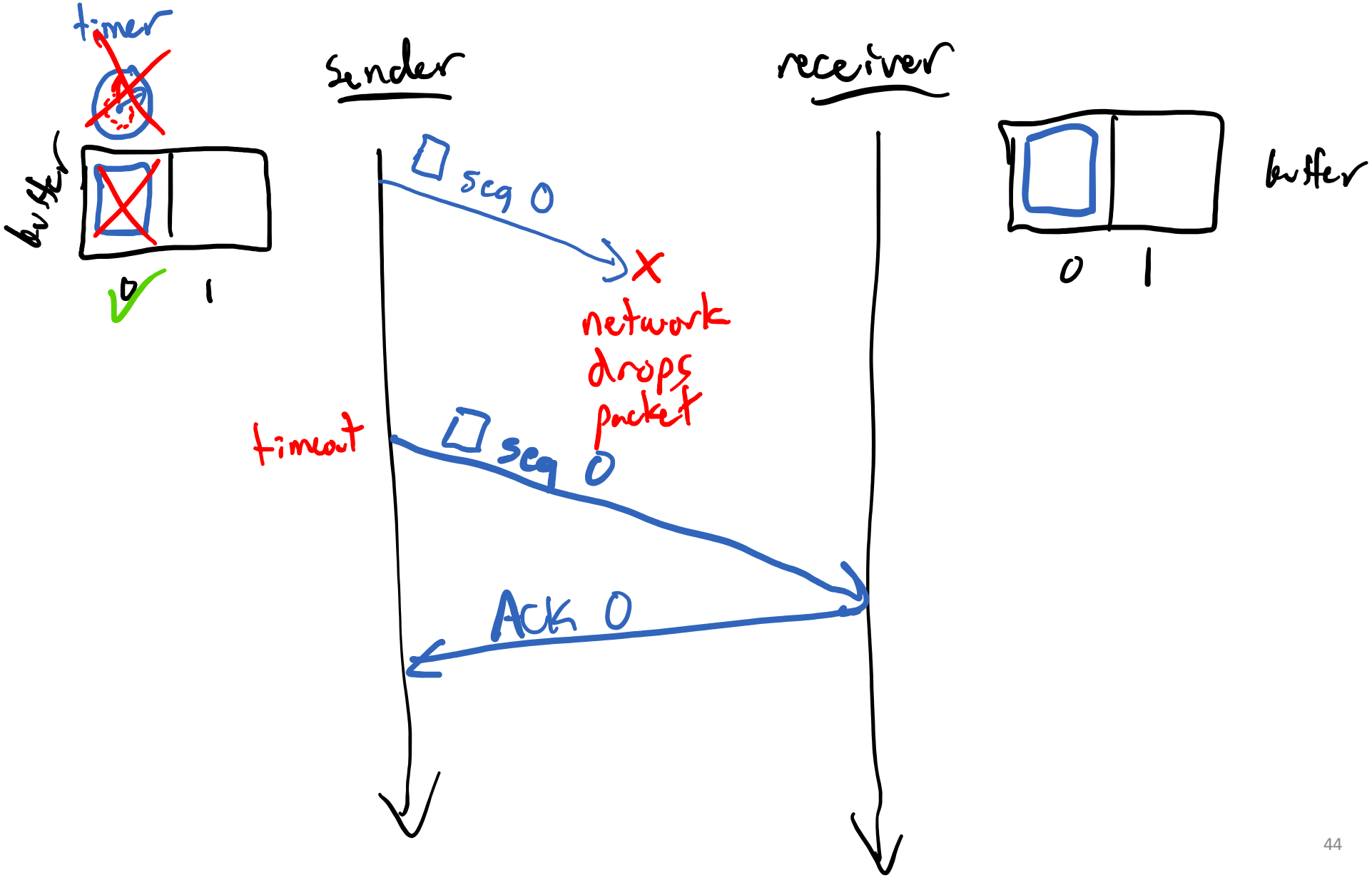
Disappearing packet example



Disappearing packet example



Disappearing packet example



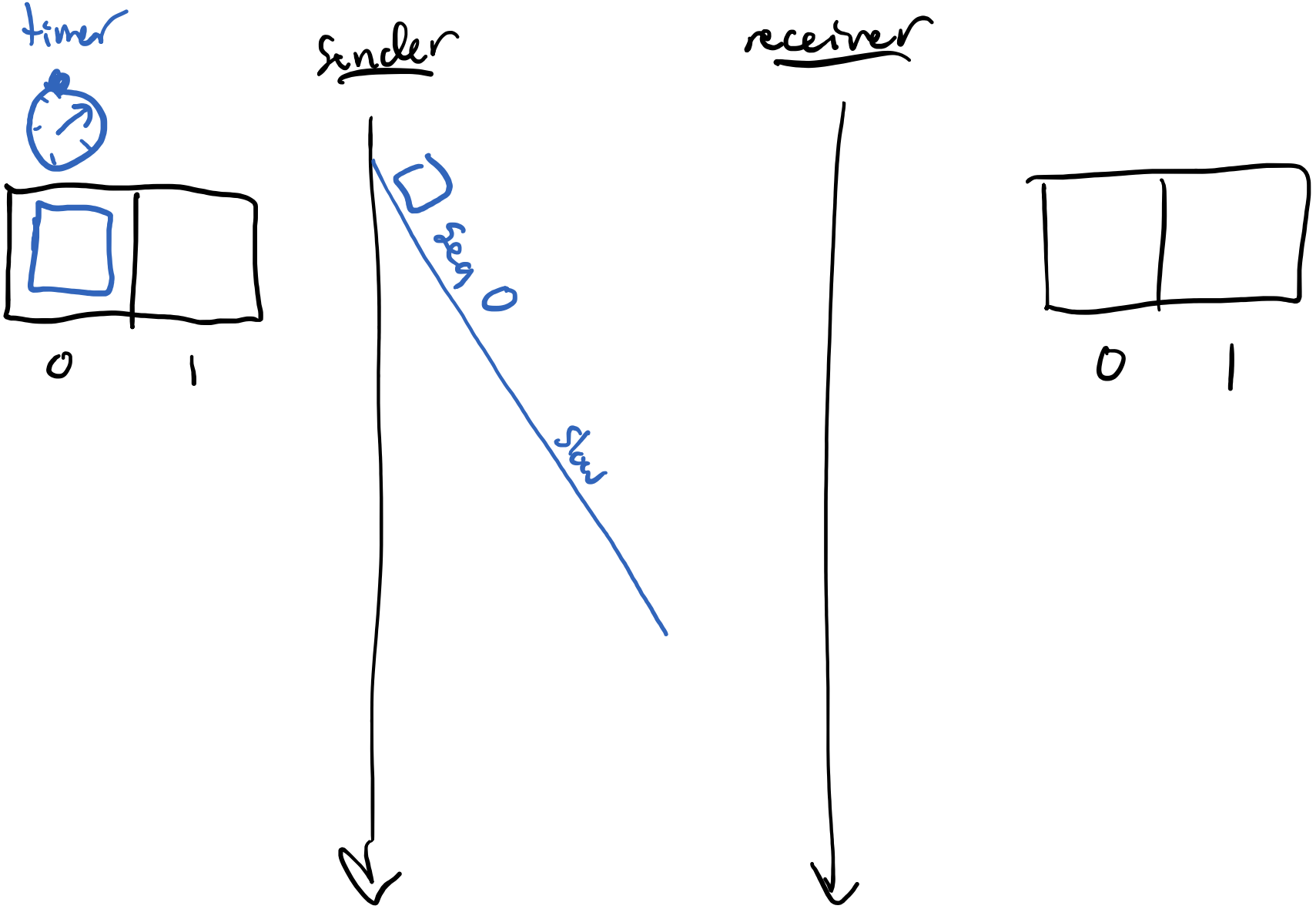
Packets may be repeated

- Problem: Packets may be repeated
- Solution: Receiver ignores sequence numbers it has seen before

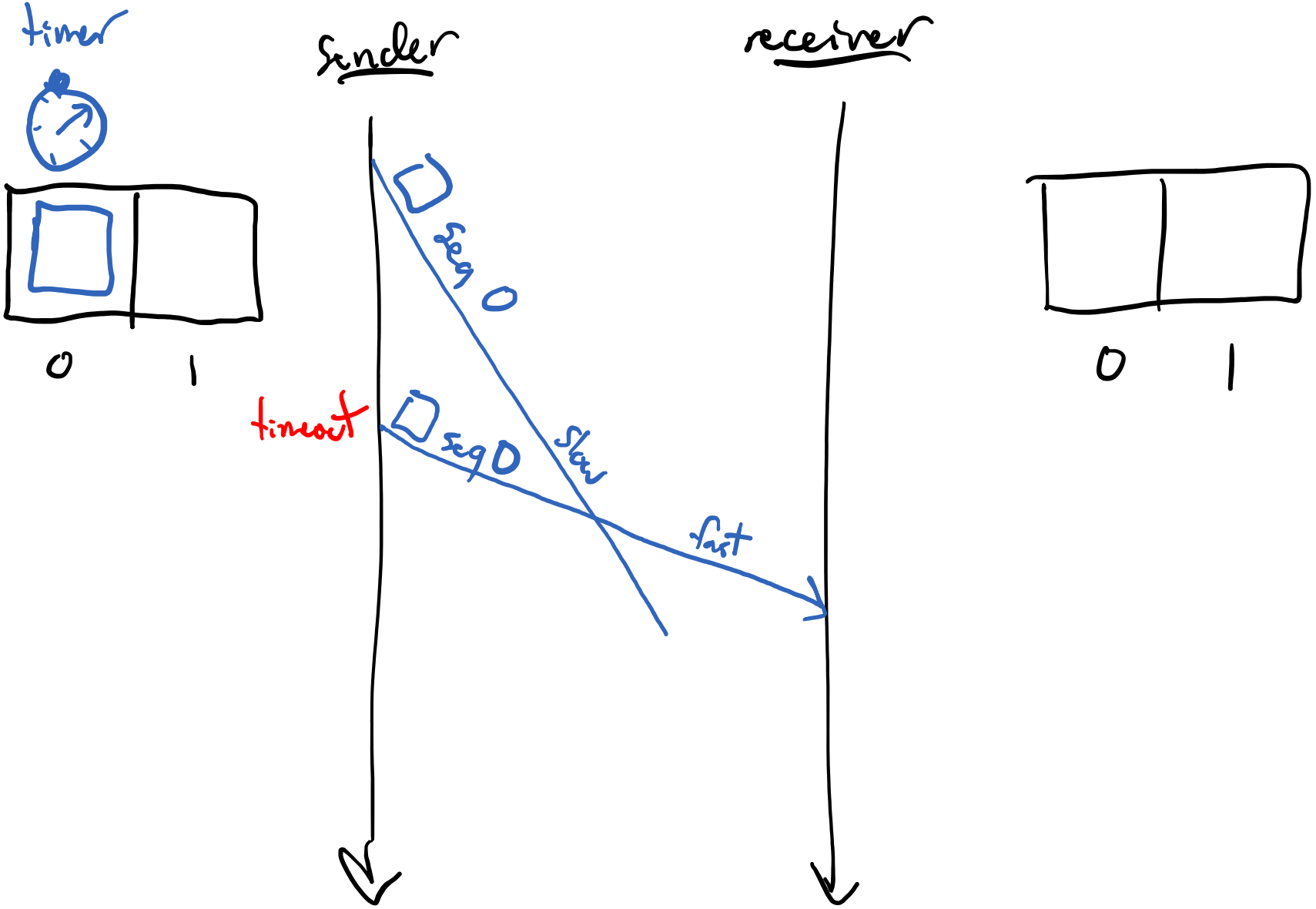
- Why would a packet be repeated?
  - ACK is dropped
  - Packet is slow

- Let's look at an example of a slow packet

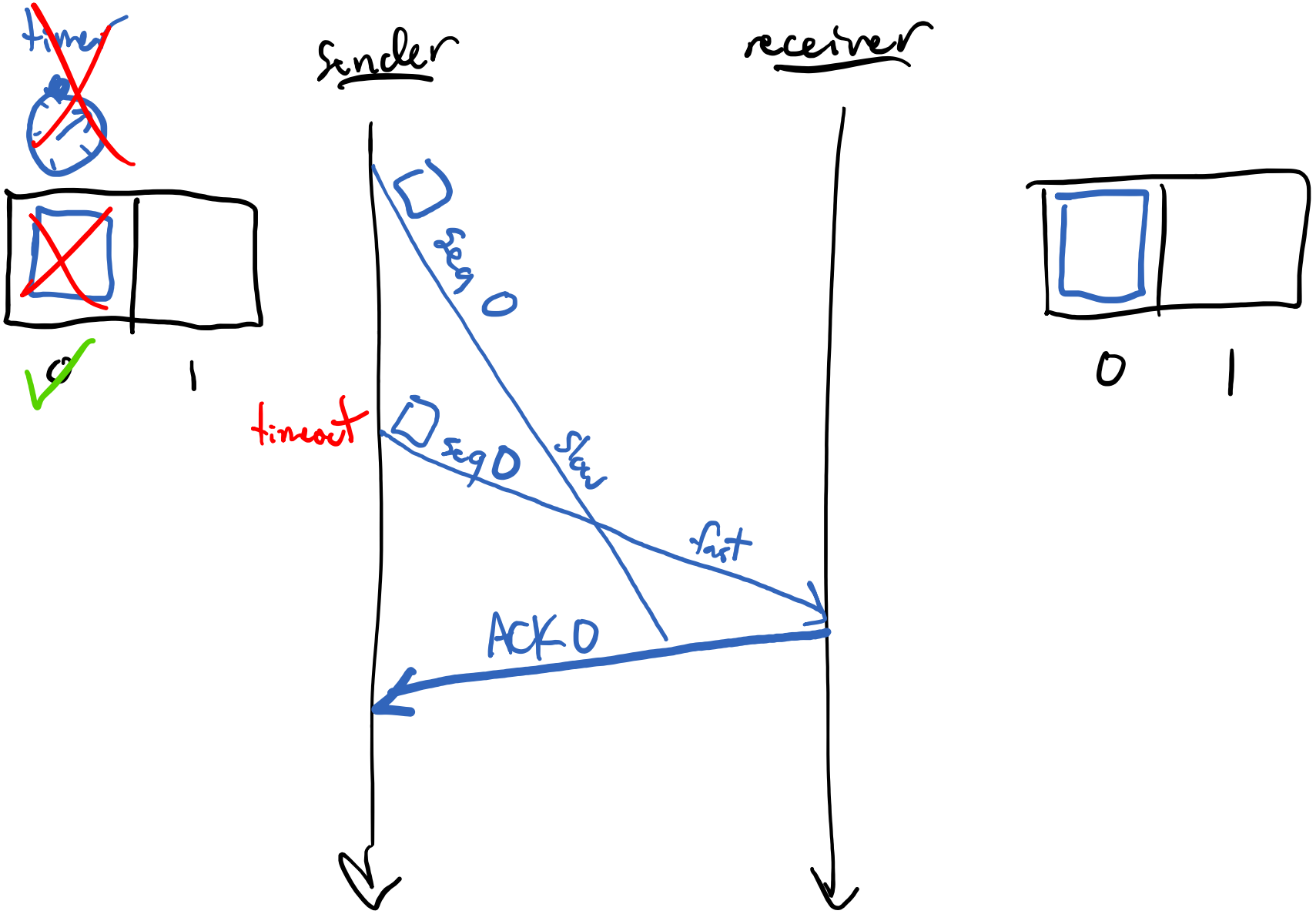
Repeated packet example



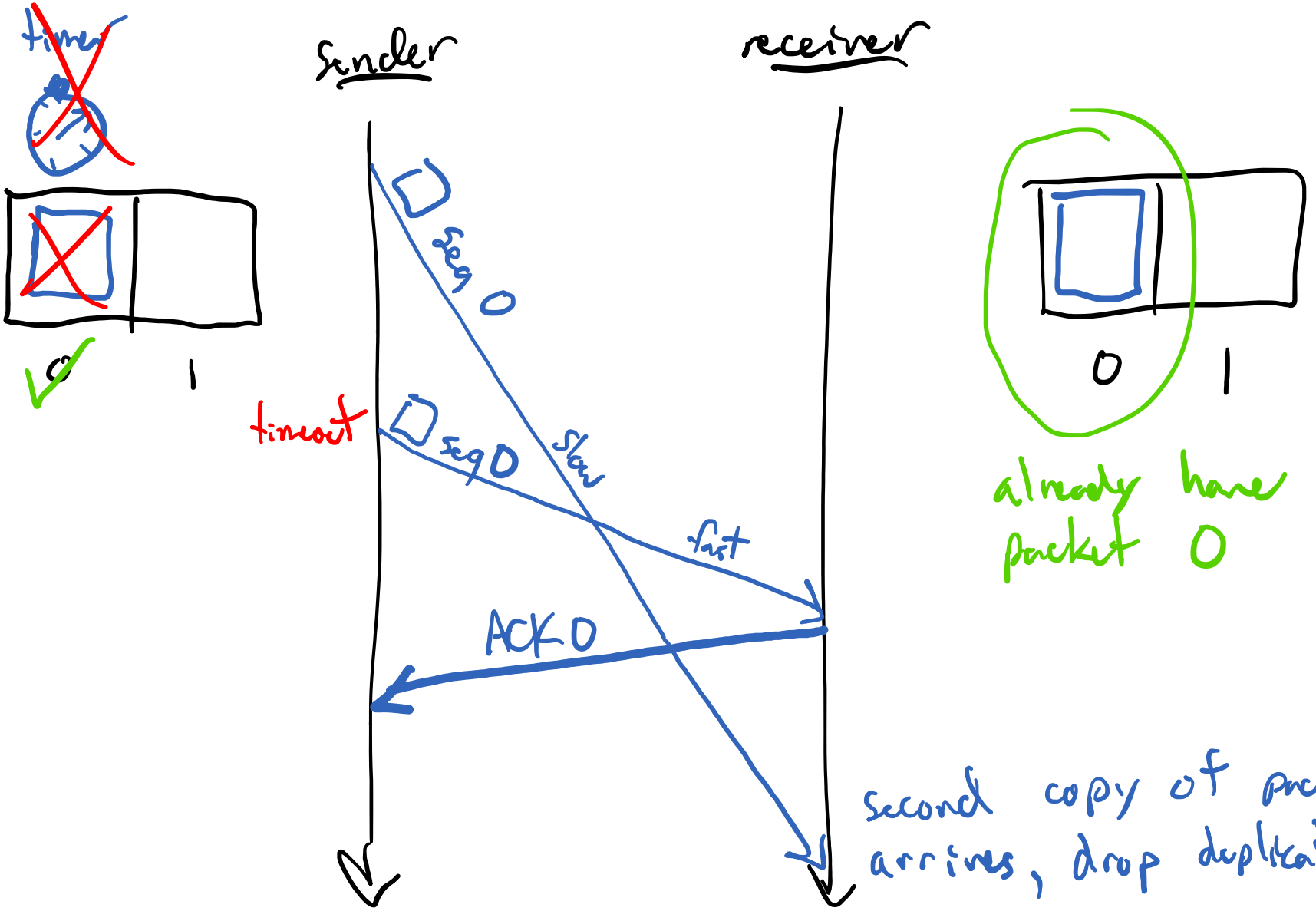
Repeated packet example



Repeated packet example



Repeated packet example

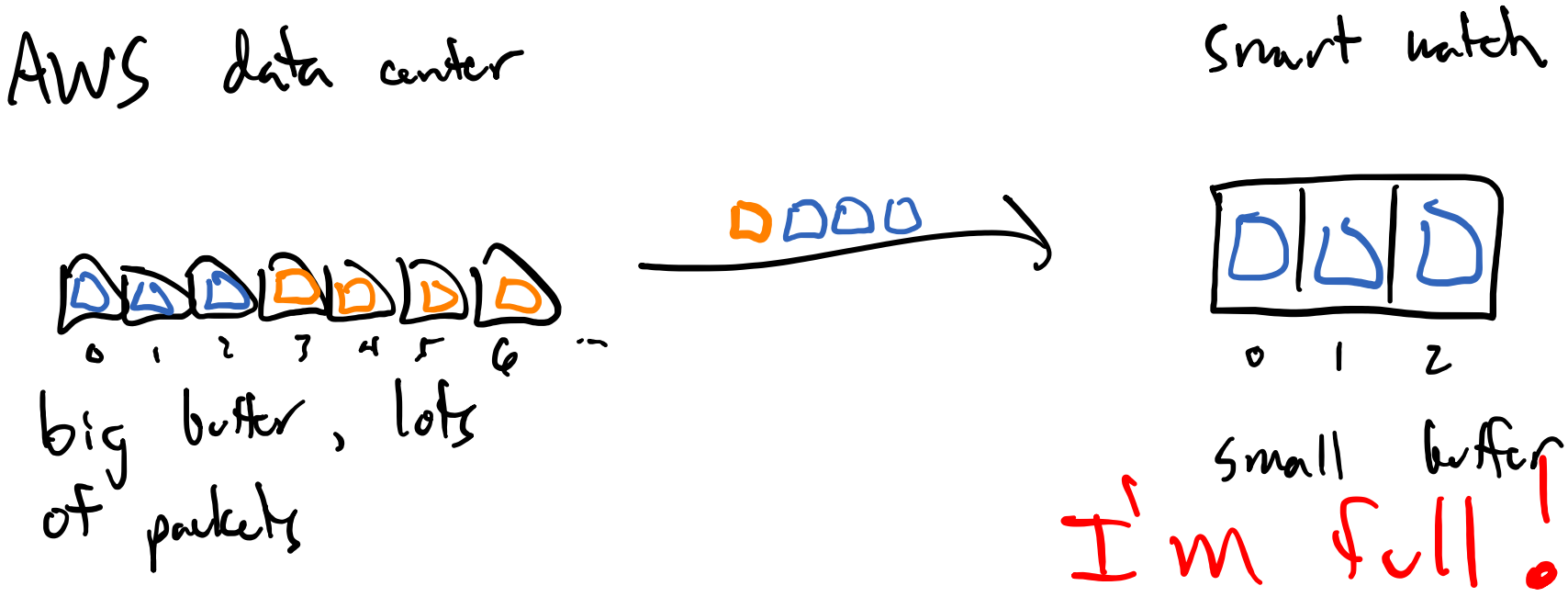


Agenda

- Motivation
- IP: Internet Protocol
- TCP: Transmission Control Protocol
  - **Flow control and congestion control**
- UDP: User Datagram Protocol
- Sockets
- Summary

Fast sender, slow receiver

- Problem: Fast sender overloads a slow receiver
- For example, sender is an AWS datacenter server and receiver is a smart watch

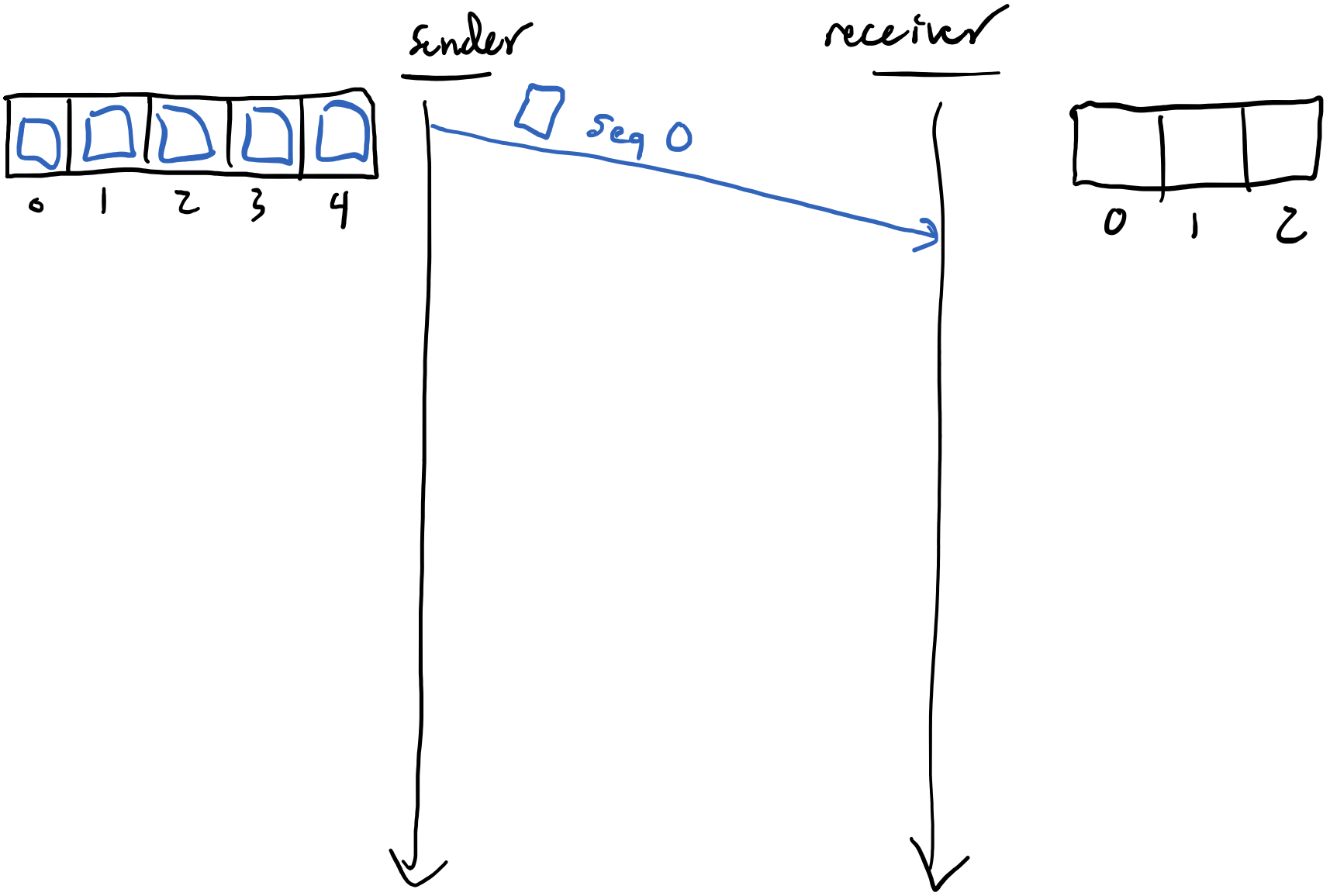


Flow control

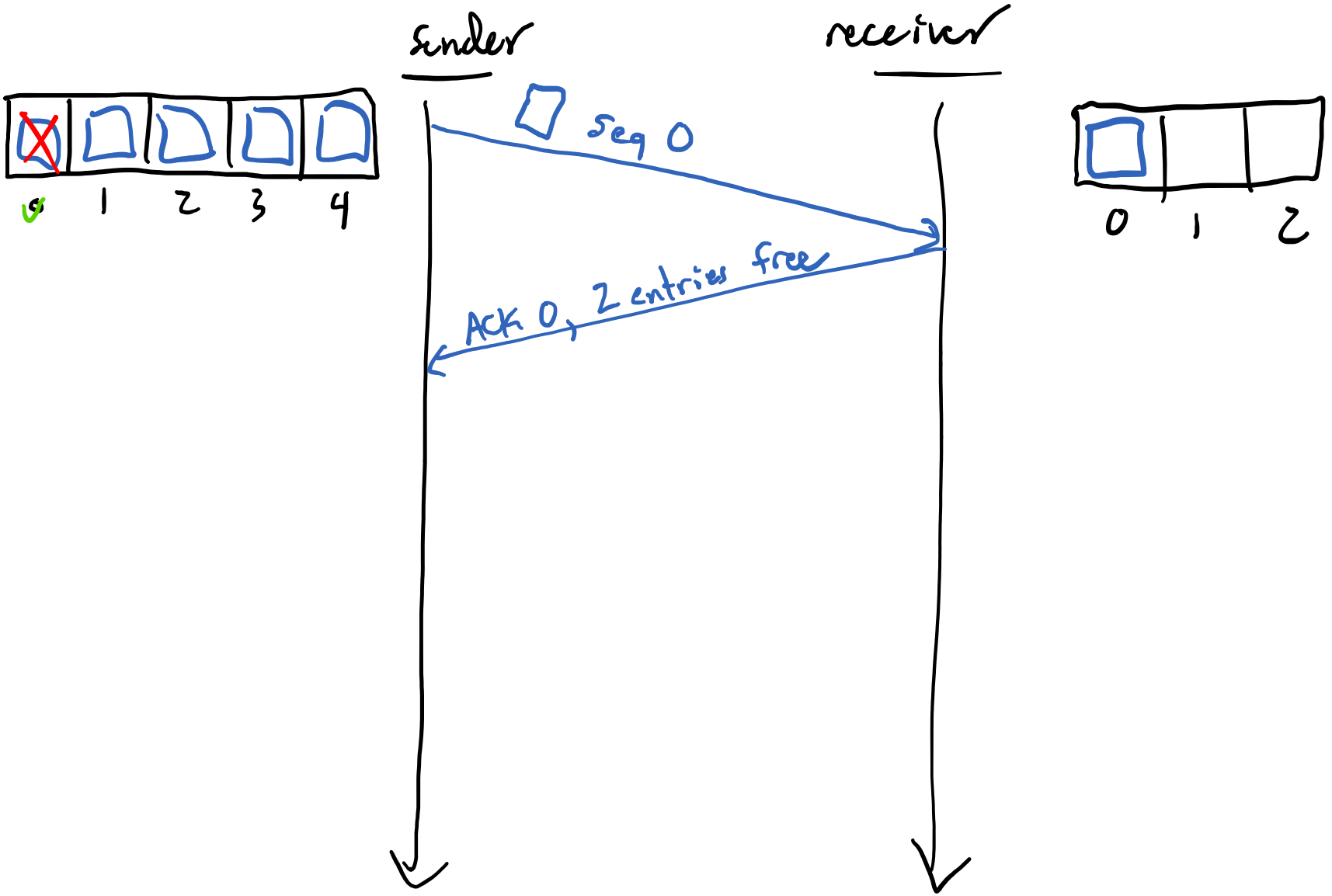
- Problem: Fast sender overloads a slow receiver
- Solution: *Flow control*

- Receiver tells the sender how many empty buffer spots it has
- Sender can put that much data on the network at one time
- This is called the *sliding window*
- Sometimes called the *receiver window* or *RWND*

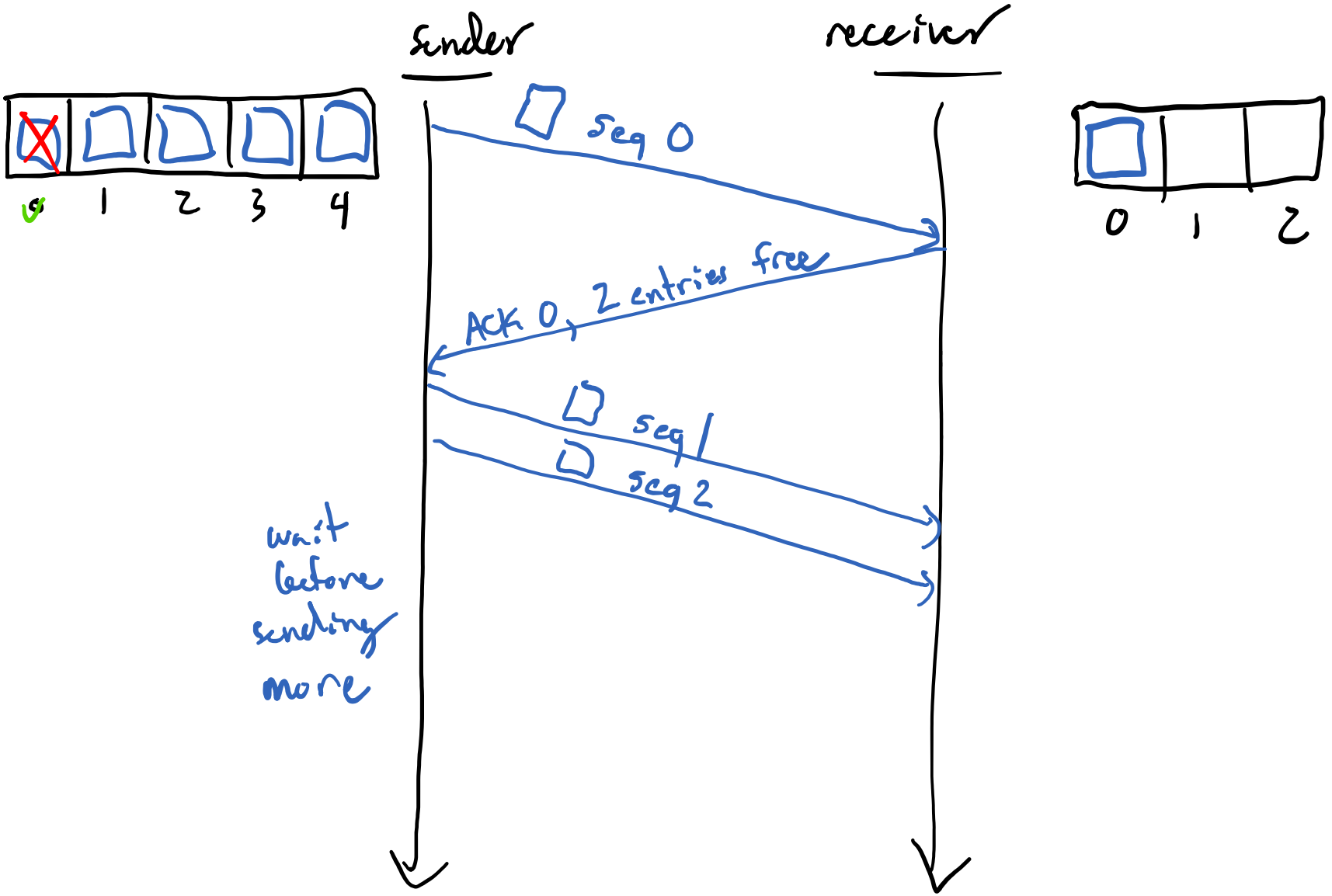
Flow control example



Flow control example



Flow control example



Exercise

- If the sender has lots of data to send, which of these situations will occur when the sliding window is working properly?

A. The receiver's queue will usually be almost full
B. The receiver's queue will usually be about half full
C. The receiver's queue will usually be almost empty

Exercise

- If the sender has lots of data to send, which of these situations will occur when the sliding window is working properly?

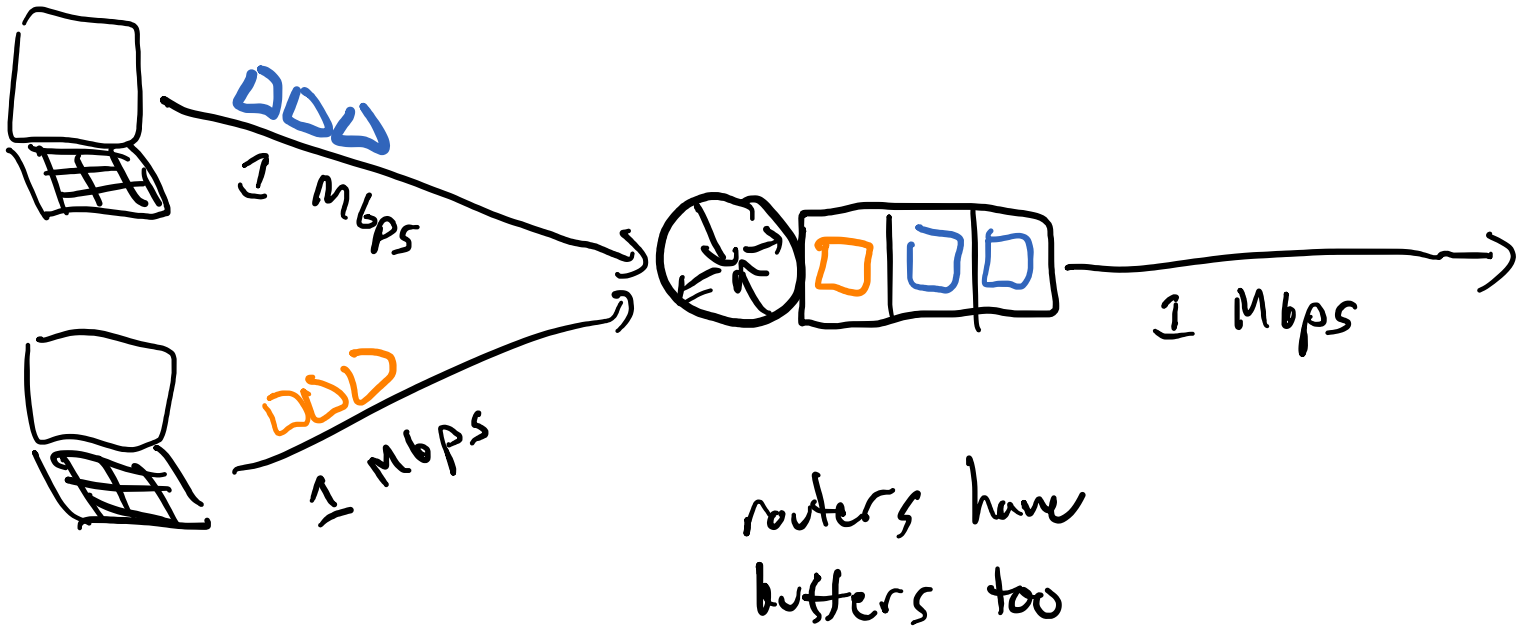
A. The receiver's queue will usually be almost full

B. The receiver's queue will usually be about half full

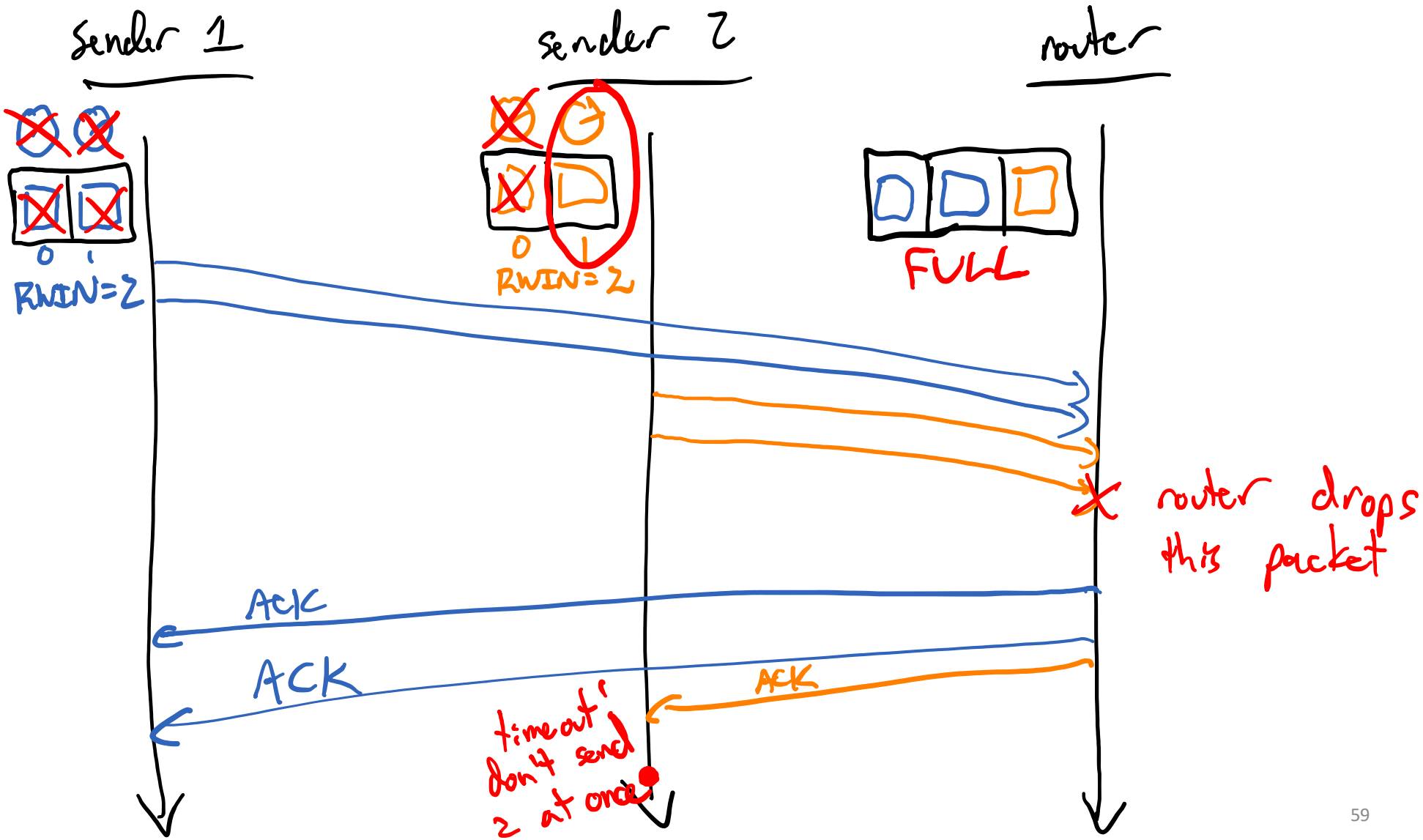
C. The receiver's queue will usually be almost empty

Congestion control

- Problem: Many senders overload a network router
  - Routers have buffers
  - Router drop packets when its buffer is full



Congestion control example



TCP congestion window

- Sender maintains a **congestion window**
  - Maximum packets awaiting acknowledgments
  - Sometimes called *CWND*

- Decrease congestion window when sender loses a packet
  - Assumes that a router dropped a packet because it was too busy
- Increase congestion window when sender receives an ACK
  - Packet got there safely, maybe there's room to send more

Flow control and congestion control

- Flow control (AKA sliding window)
  - Keep a *fast sender* from overloading a *slow receiver*
- Congestion control
  - Keep a *set of senders* from overloading the *network*

- We need both!
  - TCP flow control: receiver window (RWND)
  - TCP congestion control: congestion window (CWND)

- TCP window:
  `min(congestion_window, receiver_window)`

TCP summary

- Problem: Packets may arrive out of order
- Solution: TCP buffers reassemble packets using *sequence numbers*

- Problem: Packets may disappear
- Solution: TCP resends lost packets

- Problem: Packets may be repeated
- Solution: TCP receiver ignores duplicate sequence numbers

TCP flow and congestion control summary

- Problem: Fast sender overloads a slow receiver
- Solution: *Flow control AKA sliding window*
  - Sender limits number of packets

- Problem: Many senders overload a network router
- Solution: *Congestion control*
  - Sender limits number of packets

Agenda

- Motivation
- IP: Internet Protocol
- TCP: Transmission Control Protocol
  - Flow control and congestion control
- **UDP: User Datagram Protocol**
- Sockets
- Summary

TCP isn't good for everything

- TCP's reliability mechanism can cause packets to be late
  - "Better late than never"
- What will happen in a video chat app that uses TCP if packets 0 and 2-100 have been received but packet 1 has not?

Receiver buffer

Video chat over TCP would be terrible

- What will happen in a video chat app that uses TCP if packets 0 and 2-100 have been received but packet 1 has not?
- The application waits for packets 2-100
- Laggy video

Receiver buffer

Better never than late

- TCP is "Better late than never"
- Problem: Sometimes never is better than late
- What kinds of applications?
  - Voice chat ("Voice over IP" AKA VOIP)
  - Video chat
  - Video games, e.g., the locations of other players
  - Heartbeat messages from Workers to Main in GFS or MapReduce
  - Anything real-time

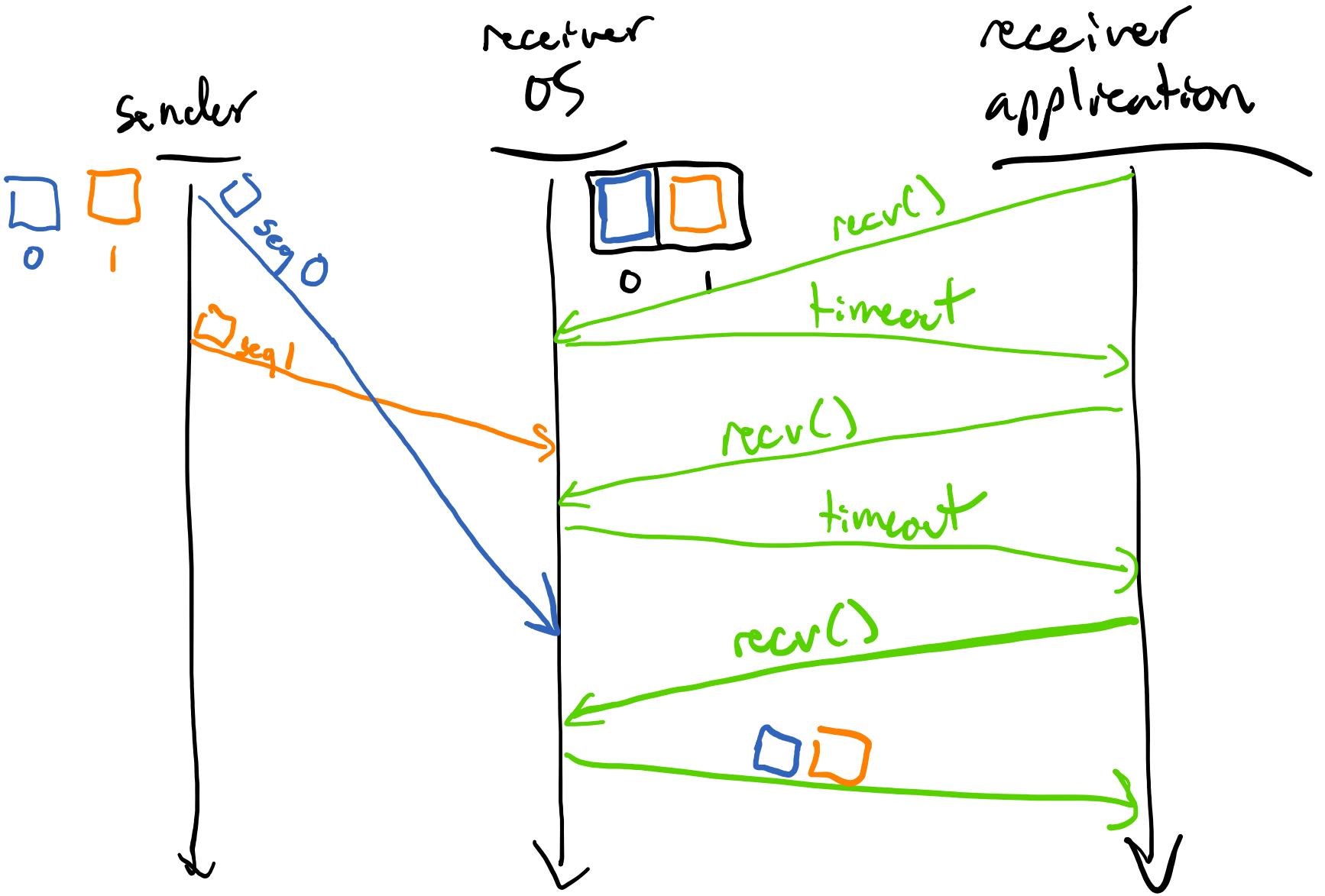
UDP

- Problem: Sometimes never is better than late
- Solution: UDP "User Datagram Protocol"
- Don't handle:
  - Packets out of order
  - Dropped packets
- Let application decide what to do
- Better for real-time applications like video chat

Agenda

- Motivation
- IP: Internet Protocol
- TCP: Transmission Control Protocol
  - Flow control and congestion control
- UDP: User Datagram Protocol
- **Sockets**
- Summary

Recall: TCP buffer



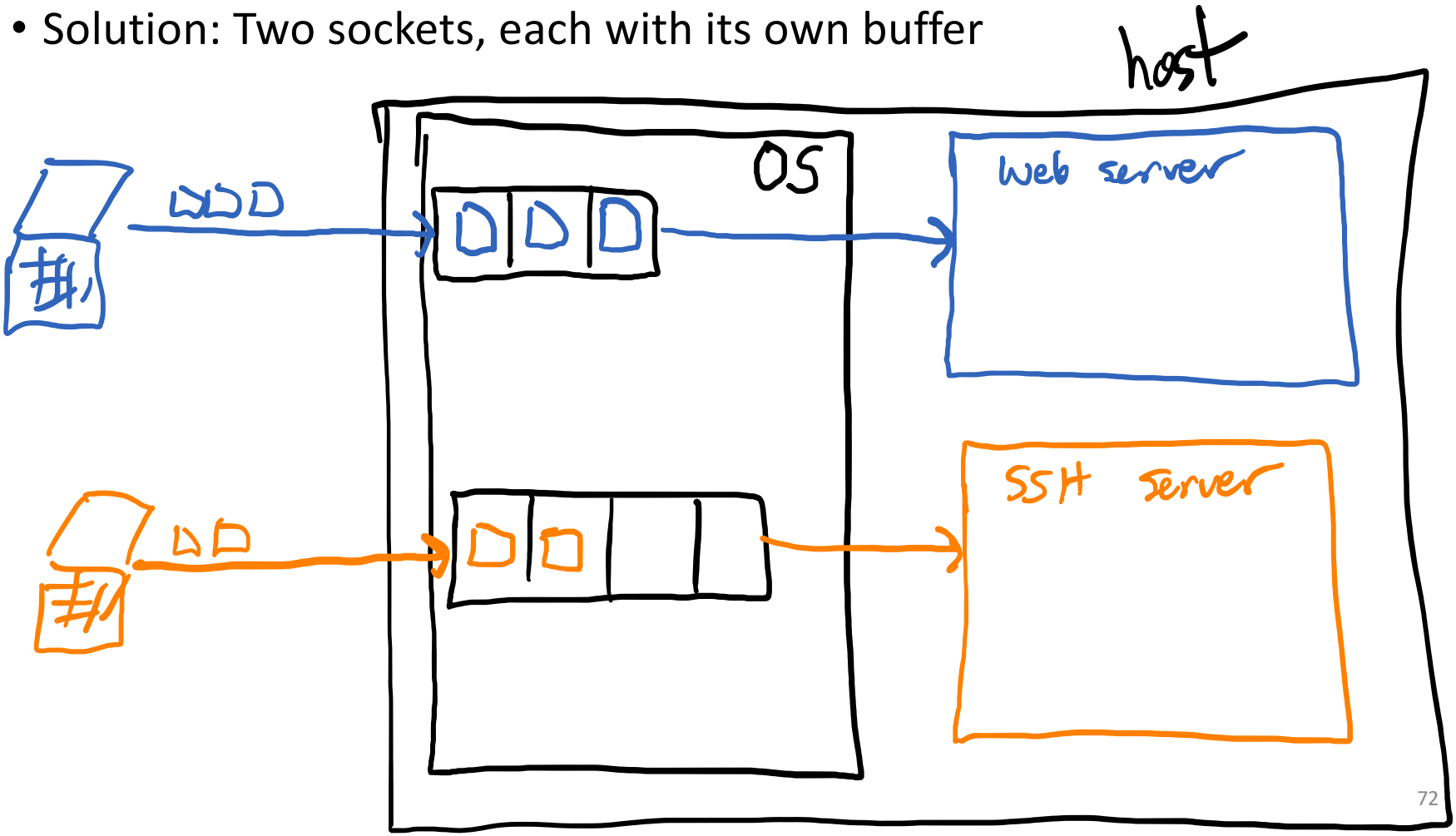
Socket API

- How does an application use TCP or UDP?
- *Socket API:* OS functions that let applications use the network

- Provides the TCP buffers that store packets
- Implements TCP sliding window

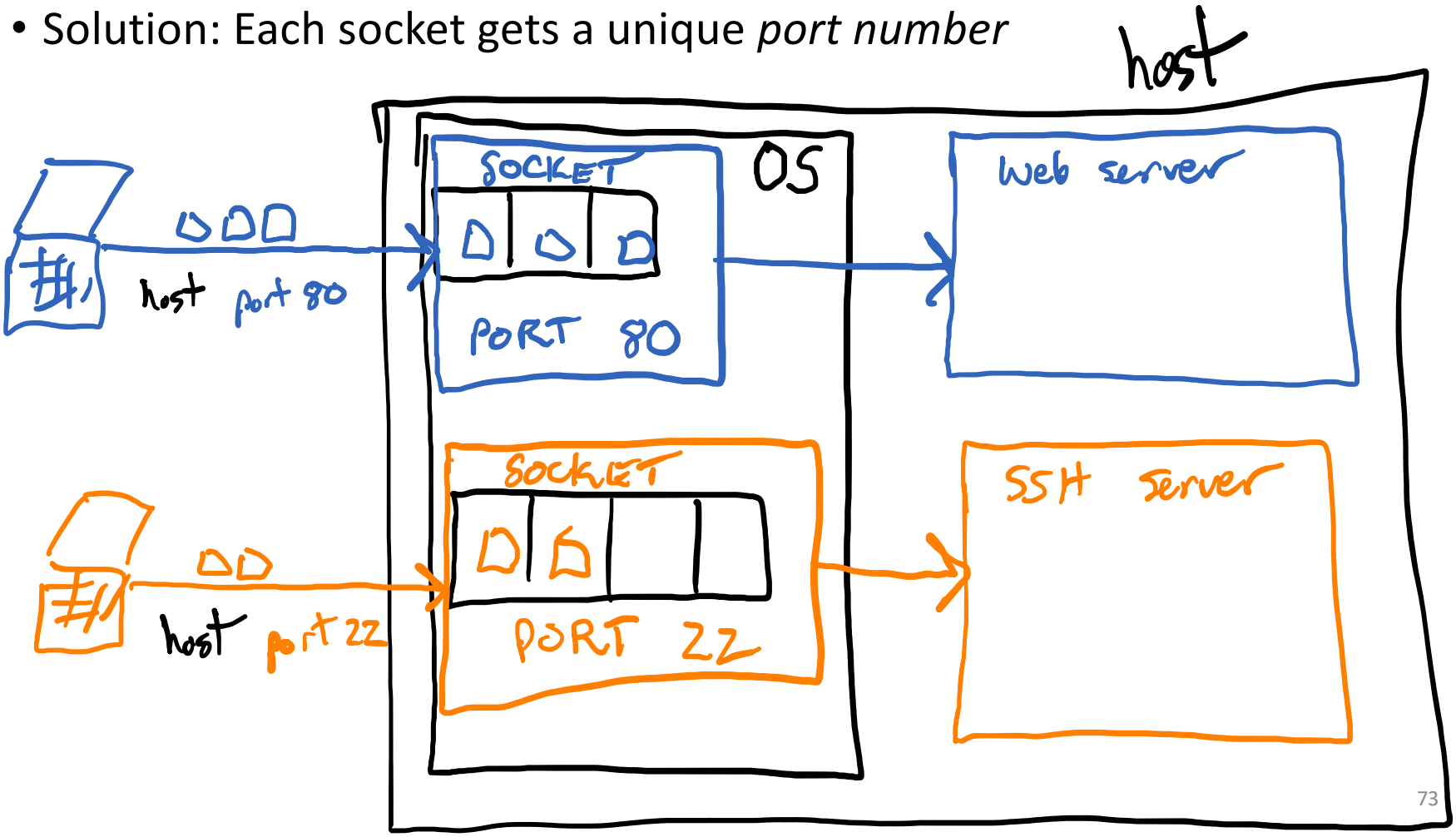
Two programs share the network

- Problem: Two programs use the network at the same time
- Solution: Two sockets, each with its own buffer



Ports

- Problem: How to tell two sockets apart?
- Solution: Each socket gets a unique *port number*



Example ports

- One server runs two programs that use the network
- Nginx listens for incoming HTTP requests
- SSHD listens for incoming SSH connections

- Both processes run on the same host using *different ports*
- HTTP is usually on port 80
  - HTTPS on port 445
- SSH is usually on port 22

Agenda

- Motivation
- IP: Internet Protocol
- TCP: Transmission Control Protocol
  - Flow control and congestion control
- UDP: User Datagram Protocol
- Sockets
- **Summary**

Networking summary

- Networking is how computers communicate
- In a distributed system, the network connects cooperating machines
  - MapReduce Manager assigns task to Worker
  - Google File System chunkserver tells Main server "I'm alive"

IP summary

- How to get a message from one computer to another computer?
  - Internet Protocol (IP)
- How to tell computers connected to the internet apart?
  - IP Address
- How are computers connected?
  - Routers
- How are long messages broken up? How do multiple computers share one network link?
  - Packets

TCP summary

- Problem: Packets may arrive out of order
- Solution: TCP buffers reassemble packets using *sequence numbers*

- Problem: Packets may disappear
- Solution: TCP resends lost packets

- Problem: Packets may be repeated
- Solution: TCP receiver ignores duplicate sequence numbers

TCP flow and congestion control summary

- Problem: Fast sender overloads a slow receiver
- Solution: *Flow control AKA sliding window*
  - Sender limits number of packets

- Problem: Many senders overload a network router
- Solution: *Congestion control*
  - Sender limits number of packets

UDP summary

- Problem: Sometimes never is better than late
- Solution: UDP "User Datagram Protocol"

- UDP lets the application decide what to do about
  - Packets out of order
  - Dropped packets

- Better for real-time applications like video chat

Sockets and ports summary

- *Socket API* provided by the operating system
- Functions that let applications use the network

- A *port* is a number that identifies a socket on one host
- Two applications on one host can both use the network on different ports