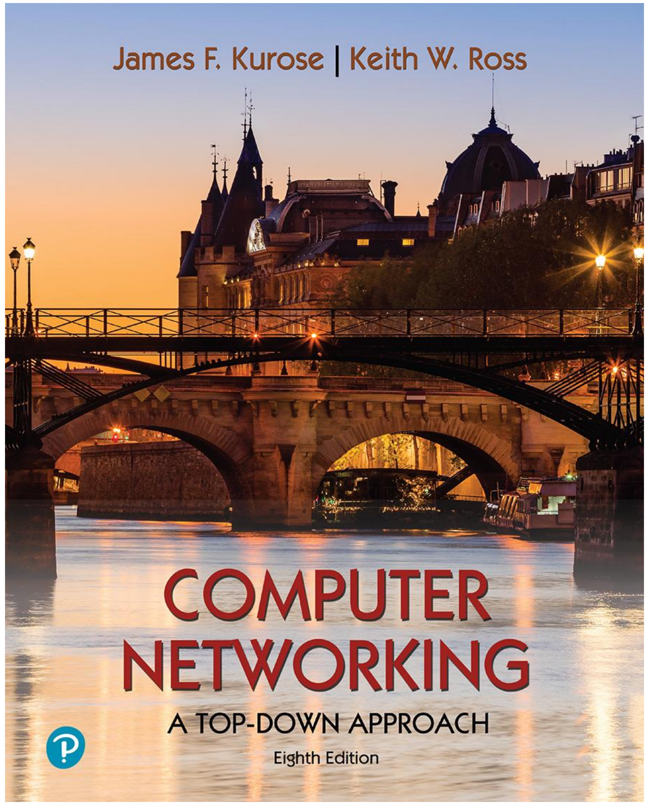
Chapter 3
Transport Layer

Yaxiong Xie

Department of Computer Science and Engineering
University at Buffalo, SUNY

Adapted from the slides of the book's authors



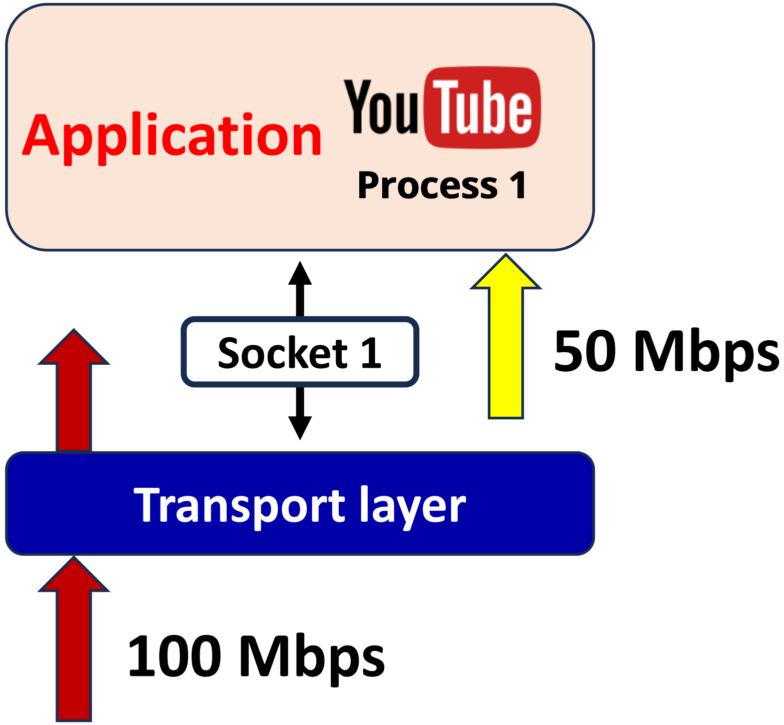
Computer Networking: A Top-Down Approach
8th edition
Jim Kurose, Keith Ross
Pearson, 2020

Chapter 3: roadmap

- Transport-layer services
- Multiplexing and demultiplexing
- Connectionless transport: UDP
- Principles of reliable data transfer
- Connection-oriented transport: TCP
  - segment structure
  - reliable data transfer
  - flow control
  - connection management
- Principles of congestion control
- TCP congestion control

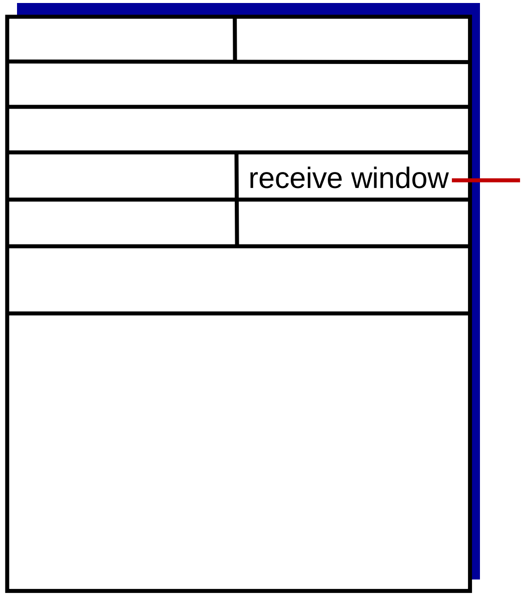
TCP flow control

Q: What happens if network layer delivers data faster than application layer removes data from socket?

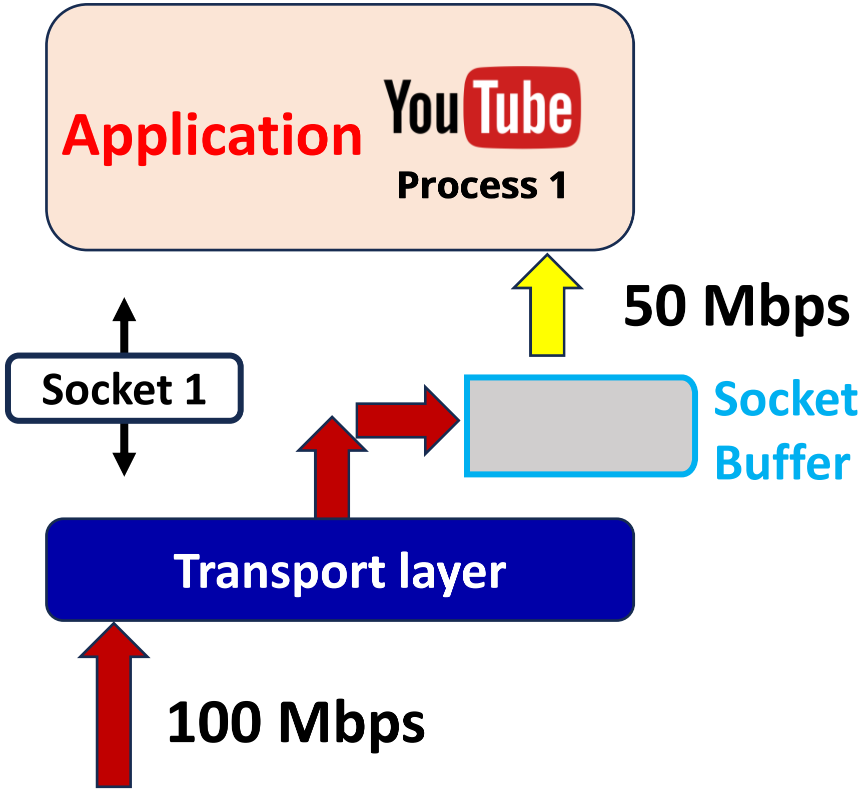


TCP flow control

Q: What happens if network layer delivers data faster than application layer removes data from socket?



flow control: # bytes receiver willing to accept

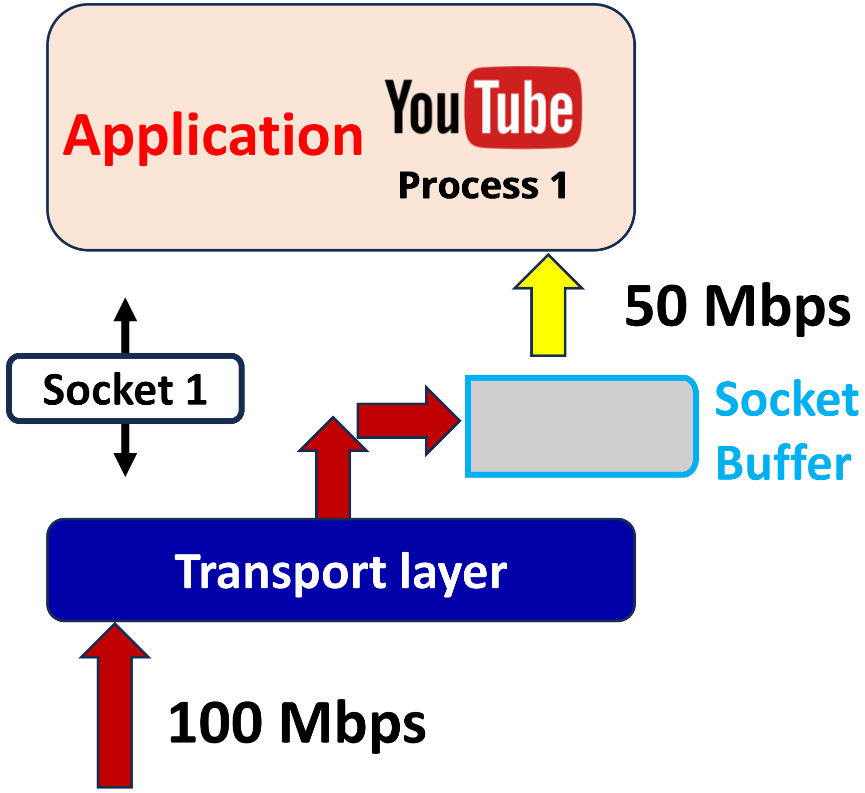


TCP flow control

Q: What happens if network layer delivers data faster than application layer removes data from socket?

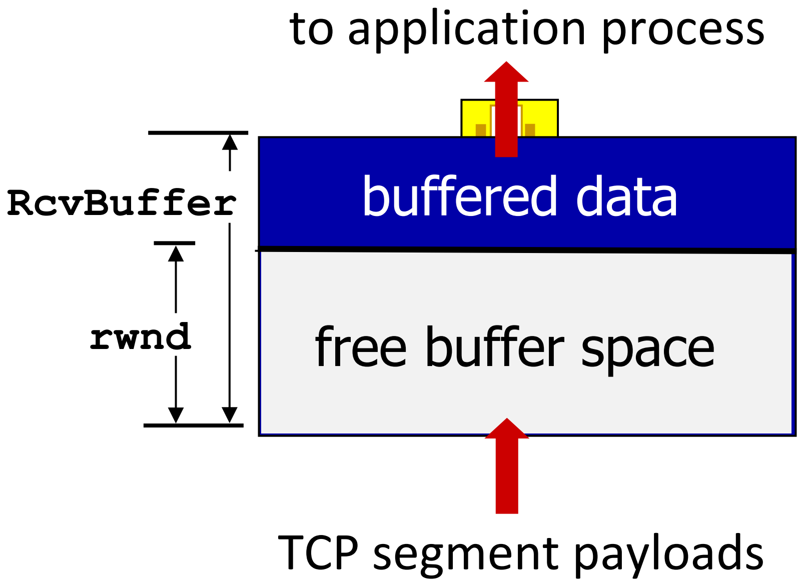
flow control

receiver controls sender, so sender won't overflow receiver's buffer by transmitting too much, too fast



TCP flow control

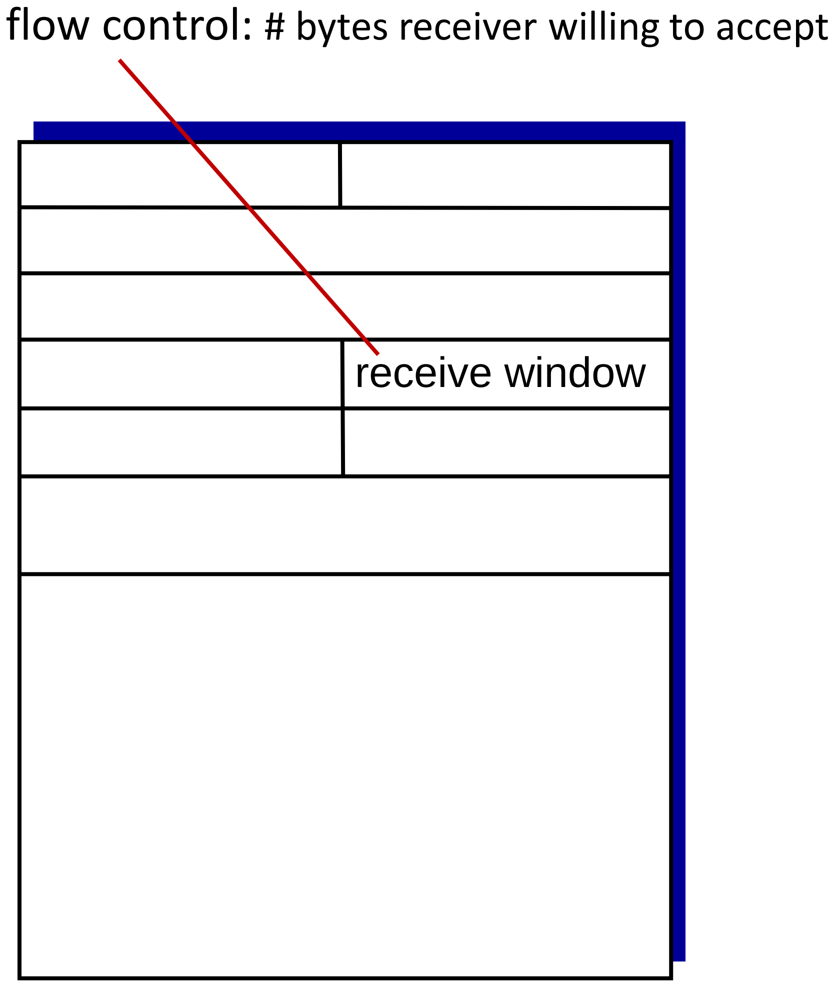
- TCP receiver "advertises" free buffer space in **rwnd** field in TCP header
  - **RcvBuffer** size set via socket options (typical default is 4096 bytes)
  - many operating systems auto-adjust **RcvBuffer**
- sender limits amount of unACKed ("in-flight") data to received **rwnd**
- guarantees receive buffer will not overflow



TCP receiver-side buffering

TCP flow control

- TCP receiver "advertises" free buffer space in **`rwnd`** field in TCP header
  - **`RcvBuffer`** size set via socket options (typical default is 4096 bytes)
  - many operating systems auto-adjust **`RcvBuffer`**
- sender limits amount of unACKed ("in-flight") data to received **`rwnd`**
- guarantees receive buffer will not overflow

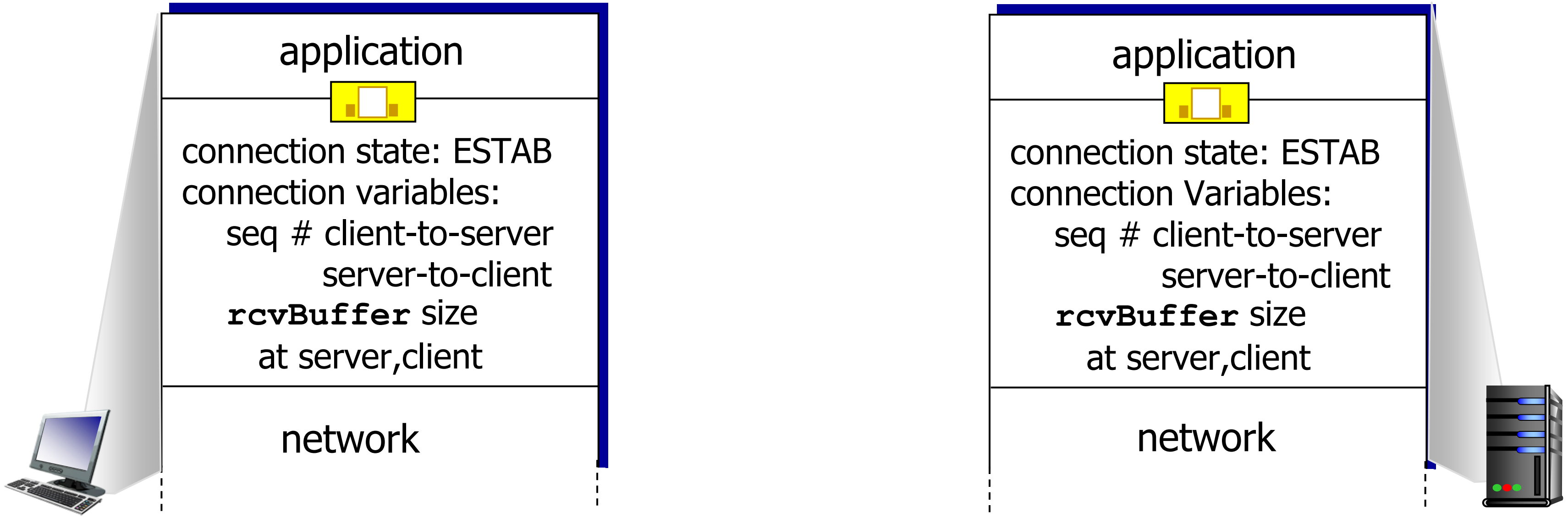


TCP segment format

TCP connection management

before exchanging data, sender/receiver “handshake”:

- agree to establish connection (each knowing the other willing to establish connection)
- agree on connection parameters (e.g., starting seq #s)

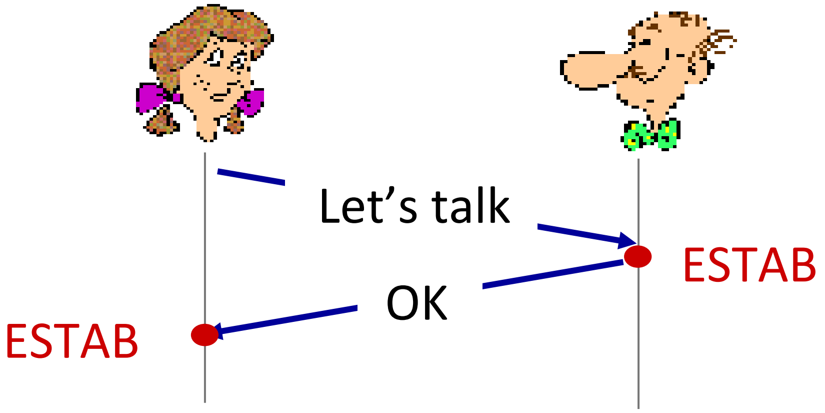


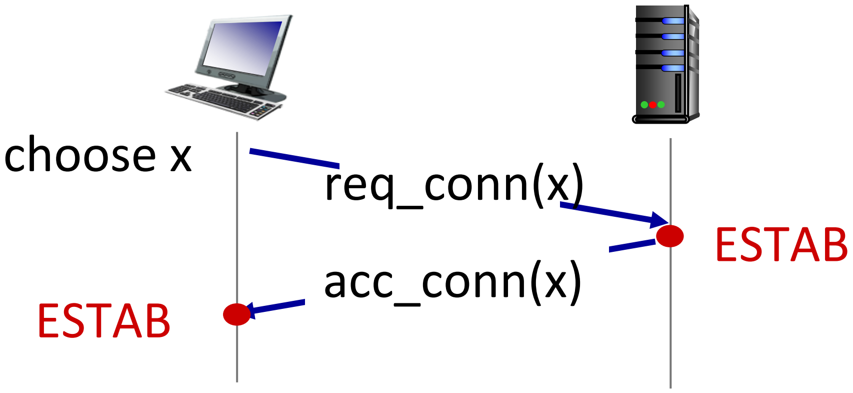
```
Socket clientSocket =
  newSocket("hostname","port number");
```

```
Socket connectionSocket =
  welcomeSocket.accept();
```

Agreeing to establish a connection

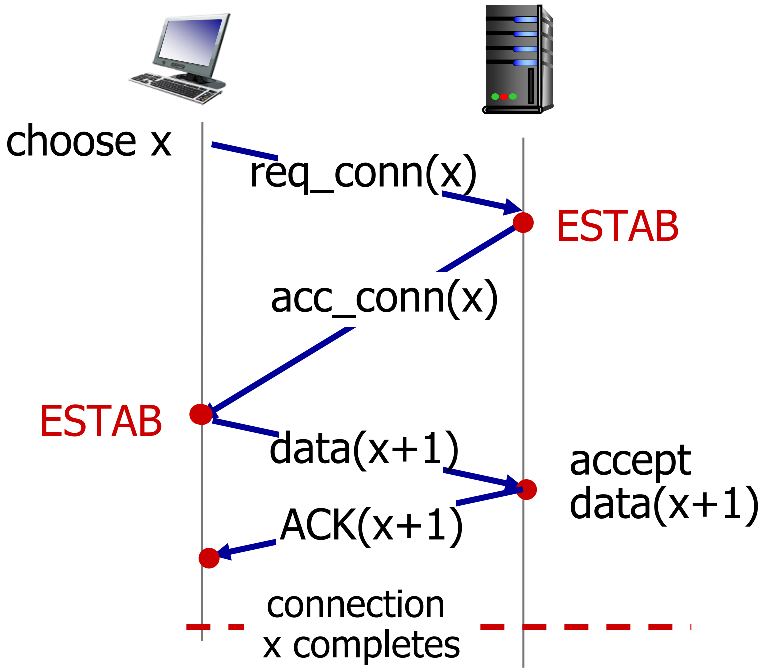
2-way handshake:





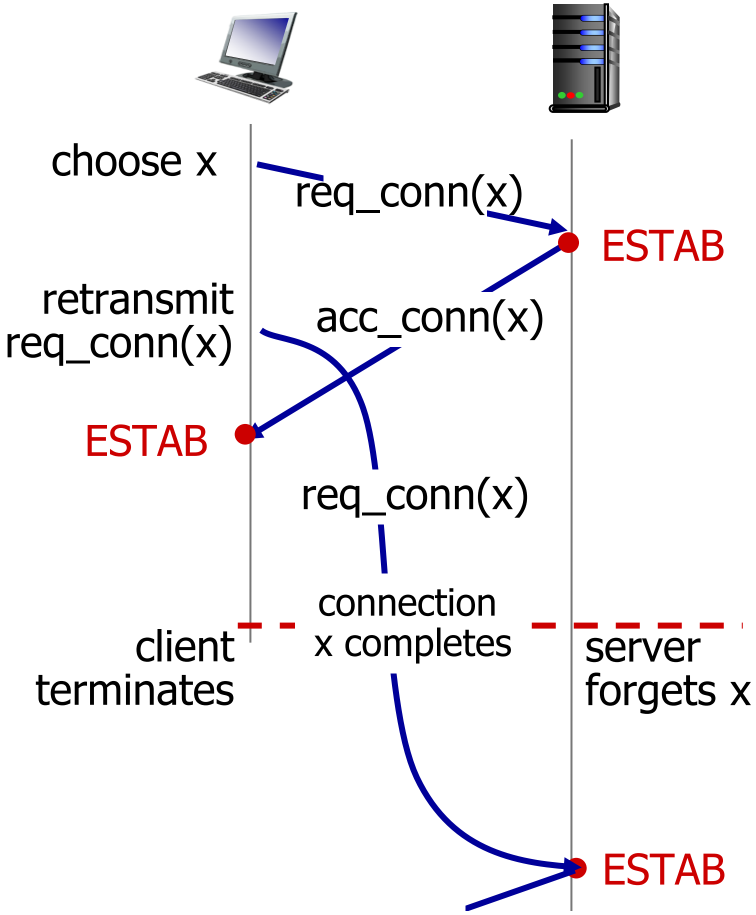
Q: will 2-way handshake always work in network?

2-way handshake scenarios



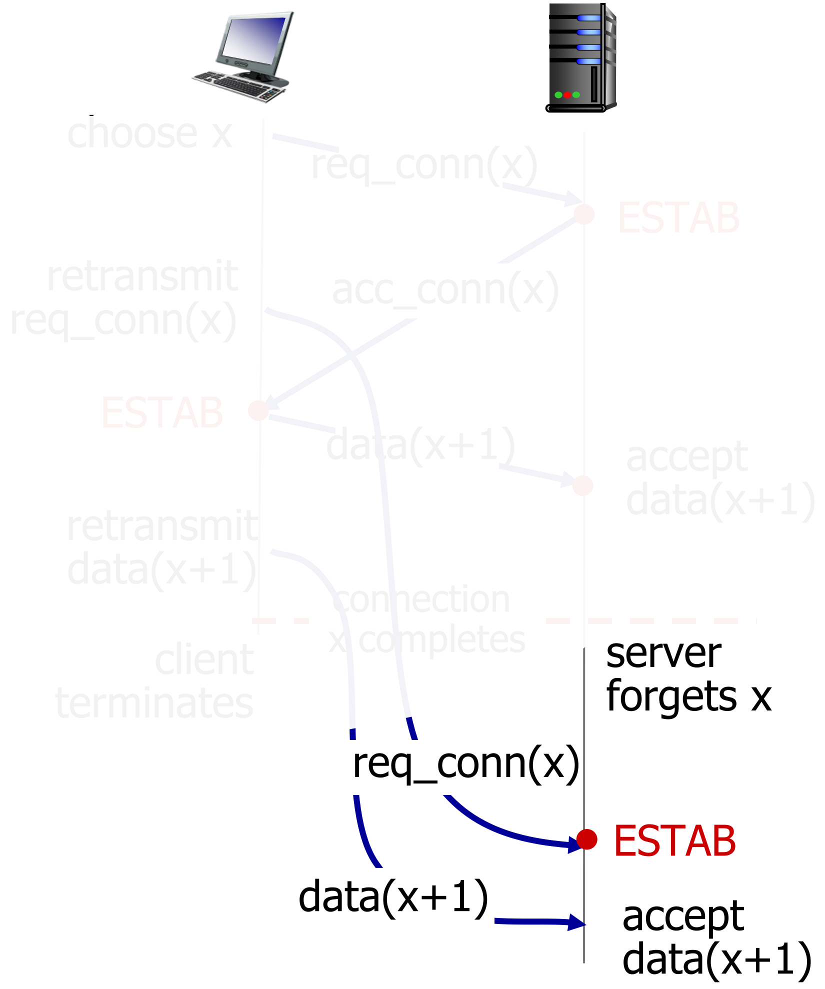
No problem! ✓

2-way handshake scenarios



✗ Problem: half open connection! (no client)

2-way handshake scenarios



❌ Problem: dup data accepted!

TCP 3-way h

Client state

```
clientSocket = socket(AF_INET, SOCK
```

LISTEN

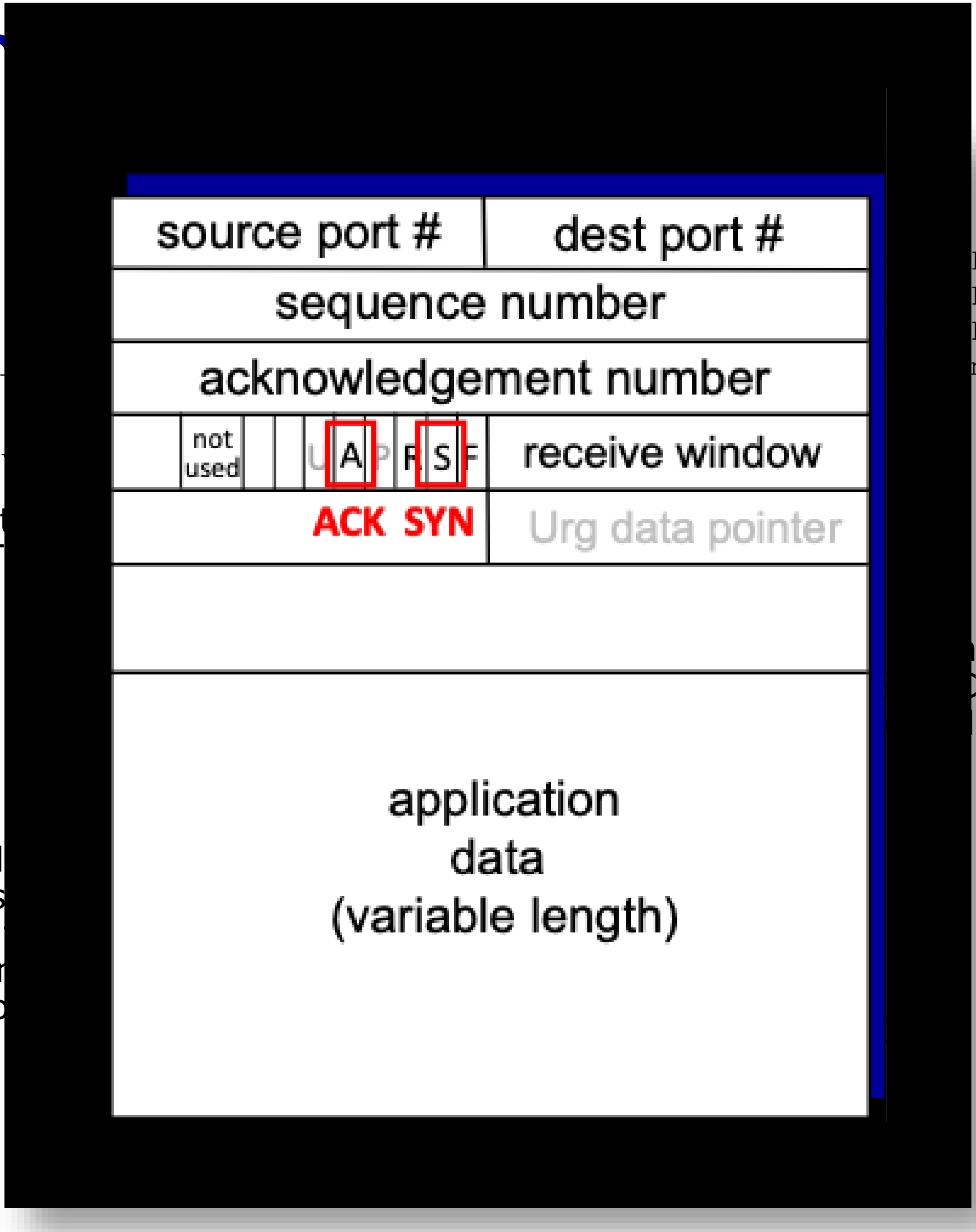
```
clientSocket.connect((serverName,ser
```

choose init
send T

SYNSENT

ESTAB

received
indicates s
send ACK
this segment
client-to



Server state

```
ket = socket(AF_INET,SOCK_STREAM)
ket.bind(('',serverPort))
ket.listen(1)
nSocket, addr = serverSocket.accept()
```

LISTEN

um, y
K

SYN RCVD

live

ESTAB

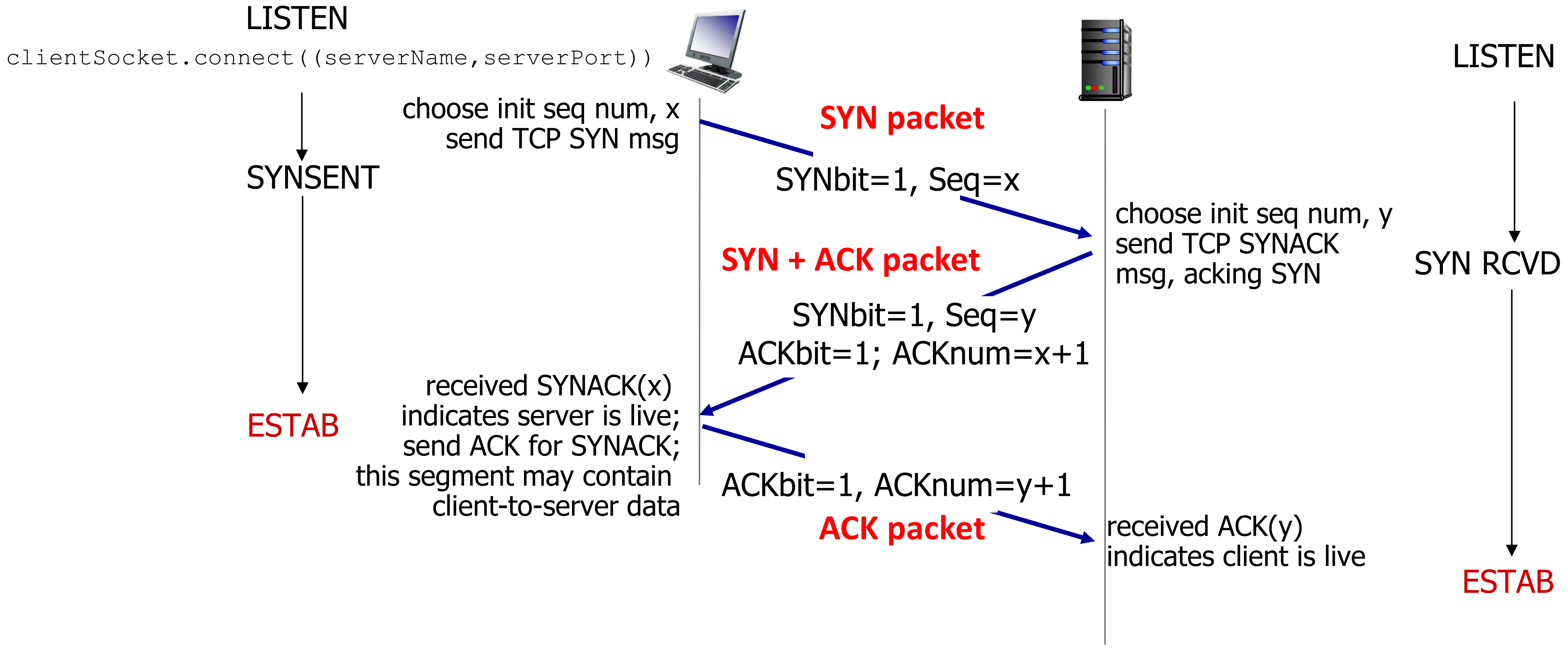
TCP 3-way handshake

Client state

```
clientSocket = socket(AF_INET, SOCK_STREAM)
```

Server state

```
serverSocket = socket(AF_INET,SOCK_STREAM)
serverSocket.bind((‘’,serverPort))
serverSocket.listen(1)
connectionSocket, addr = serverSocket.accept()
```



LISTEN

```
clientSocket.connect((serverName,serverPort))
```

LISTEN

choose init seq num, x
send TCP SYN msg

SYNSENT

SYN packet

SYNbit=1, Seq=x

choose init seq num, y
send TCP SYNACK
msg, acking SYN

SYN RCVD

SYN + ACK packet

SYNbit=1, Seq=y
ACKbit=1; ACKnum=x+1

ESTAB

received SYNACK(x)
indicates server is live;
send ACK for SYNACK;
this segment may contain
client-to-server data

ACKbit=1, ACKnum=y+1

ACK packet

received ACK(y)
indicates client is live

ESTAB

Closing a TCP connection

- client, server each close their side of connection
  - send TCP segment with FIN bit = 1
- respond to received FIN with ACK
  - on receiving FIN, ACK can be combined with own FIN
- simultaneous FIN exchanges can be handled

Chapter 3: roadmap

- Transport-layer services
- Multiplexing and demultiplexing
- Connectionless transport: UDP
- Principles of reliable data transfer
- Connection-oriented transport: TCP
- Principles of congestion control
- TCP congestion control
- Evolution of transport-layer functionality

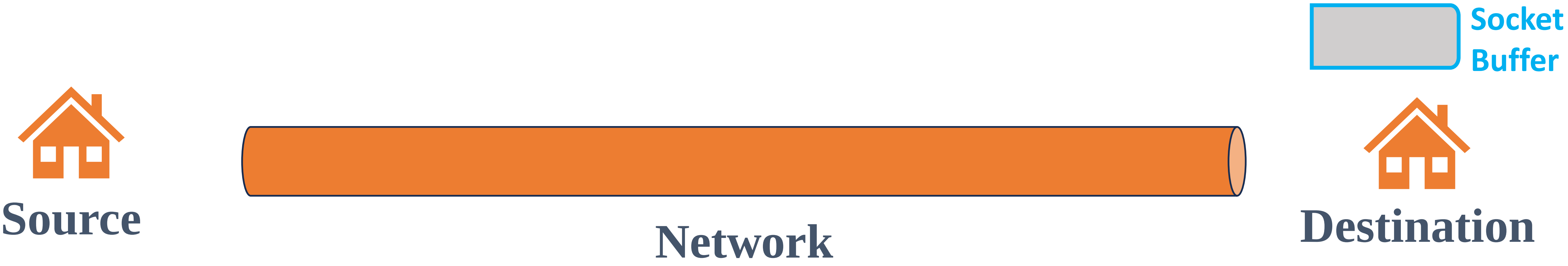
Principles of congestion control

Flow Control:

- Preventing the **sender** overwhelm the **receiver** (the socket buffer)

Congestion Control:

- Preventing the **sender** overwhelm the **network**



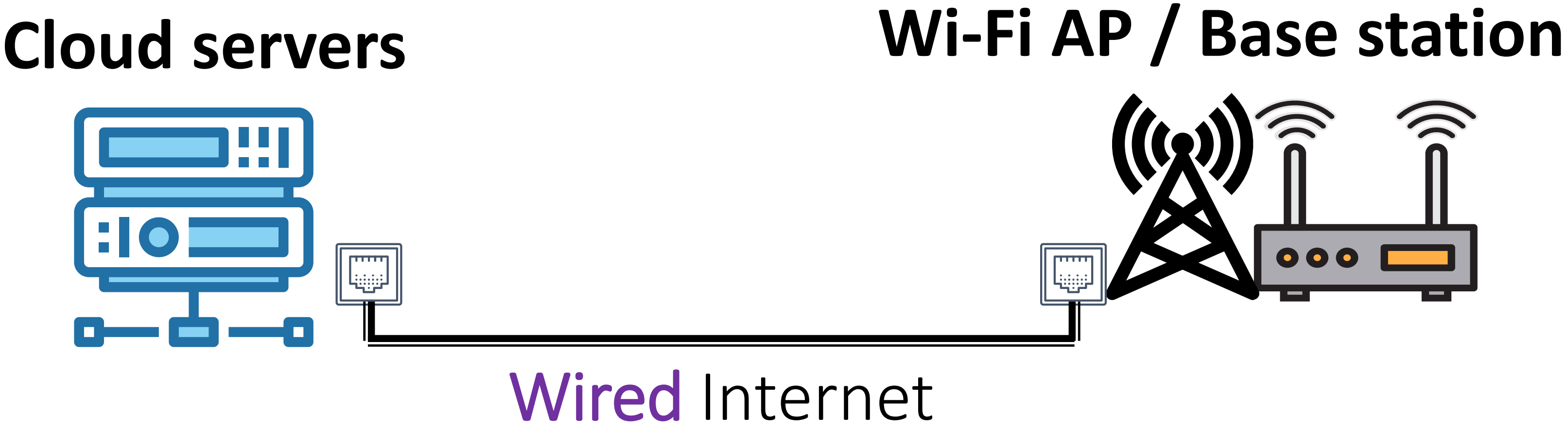
Principles of congestion control

Network Congestion:

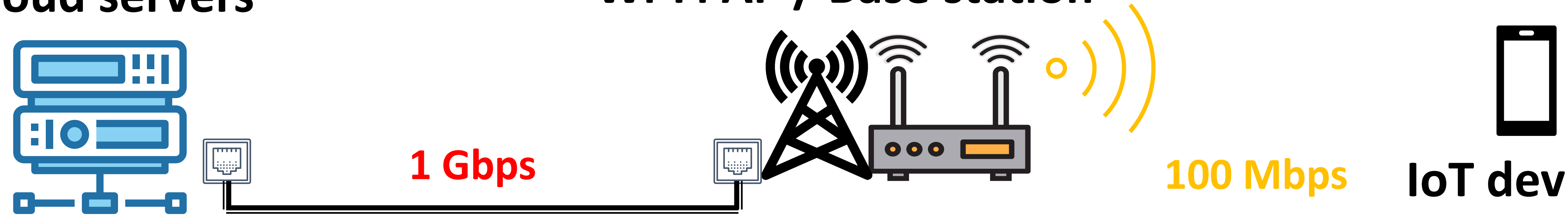
- informally: “too many sources sending too much data too fast for *network* to handle”
- manifestations:
  - long delays (queueing in router buffers)
  - packet loss (buffer overflow at routers)
- a top-10 problem in computer network!

Socket
Buffer
Source
Network
Destination

A wireless connection consists a wired Internet hop and a wireless last hop



A wireless connection consists a wired Internet hop and a wireless last hop



Cloud servers

Wi-Fi AP / Base station

IoT devices

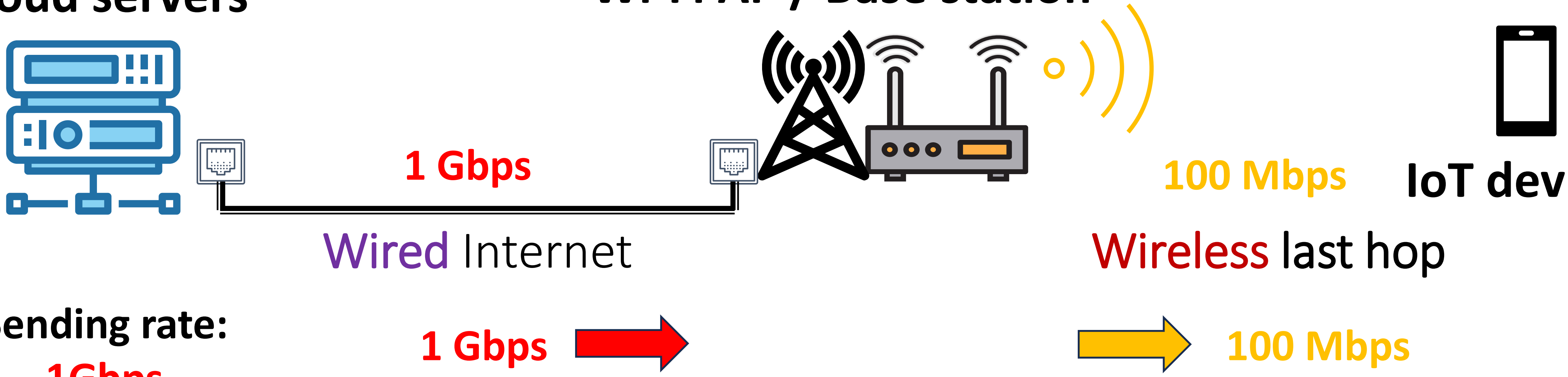
1 Gbps

Wired Internet

100 Mbps

Wireless last hop

A wireless connection consists a wired Internet hop and a wireless last hop



Cloud servers

Wi-Fi AP / Base station

1 Gbps

Wired Internet

100 Mbps

IoT devices

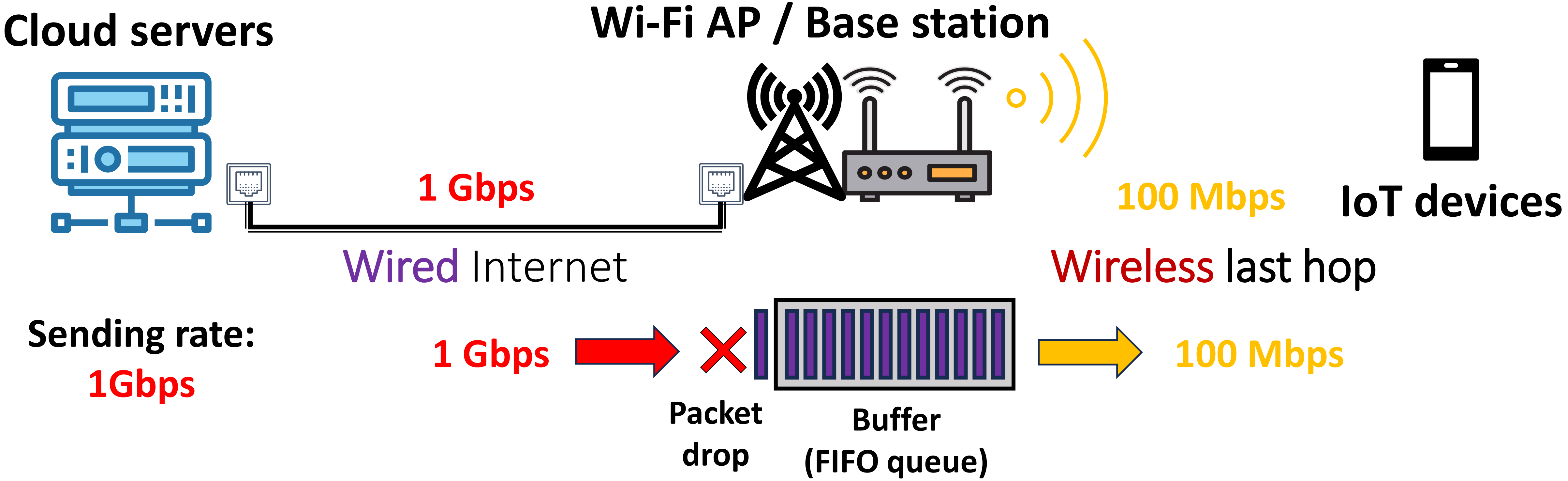
Wireless last hop

Sending rate:
1Gbps

1 Gbps

100 Mbps

A wireless connection consists a wired Internet hop and a wireless last hop



Cloud servers

Wi-Fi AP / Base station

1 Gbps

100 Mbps

IoT devices

Wired Internet

Wireless last hop

Sending rate:
1Gbps

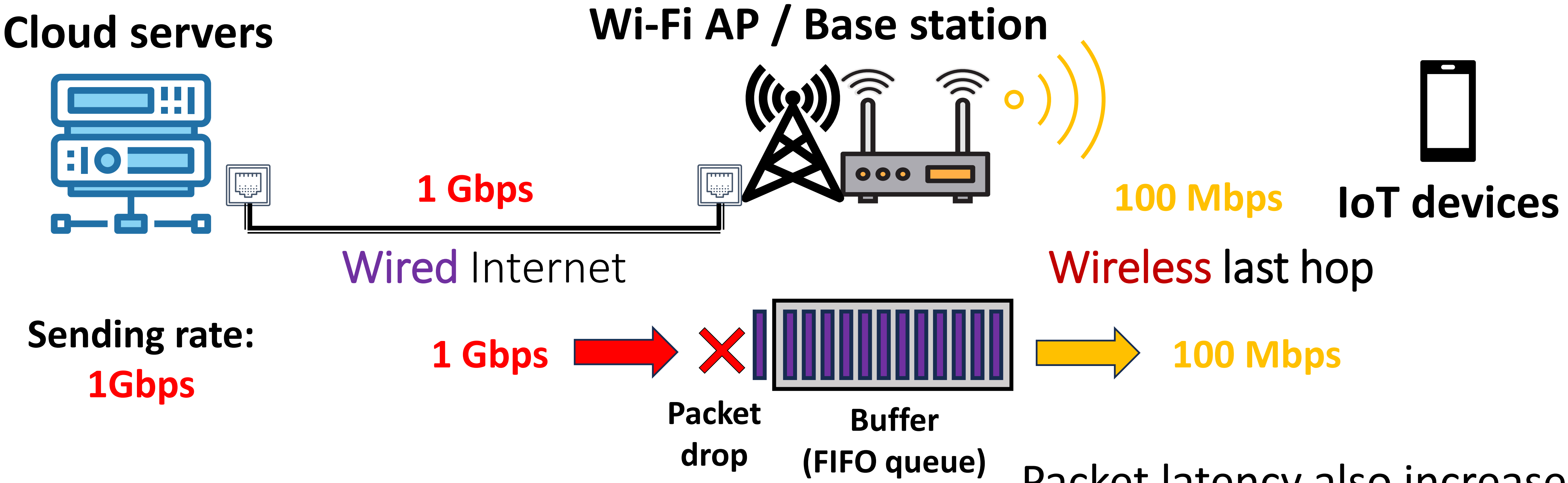
1 Gbps

100 Mbps

Packet
drop

Buffer
(FIFO queue)

A wireless connection consists a wired Internet hop and a wireless last hop



Cloud servers

Wi-Fi AP / Base station

1 Gbps

Wired Internet

100 Mbps

IoT devices

Wireless last hop

Sending rate: 1Gbps

1 Gbps

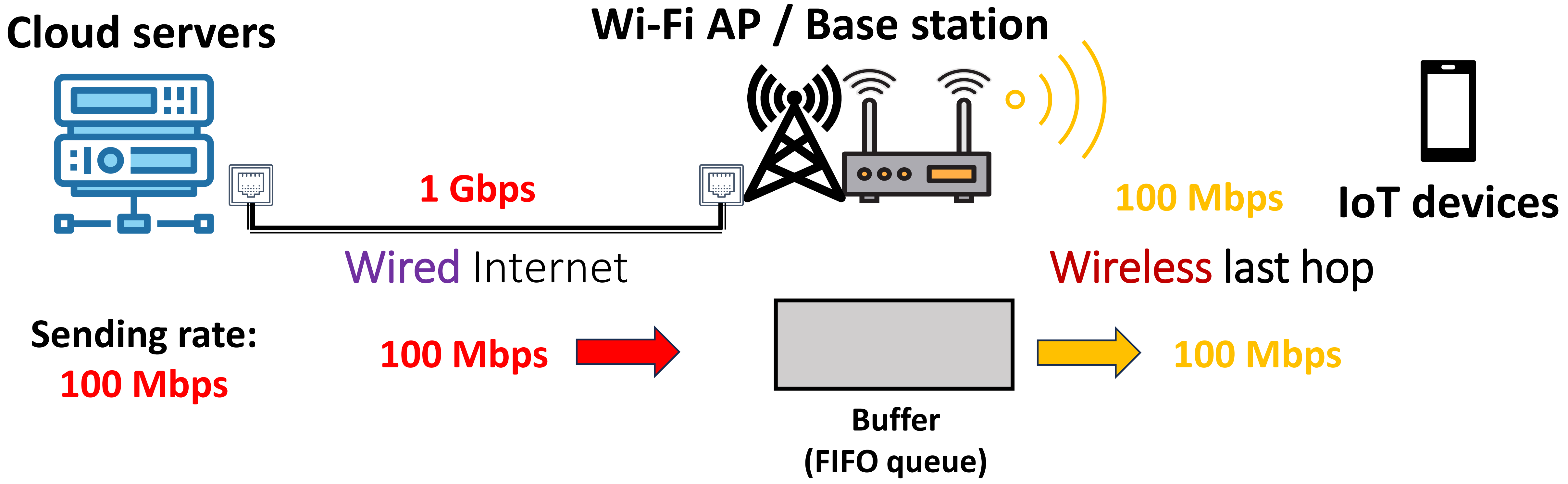
Packet drop

Buffer (FIFO queue)

100 Mbps

Packet latency also increases because of the packet queuing inside the buffer

Solution: congestion control



Cloud servers

Wi-Fi AP / Base station

IoT devices

1 Gbps

Wired Internet

100 Mbps

Wireless last hop

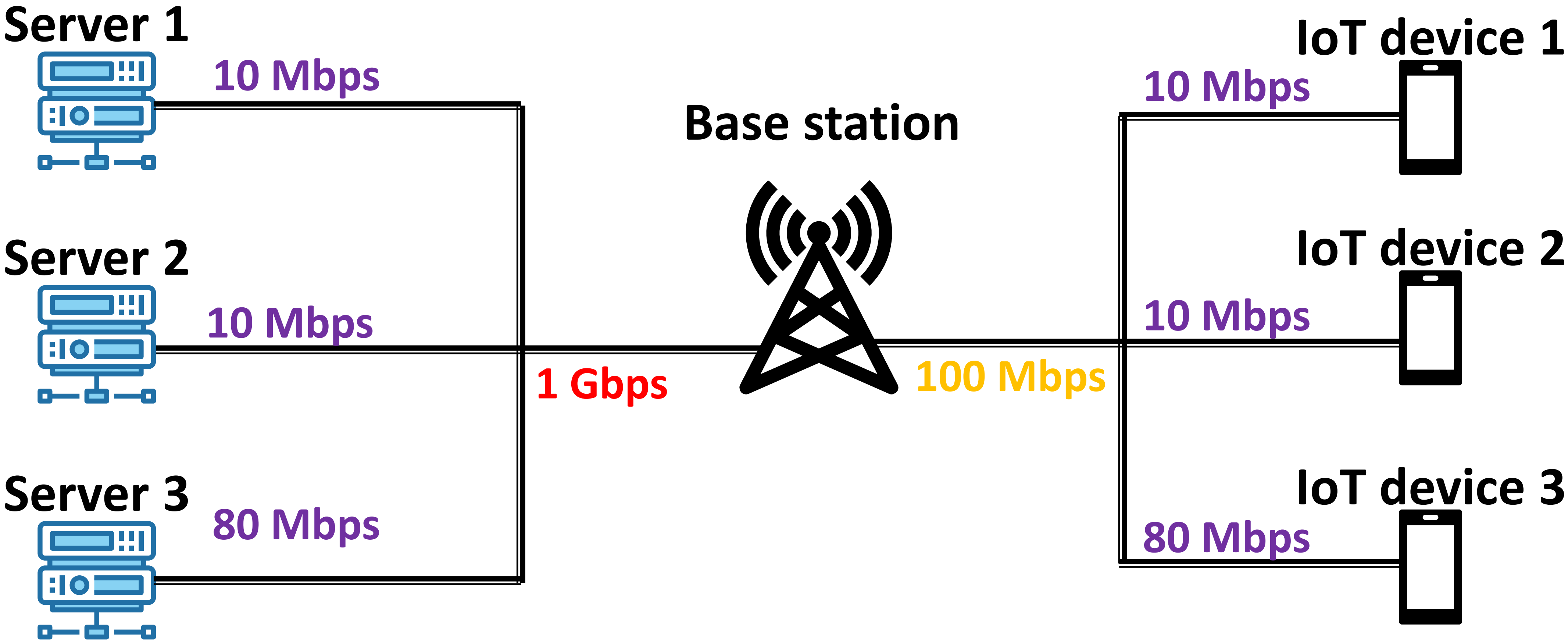
Sending rate:
100 Mbps

100 Mbps

Buffer
(FIFO queue)

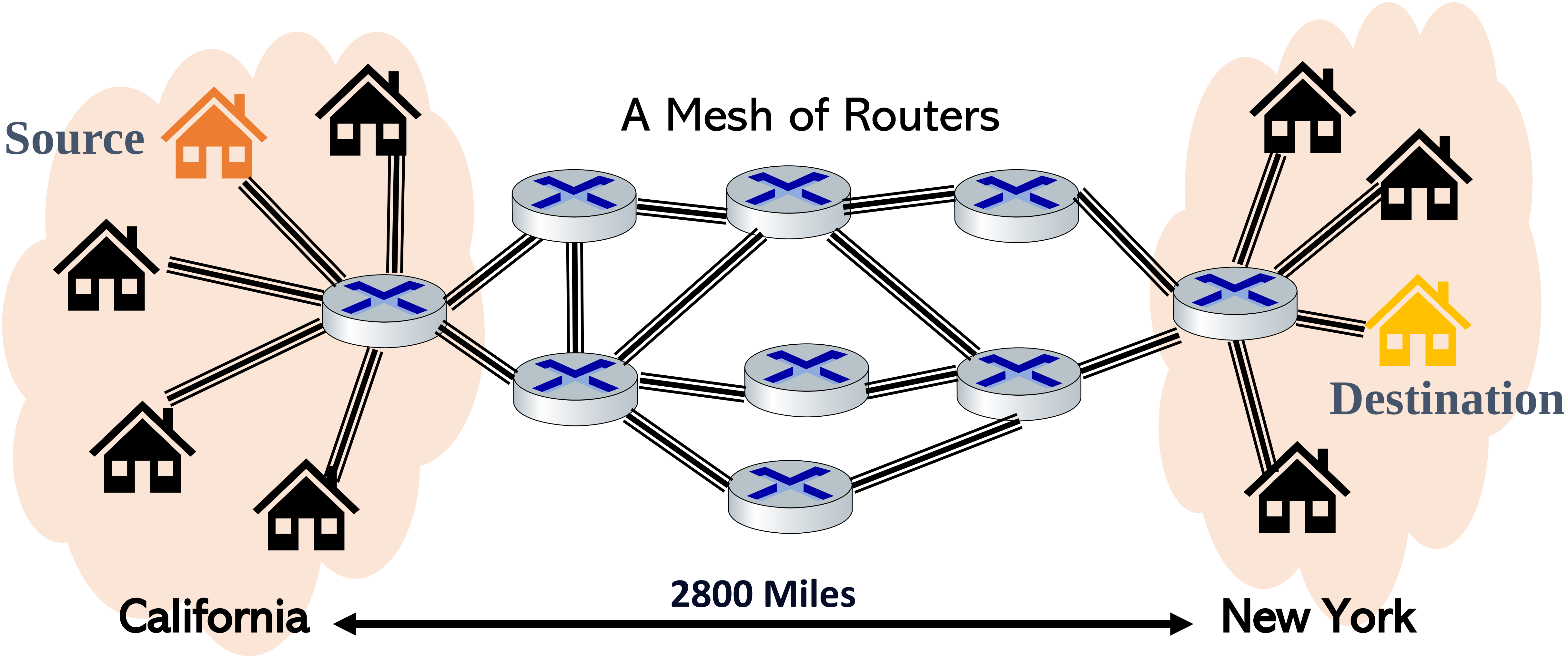
100 Mbps

Solution: congestion control



Congestion control must guarantee fairness between connections

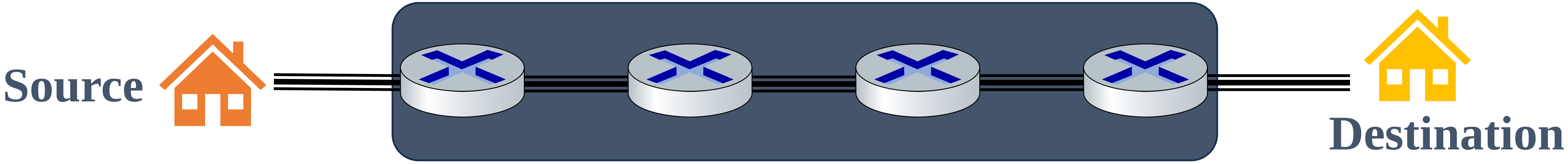
Solution: congestion control in practice



Approaches towards congestion control

End-end congestion control:

- no explicit feedback from network
- congestion *inferred* from observed loss, delay
- approach taken by TCP

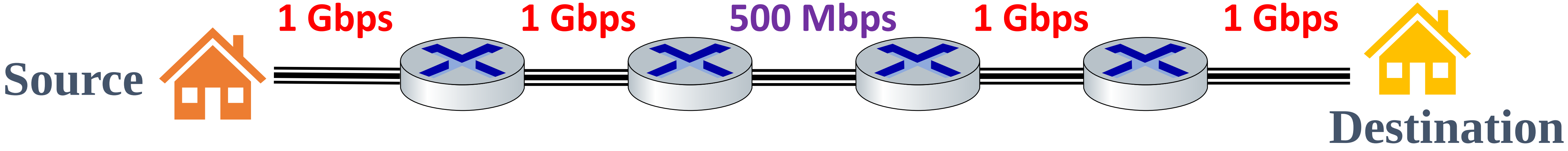


Treating network as a black box

Approaches towards congestion control

Network-assisted congestion control:

- routers provide *direct* feedback to sending/receiving hosts with flows passing through congested router



Treating network as a black box