Chapter 3 outline

3.1 transport-layer services

3.2 multiplexing and demultiplexing

3.3 connectionless transport: UDP

3.4 principles of reliable data transfer

3.5 connection-oriented transport: TCP
- segment structure
- reliable data transfer
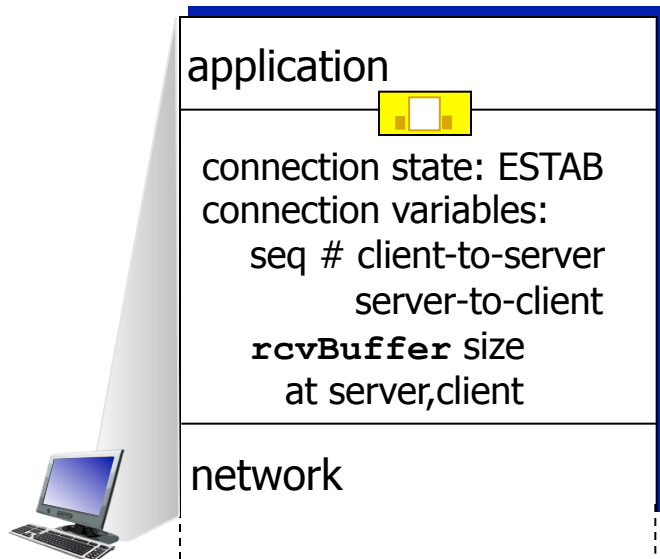- flow control
- connection management

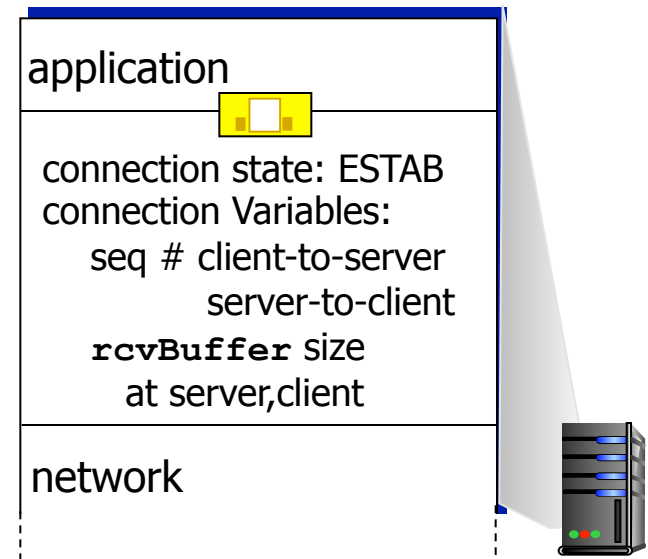3.6 principles of congestion control

3.7 TCP congestion control

Connection Management

before exchanging data, sender/receiver "handshake":

- agree to establish connection (each knowing the other willing to establish connection)
- agree on connection parameters

application

connection state: ESTAB
connection variables:
  seq # client-to-server
     server-to-client
  rcvBuffer size
    at server,client

network

application

connection state: ESTAB
connection Variables:
  seq # client-to-server
     server-to-client
  rcvBuffer size
    at server,client

network

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

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

Agreeing to establish a connection

2-way handshake:

Let's talk
OK
ESTAB
ESTAB
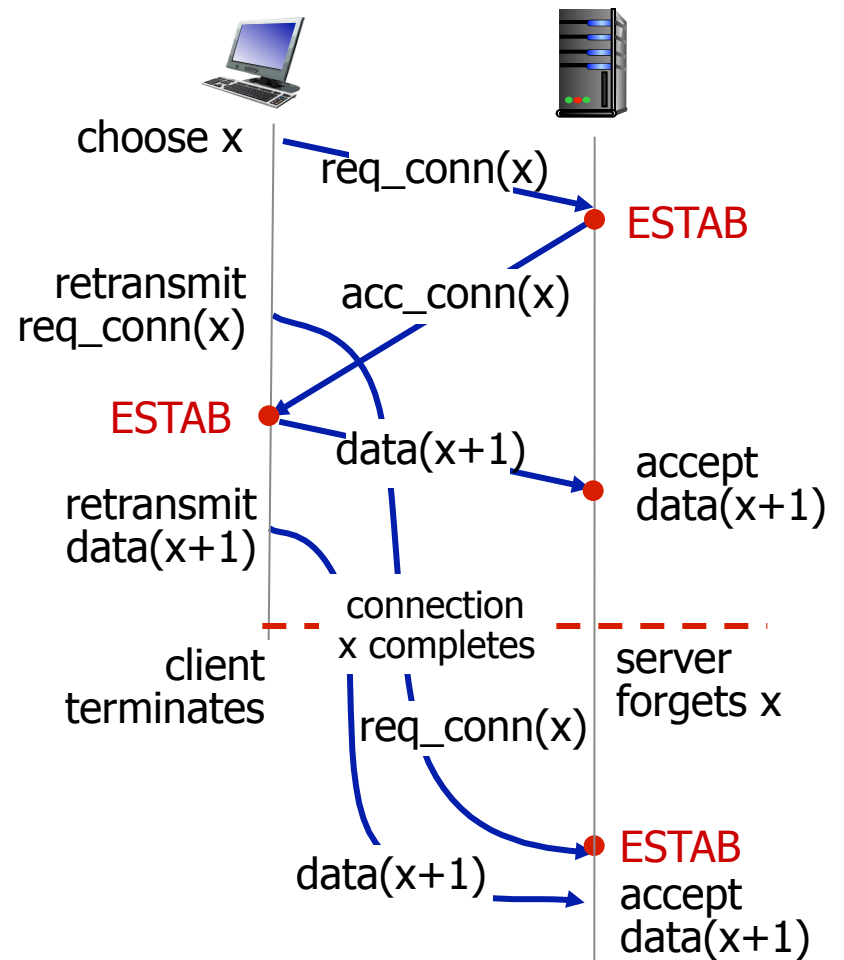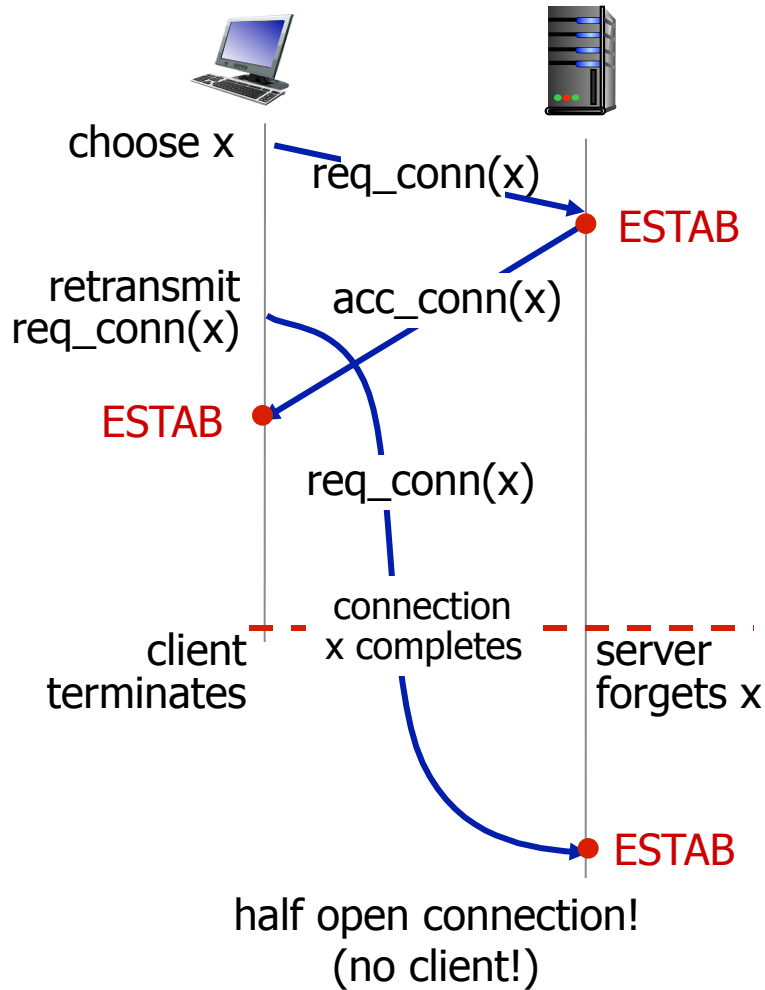
choose x
req_conn(x)
acc_conn(x)
ESTAB
ESTAB
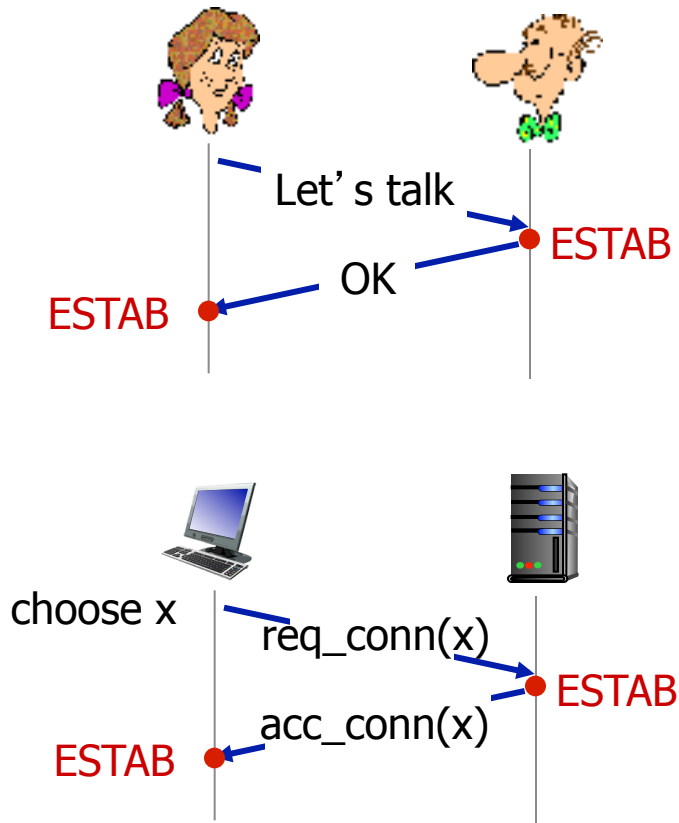
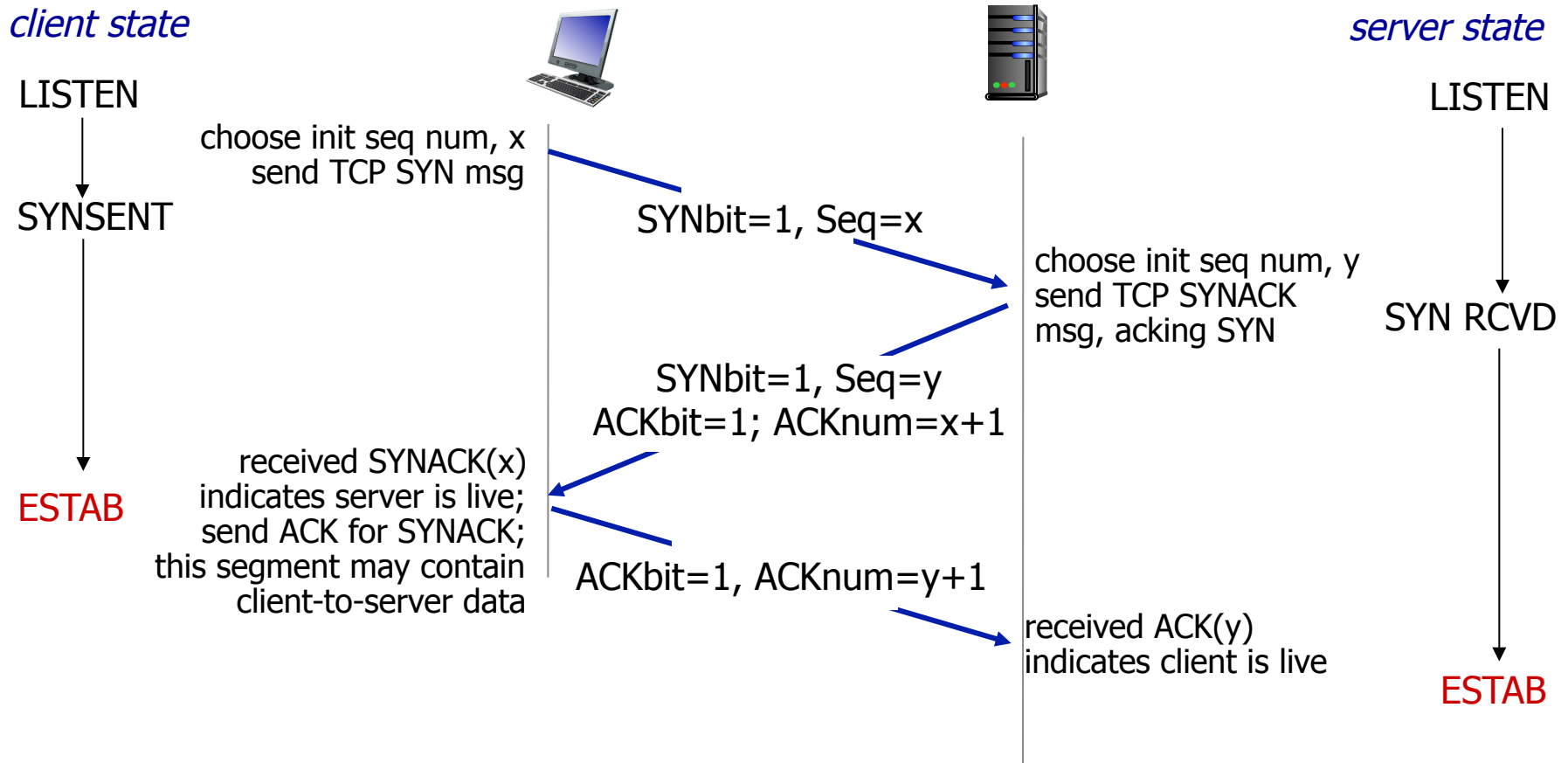*Q:* will 2-way handshake always work in network?

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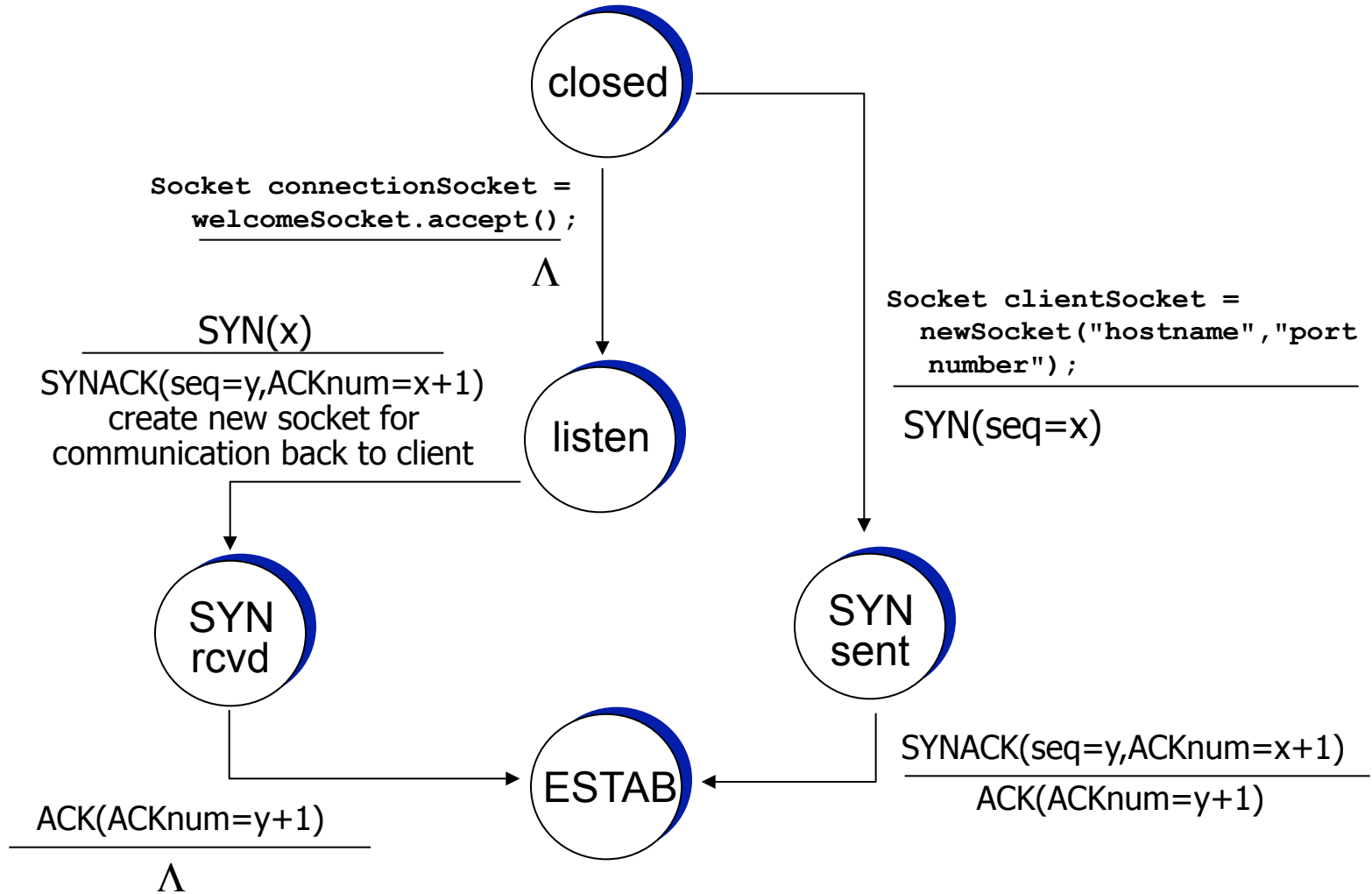
Agreeing to establish a connection

2-way handshake failure scenarios:

Left diagram:

choose x → req_conn(x) → ESTAB

retransmit req_conn(x) ← acc_conn(x)

ESTAB

req_conn(x)

client terminates — connection x completes — server forgets x

ESTAB

half open connection!
(no client!)

Right diagram:

choose x → req_conn(x) → ESTAB

retransmit req_conn(x) ← acc_conn(x)

ESTAB

data(x+1) → accept data(x+1)

retransmit data(x+1)

client terminates — connection x completes — server forgets x

req_conn(x)

data(x+1) → ESTAB accept data(x+1)

TCP 3-way handshake

client state

LISTEN

SYNSENT

ESTAB

server state

LISTEN

SYN RCVD

ESTAB

choose init seq num, x
send TCP SYN msg

SYNbit=1, Seq=x

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

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

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

ACKbit=1, ACKnum=y+1

received ACK(y)
indicates client is live

TCP 3-way handshake: FSM

closed

Socket connectionSocket =
    welcomeSocket.accept();
————————————————————
                Λ

SYN(x)
————————————————————
SYNACK(seq=y,ACKnum=x+1)
create new socket for
communication back to client

listen

Socket clientSocket =
  newSocket("hostname","port
  number");
————————————————————
SYN(seq=x)

SYN
rcvd

SYN
sent

ACK(ACKnum=y+1)
————————————————————
        Λ

ESTAB

SYNACK(seq=y,ACKnum=x+1)
————————————————————
   ACK(ACKnum=y+1)

TCP: closing a 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

TCP: closing a connection

client state

server state

ESTAB

ESTAB

`clientSocket.close()`

FIN_WAIT_1   can no longer
              send but can
              receive data

FINbit=1, seq=x

CLOSE_WAIT

ACKbit=1; ACKnum=x+1

can still
send data

FIN_WAIT_2   wait for server
              close

LAST_ACK

TIMED_WAIT

FINbit=1, seq=y

can no longer
send data

ACKbit=1; ACKnum=y+1

CLOSED

timed wait
for 2*max
segment lifetime

CLOSED

Chapter 3 outline

Principles of congestion control

congestion:
- informally: "too many sources sending too much data too fast for *network* to handle"
- different from flow control!
- manifestations:
  - lost packets (buffer overflow at routers)
  - long delays (queueing in router buffers)
- a top-10 problem!
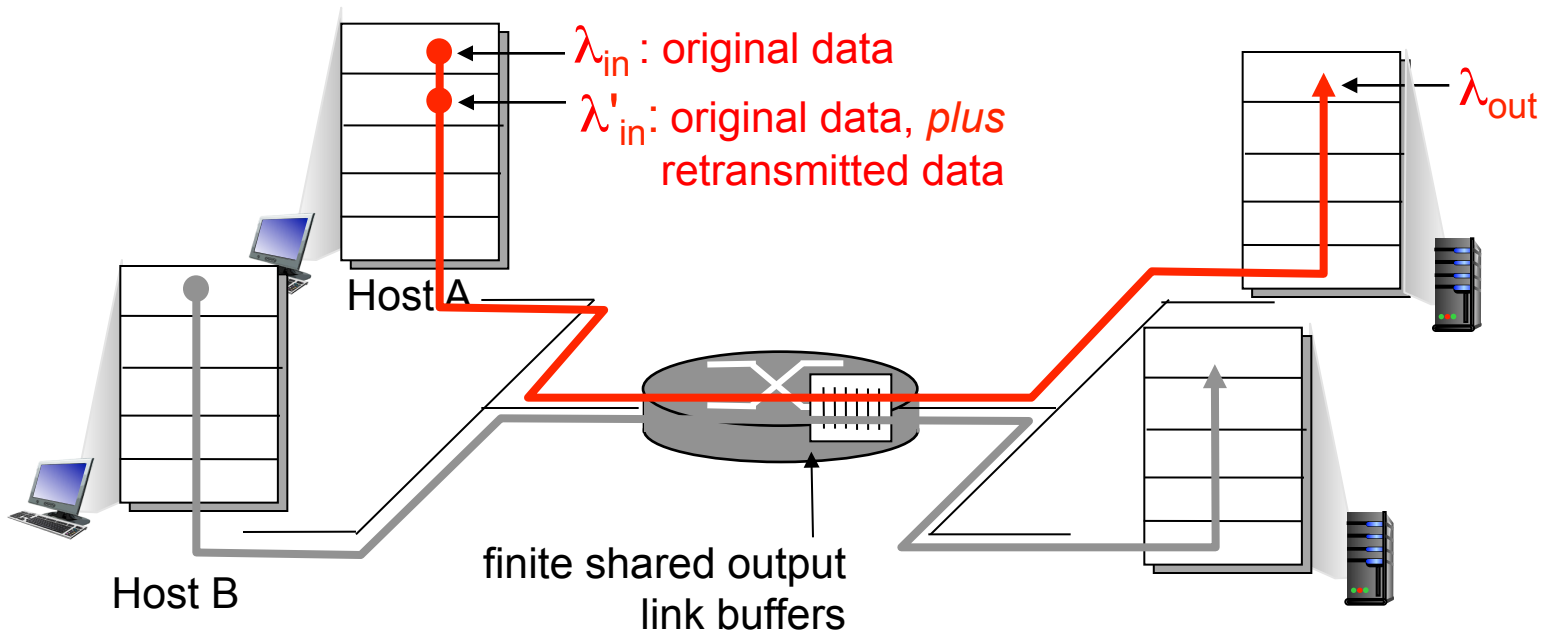
Causes/costs of congestion: scenario 1

- two senders, two receivers
- one router, infinite buffers
- output link capacity: R
- no retransmission

original data: λ_{in}

throughput: λ_{out}

Host A

unlimited shared output link buffers

Host B
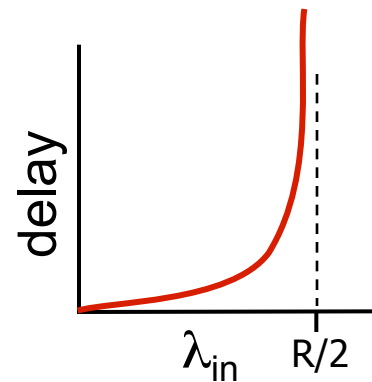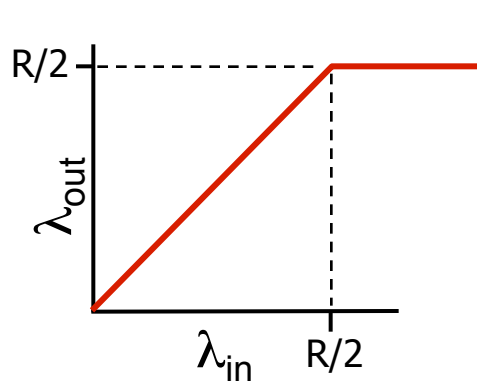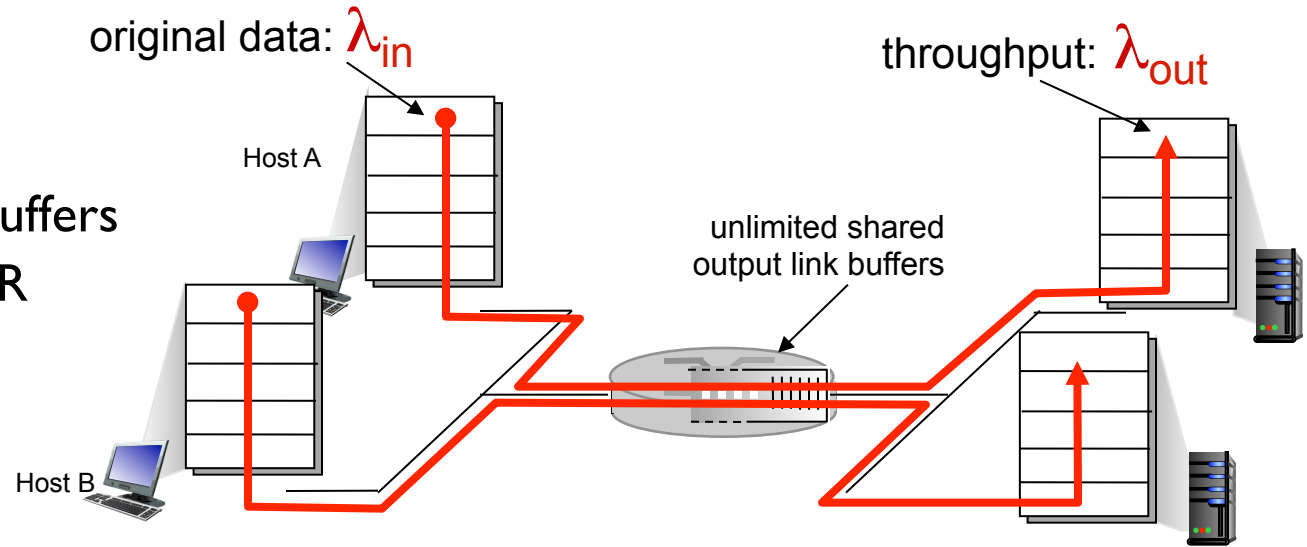
- maximum per-connection throughput: R/2

- large delays as arrival rate, λ_{in}, approaches capacity

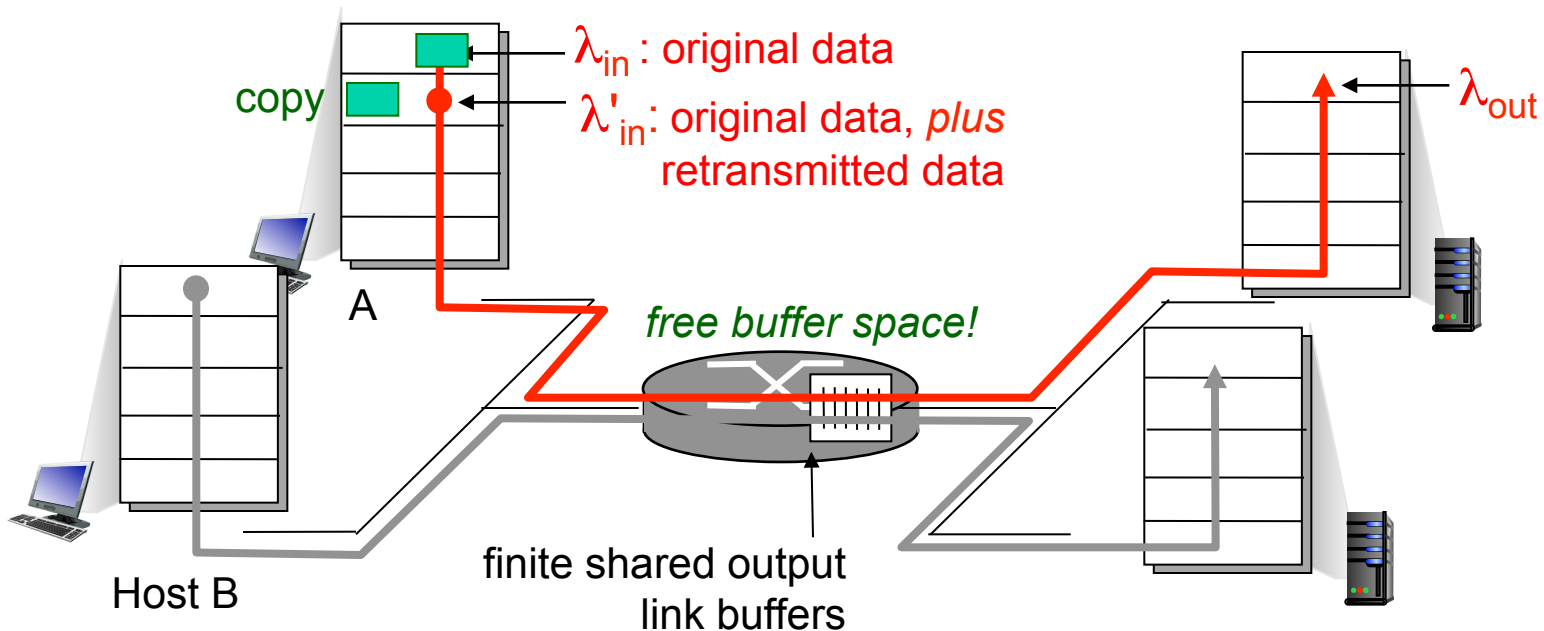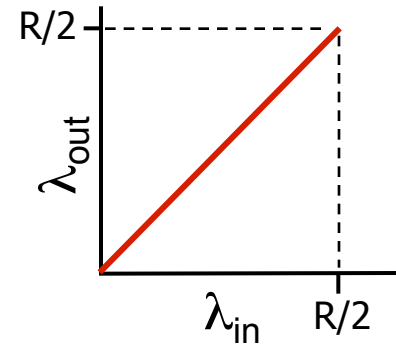# Causes/costs of congestion: scenario 2

- **one router, *finite* buffers**
- **sender retransmission of timed-out packet**
  - application-layer input = application-layer output: $\lambda_{in} = \lambda_{out}$
  - transport-layer input includes *retransmissions* : $\lambda'_{in} \geq \lambda_{in}$

λ_{in} : original data

λ'_{in}: original data, *plus* retransmitted data

λ_{out}

Host A

Host B

finite shared output link buffers

Causes/costs of congestion: scenario 2

idealization: perfect knowledge

- sender sends only when router buffers available

λ_{in} : original data

λ'_{in}: original data, *plus* retransmitted data

copy

λ_{out}

A

free buffer space!
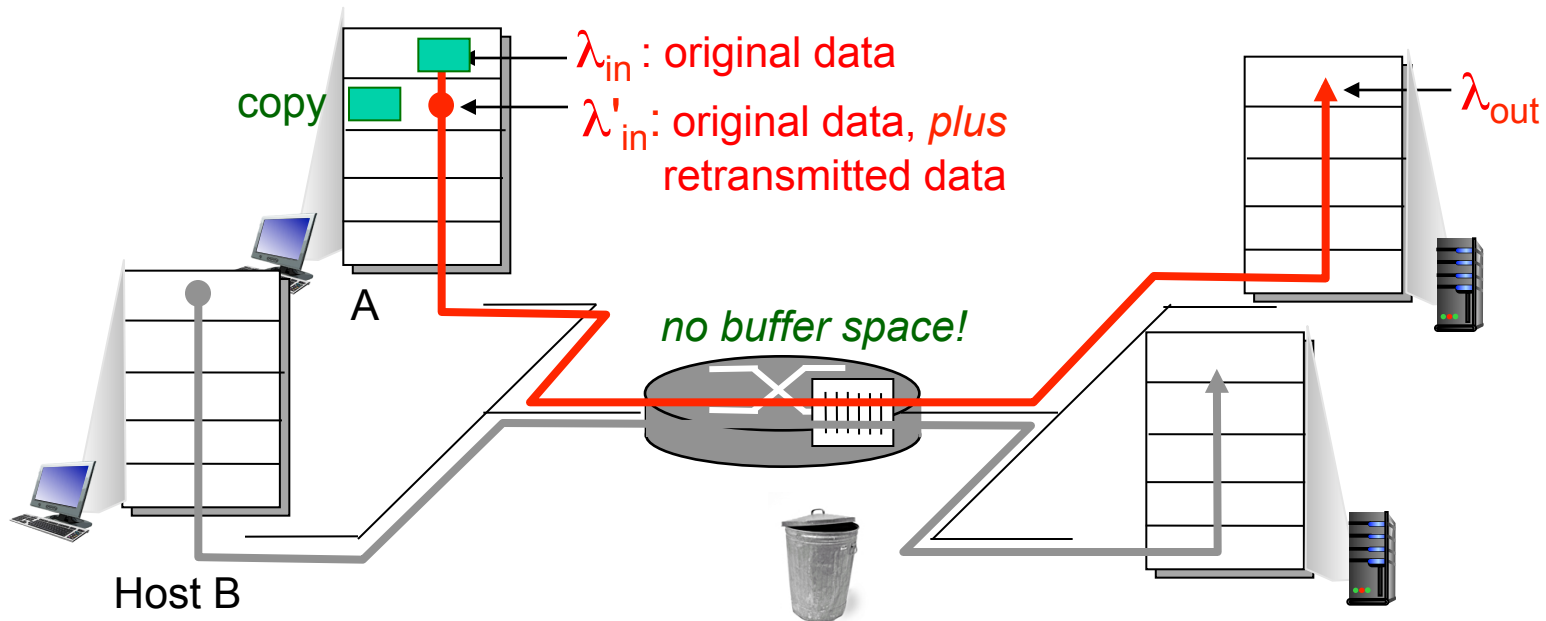
Host B

finite shared output link buffers

Causes/costs of congestion: scenario 2

Idealization: known loss
packets can be lost, dropped at router due to full buffers

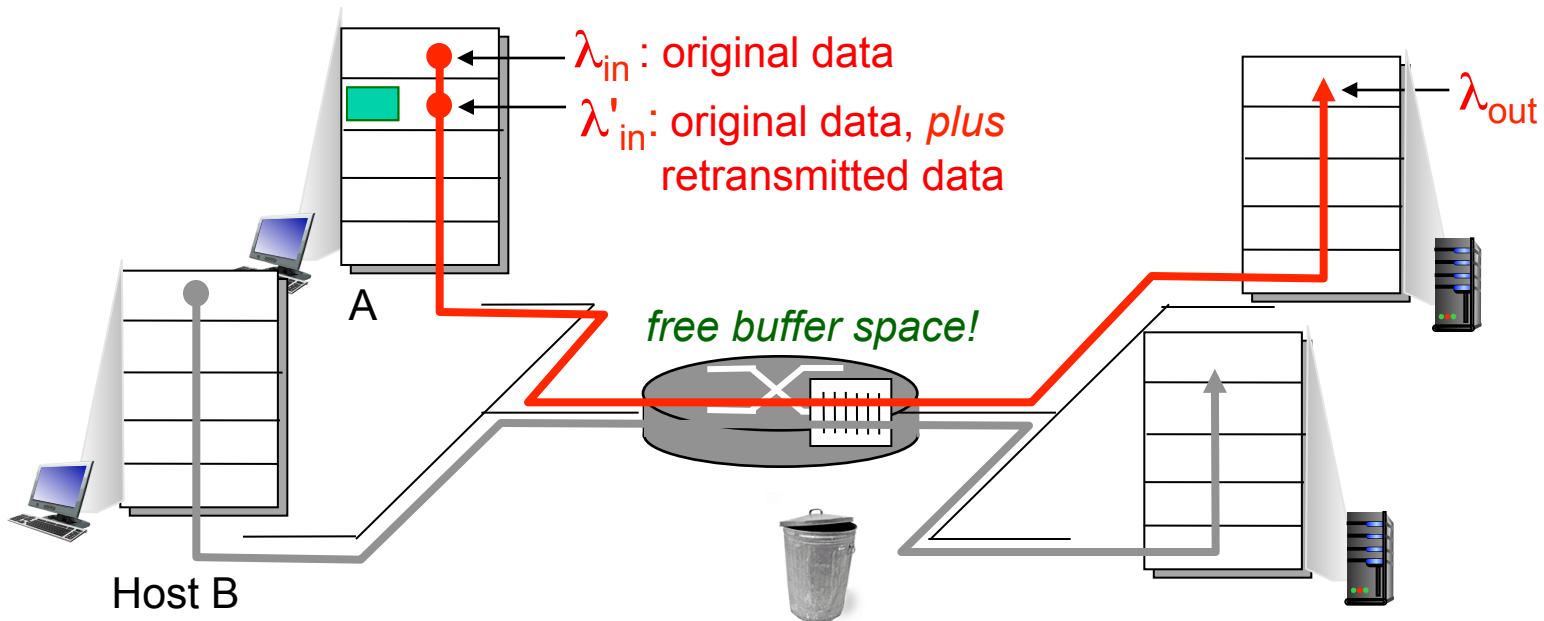- sender only resends if packet *known* to be lost

λ_{in} : original data

λ'_{in}: original data, *plus* retransmitted data

λ_{out}

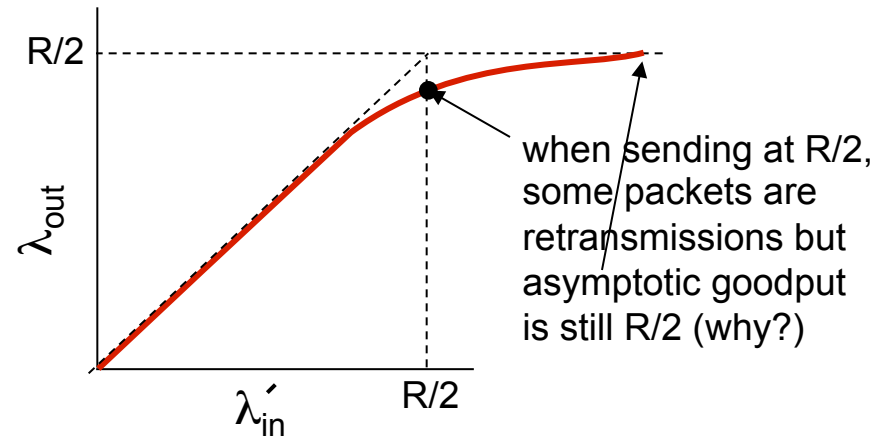copy

no buffer space!

A

Host B

Causes/costs of congestion: scenario 2

Idealization: known loss
packets can be lost, dropped at router due to full buffers

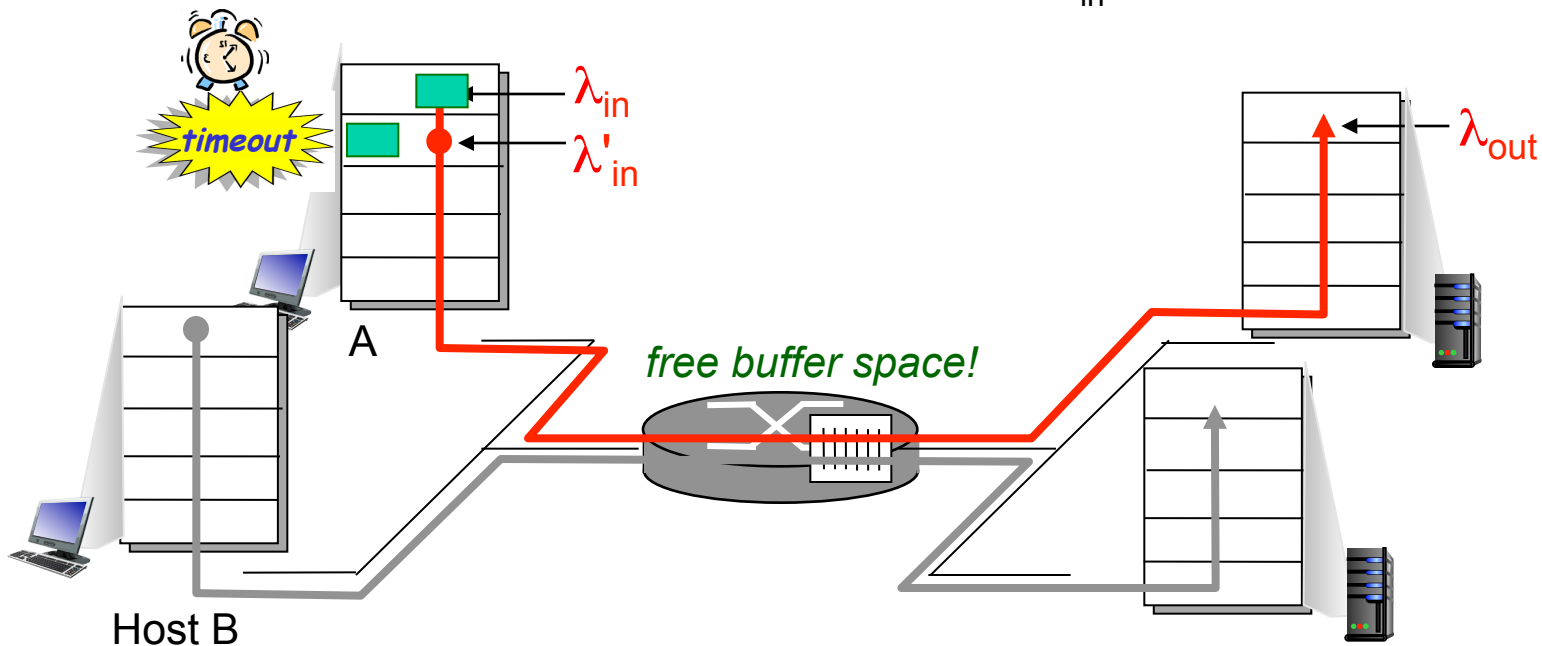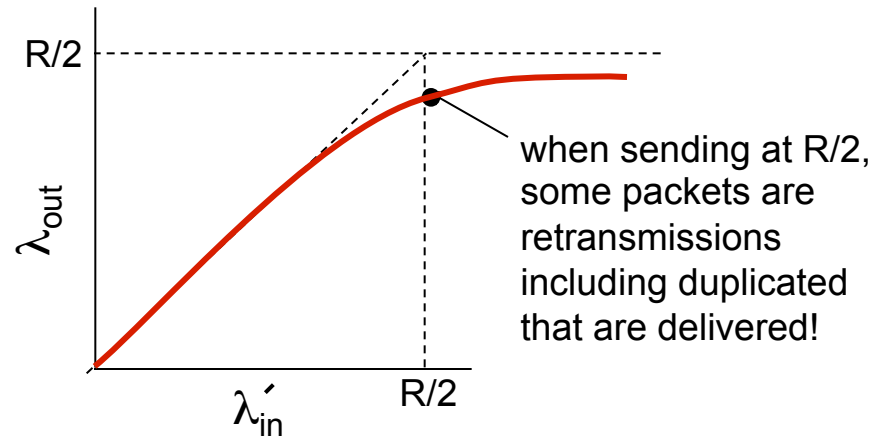- sender only resends if packet *known* to be lost

R/2

λ_{out}

λ_{in}' R/2

when sending at R/2, some packets are retransmissions but asymptotic goodput is still R/2 (why?)

λ_{in} : original data

λ'_{in}: original data, *plus* retransmitted data

λ_{out}

A

free buffer space!

Host B

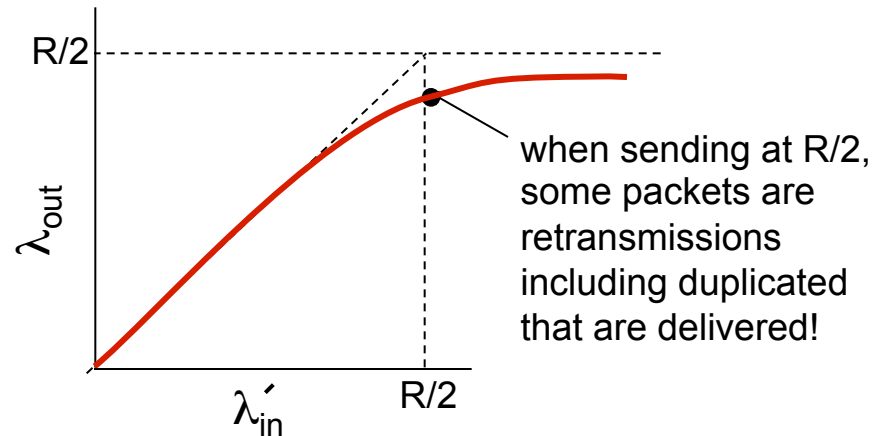# Causes/costs of congestion: scenario 2

Realistic: duplicates

- packets can be lost, dropped at router due to full buffers

- sender times out prematurely, sending *two* copies, both of which are delivered

when sending at R/2, some packets are retransmissions including duplicated that are delivered!

λ_{out}

λ'_{in} R/2

R/2

timeout

λ_{in}

λ'_{in}

A

free buffer space!

λ_{out}

Host B

Causes/costs of congestion: scenario 2

Realistic: duplicates

- packets can be lost, dropped at router due to full buffers
- sender times out prematurely, sending *two* copies, both of which are delivered

R/2

λ_{out}

λ'_{in} R/2

when sending at R/2, some packets are retransmissions including duplicated that are delivered!
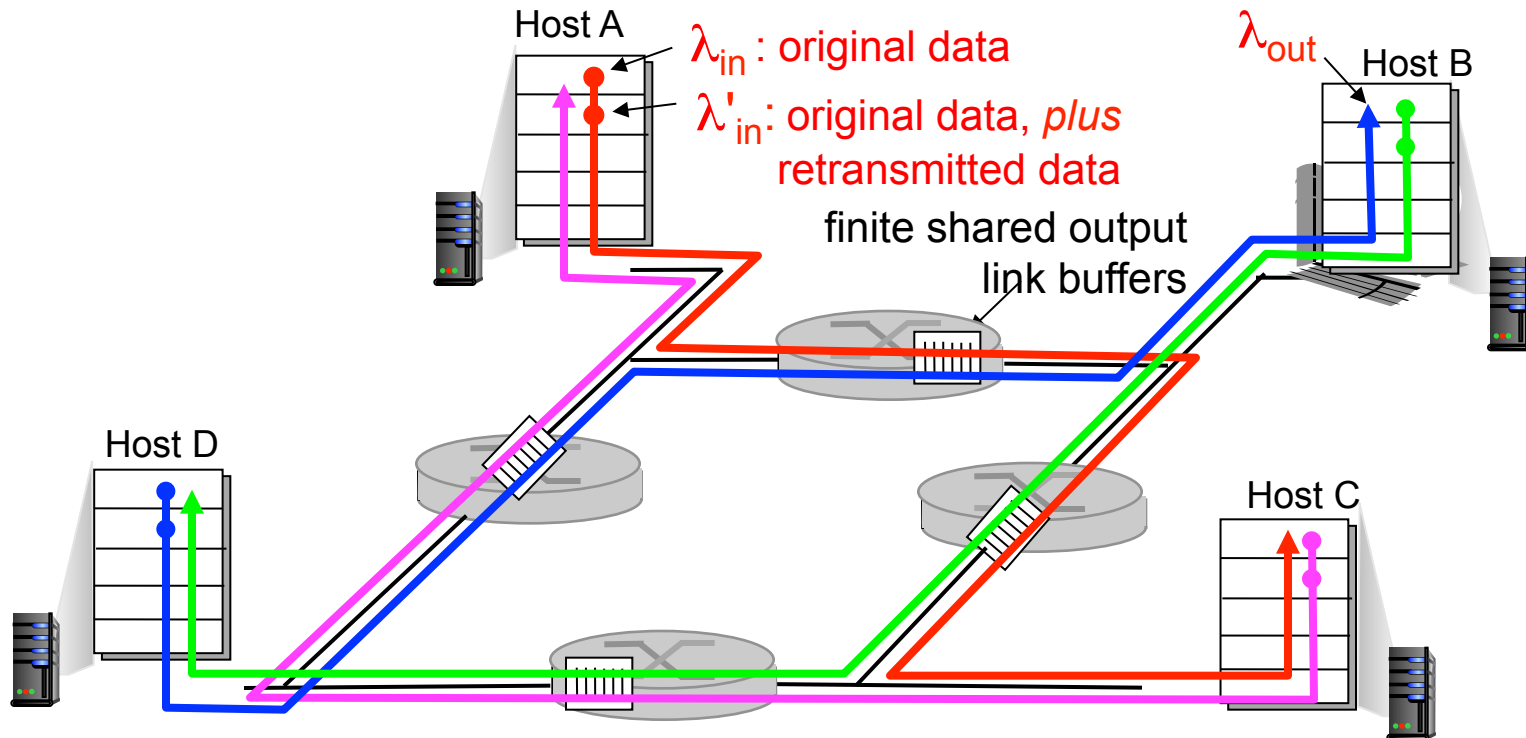
"costs" of congestion:
- more work (retrans) for given "goodput"
- unneeded retransmissions: link carries multiple copies of pkt
  - decreasing goodput
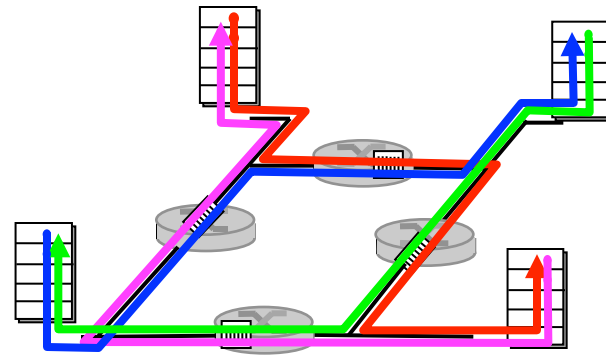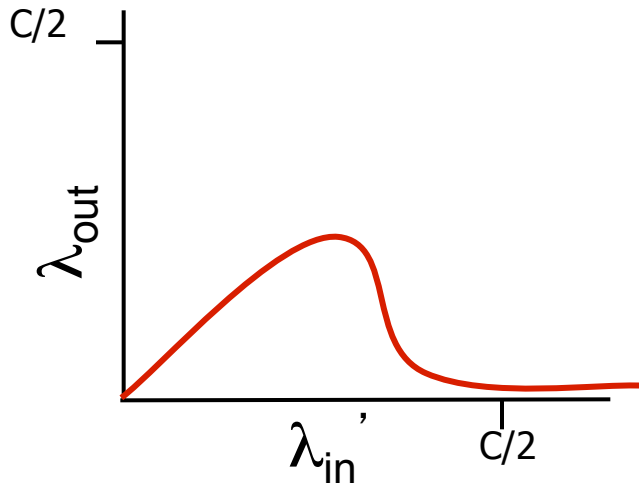
Causes/costs of congestion: scenario 3

- four senders
- multihop paths
- timeout/retransmit

Q: what happens as λ_{in} and λ_{in}' increase?

A: as red λ_{in}' increases, all arriving blue pkts at upper queue are dropped, blue throughput $\rightarrow 0$

Host A

λ_{in} : original data

λ_{in}' : original data, plus retransmitted data

finite shared output link buffers

λ_{out}

Host B

Host D

Host C

Causes/costs of congestion: scenario 3

another "cost" of congestion:

- when packet dropped, any "upstream transmission capacity used for that packet was wasted!

Chapter 3 outline

3.1 transport-layer services

3.2 multiplexing and demultiplexing

3.3 connectionless transport: UDP

3.4 principles of reliable data transfer

3.5 connection-oriented transport: TCP
- segment structure
- reliable data transfer
- flow control
- connection management

3.6 principles of congestion control

3.7 TCP congestion control