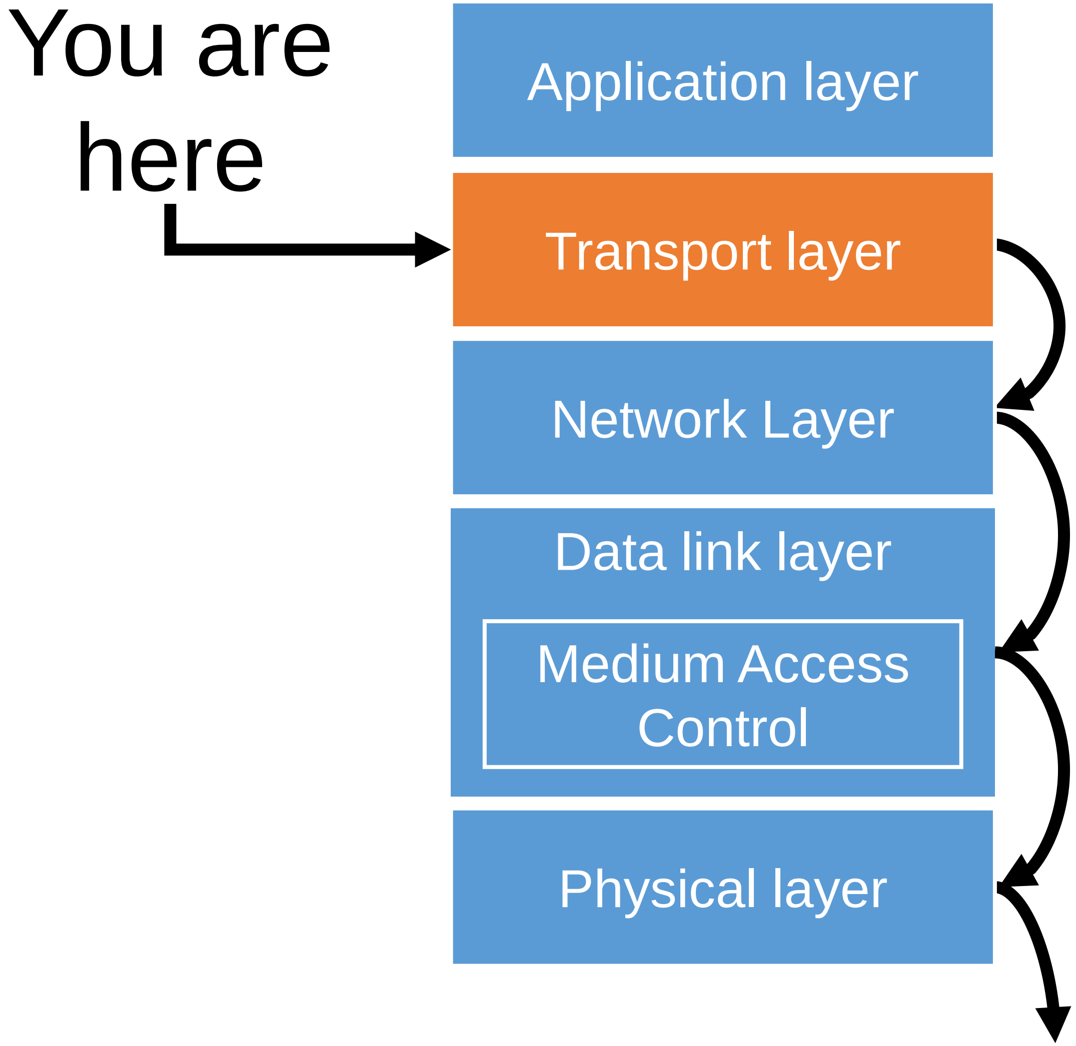
Computer Networks X_400487

Lecture 9

Chapter 6: The Transport Layer—Part 1

Lecturer: Jesse Donkervliet

Transport layer



Send segments between processes

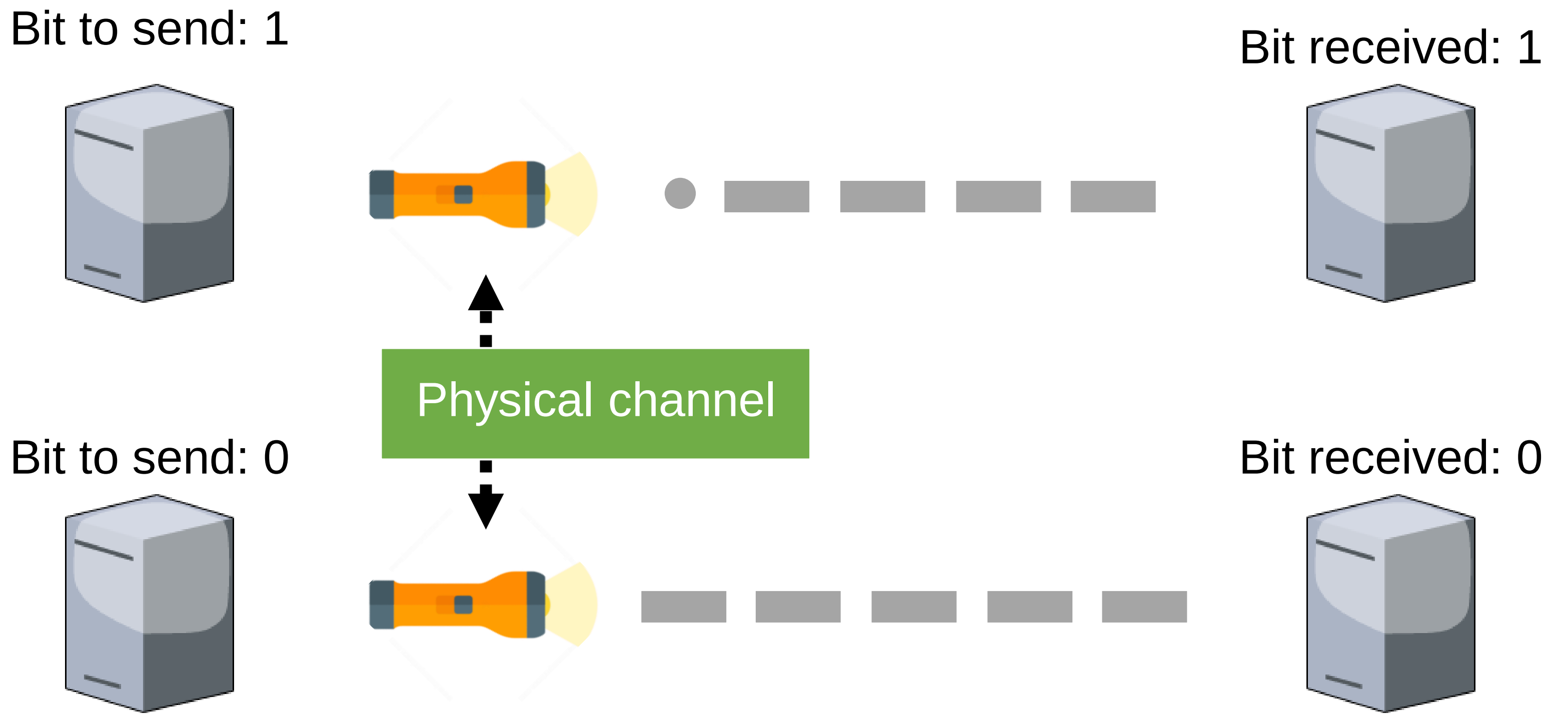
Send packets over (multiple) networks

Send frames over single link

Send digital signals

Recap of lower layers
The physical layer

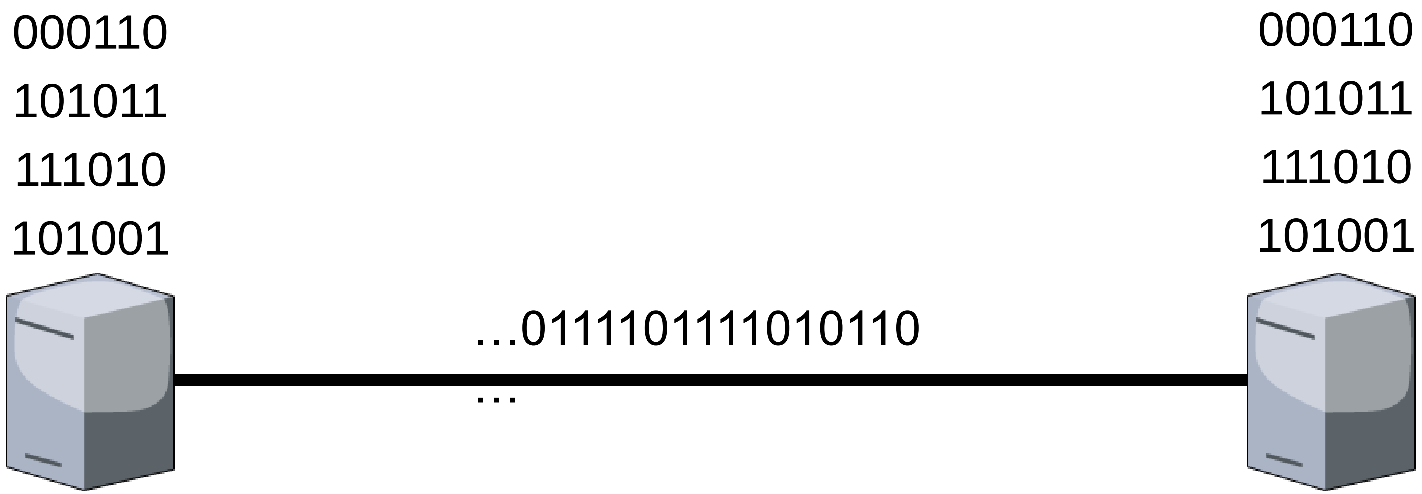
Moves bits over a physical channel.



Recap of lower layers
The data link layer

Translates frames to and from bit/byte streams.

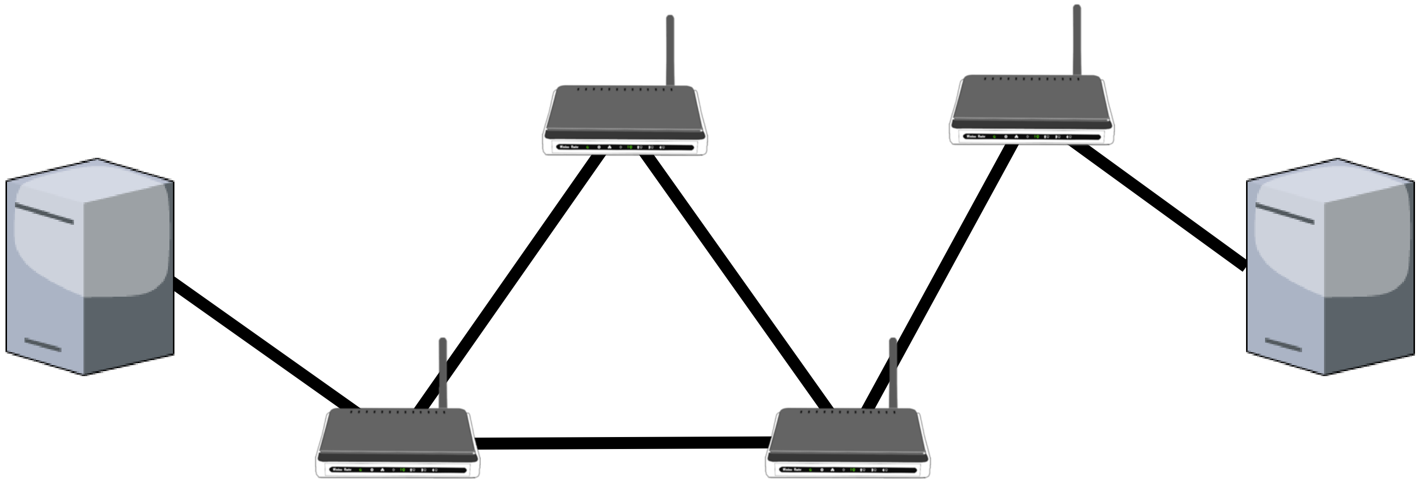
Provides error detection/correction and flow control.



Recap of lower layers
The network layer

Transmits packets across the network from a source host to a destination ***host***

Provides ***congestion control*** together with the transport layer



Roadmap: Transport Layer

1. Transport layer responsibilities and challenges
2. Connection establishment and release
3. Revisiting reliable delivery and flow control
4. Congestion control and bandwidth allocation
5. TCP and UDP

The transport layer Provided services

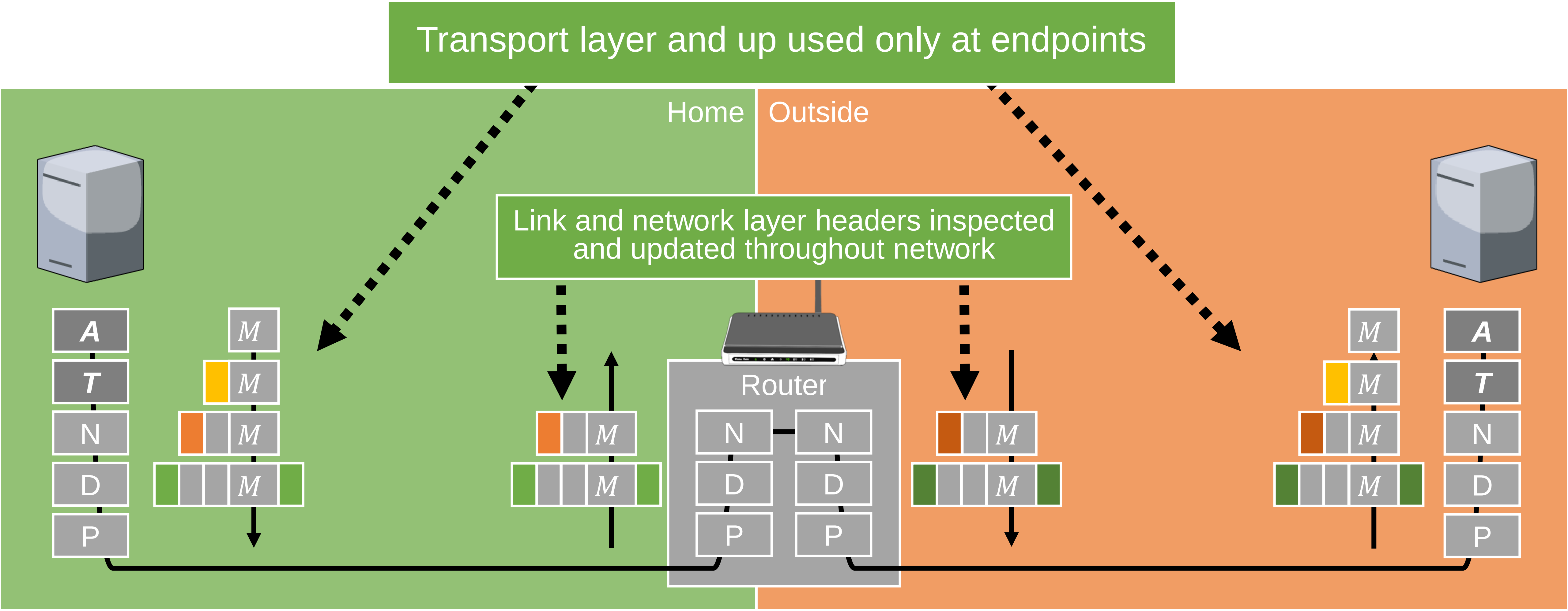
Runs only on the host and destination

Provides a ***reliable*** data stream over an ***unreliable*** network.

Provides communication between ***processes***.

Q: Does this resemble a layer we have seen?

Transport layer only present at source and destination



Primitives used to offer this service

The interface exposed to the application layer

1. Listen – wait for another process to contact us.
2. Connect – connect to a process that is ***listening***.
3. Send – send data over the established ***connection***.
4. Receive – receive data over the established ***connection***.
5. Disconnect – release the ***connection***.

Connection-oriented service over (possibly) connectionless network!

Berkeley Socket primitives

The interface exposed to the application layer

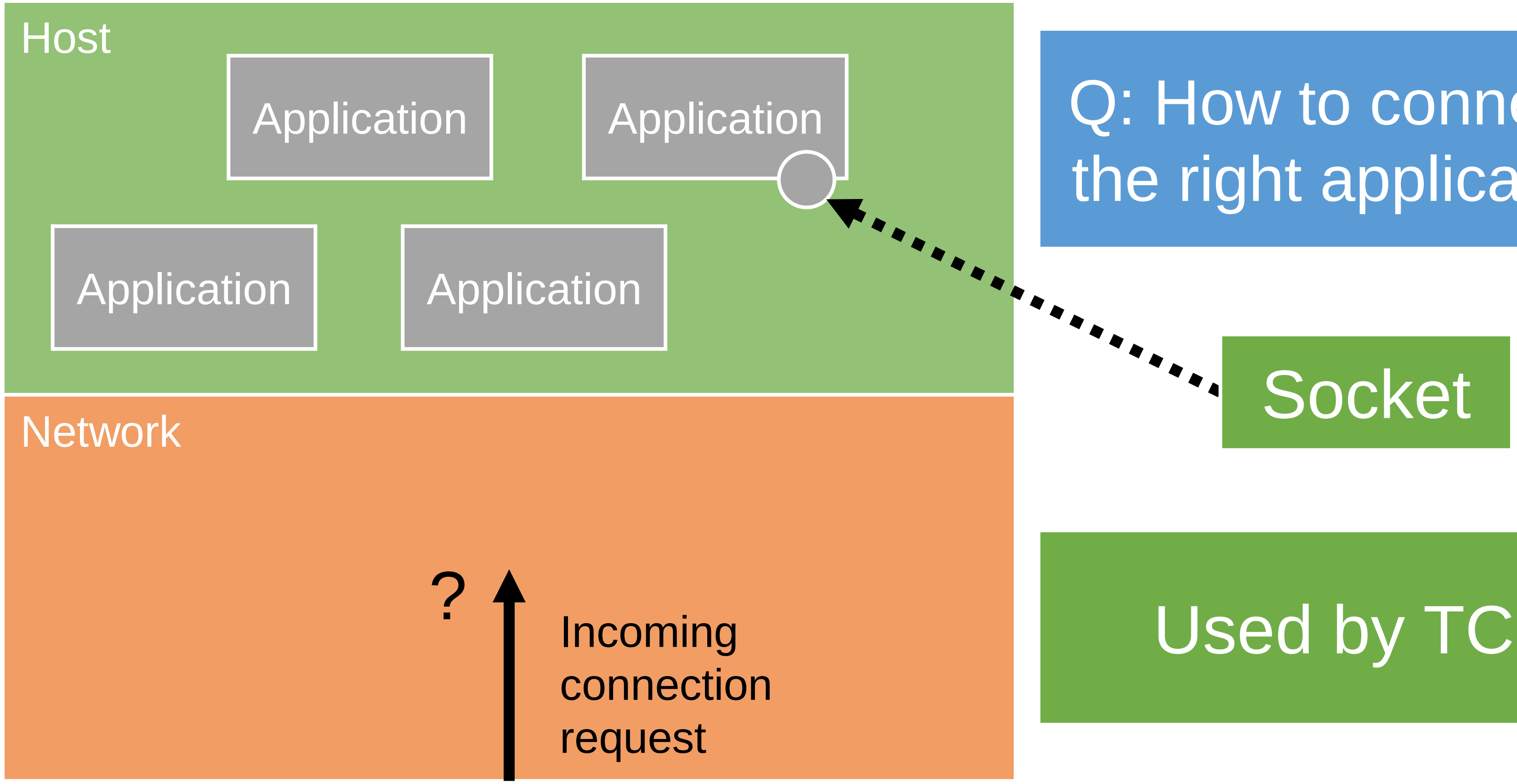
1. Socket – create a new communication ***endpoint***.
2. Bind – assign a ***local address*** to an endpoint (socket).
3. Listen.
4. Accept – passively establish an incoming connection.
5. Connect.
6. Send.
7. Receive.
8. Close.

Q: Which ones (not) used by UDP?

Used by TCP!

Berkeley Socket primitives

1. Socket – create a new communication ***endpoint***.

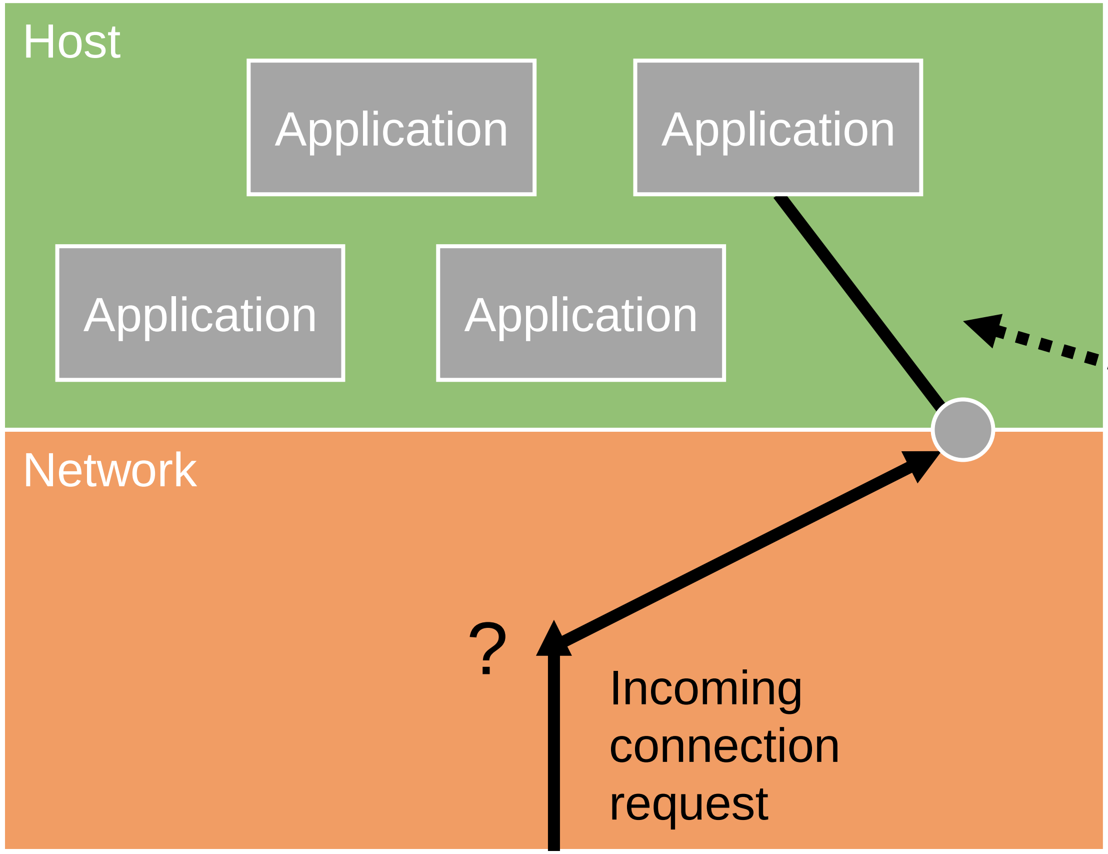


Q: How to connect to the right application?

Used by TCP!

Berkeley Socket primitives

1. Socket – create a new communication ***endpoint***.



Q: How to connect to the right application?

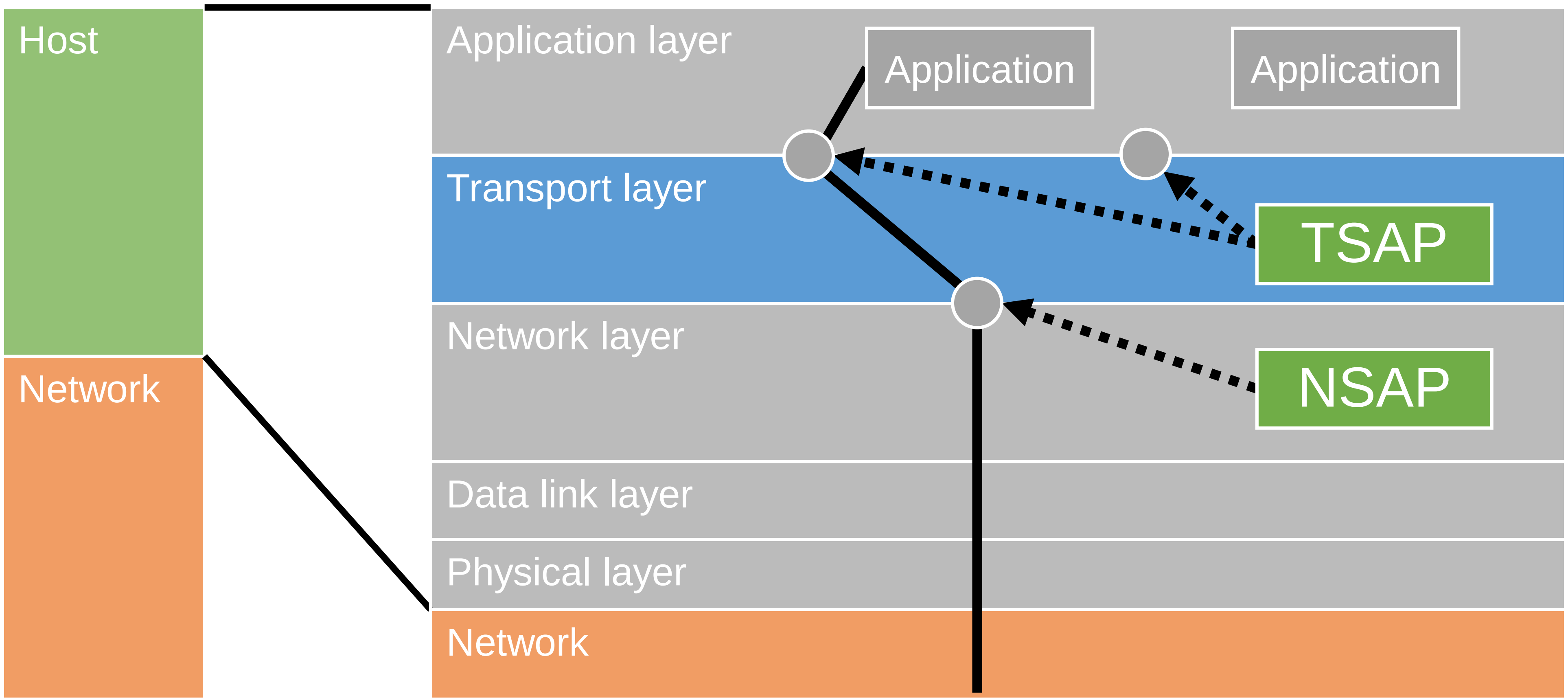
BIND

Used by TCP!

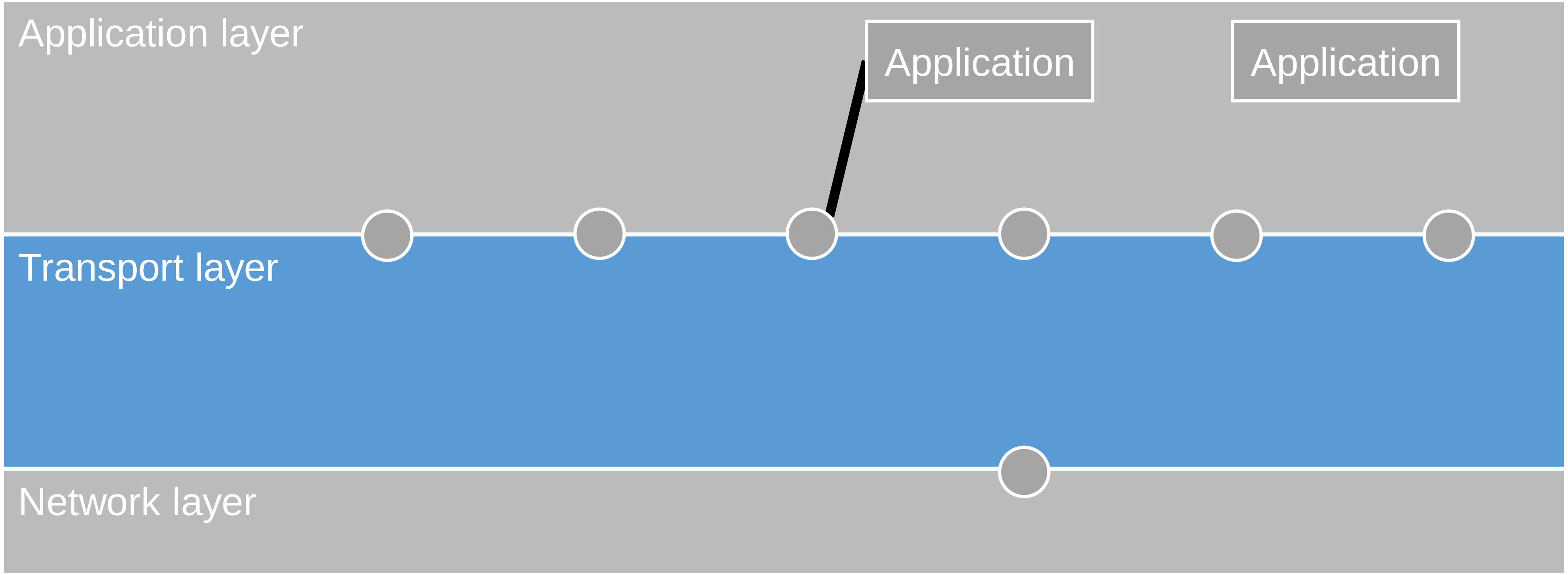
Addressing

TSAP = Transport Service Access Point
NSAP = Network Service Access Point

Internet uses IP addresses for NSAPs and ***ports*** for TSAPs



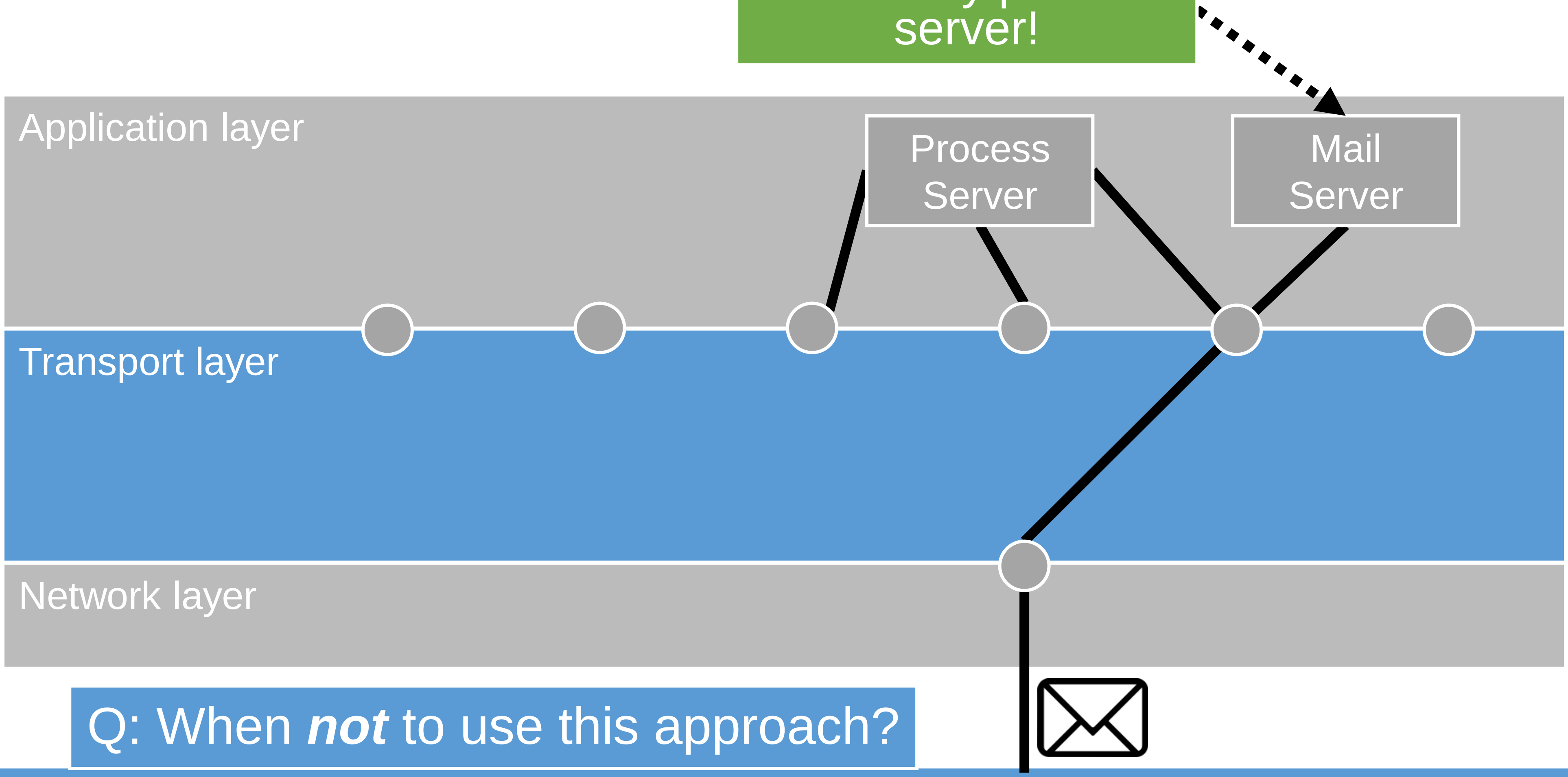
Process servers



Process servers

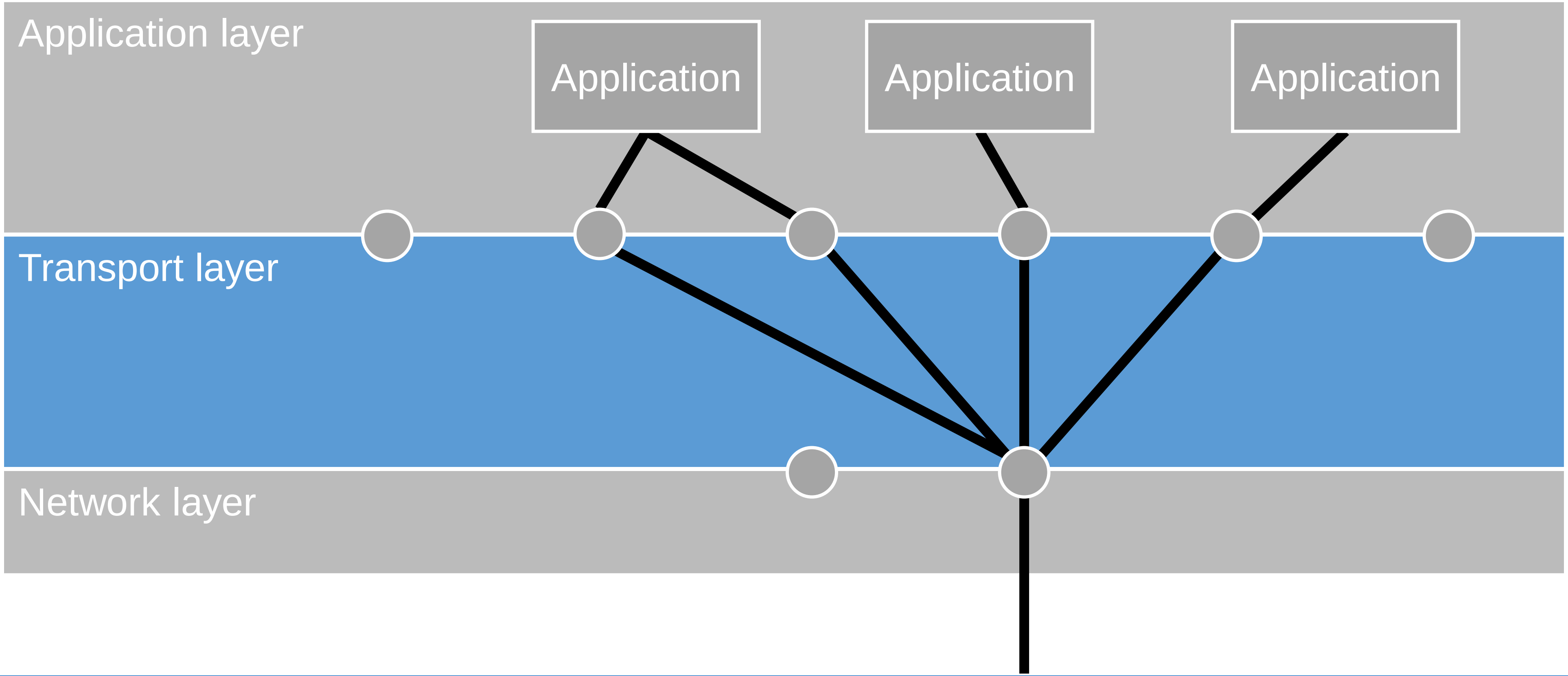
Mail server is only started when needed.

Started by process server!

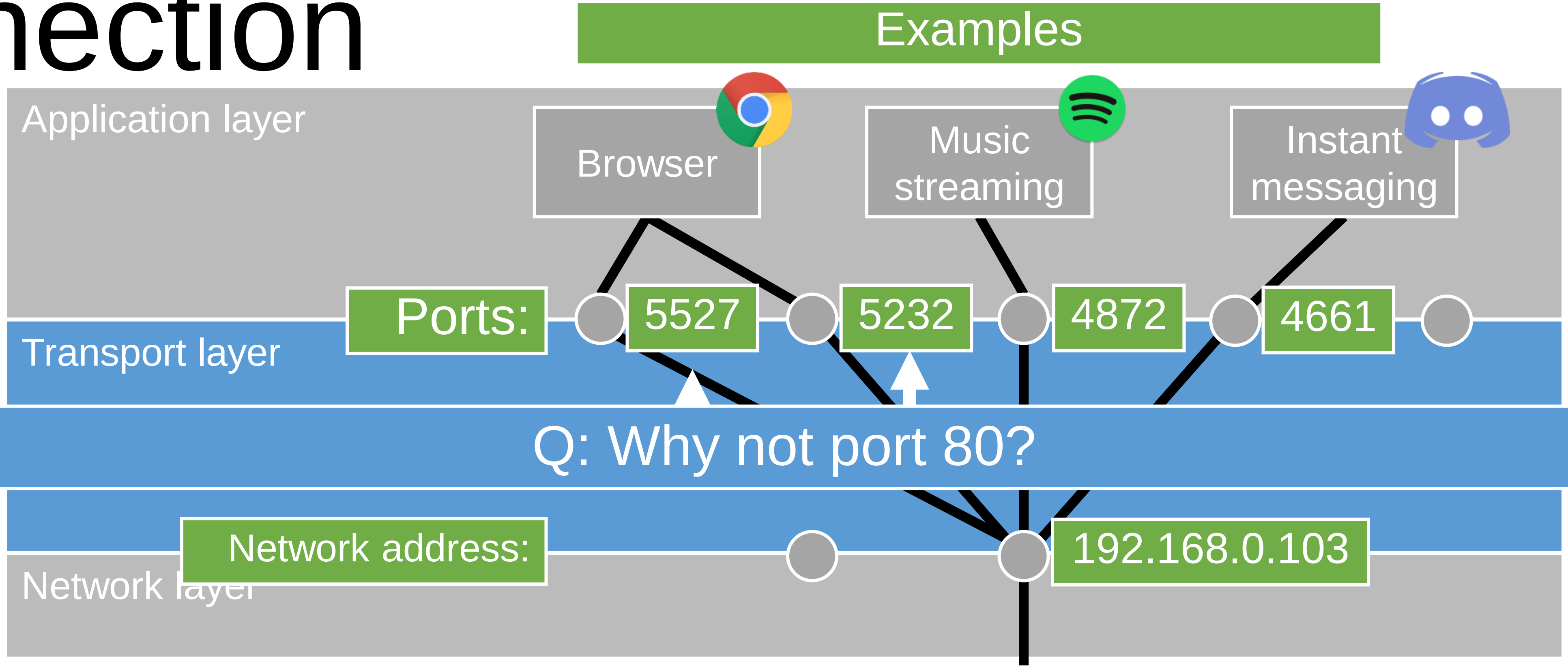


Q: When ***not*** to use this approach?

Multiplexing: Multiple transport connections over one network connection



Multiplexing: Multiple transport connections over one network connection

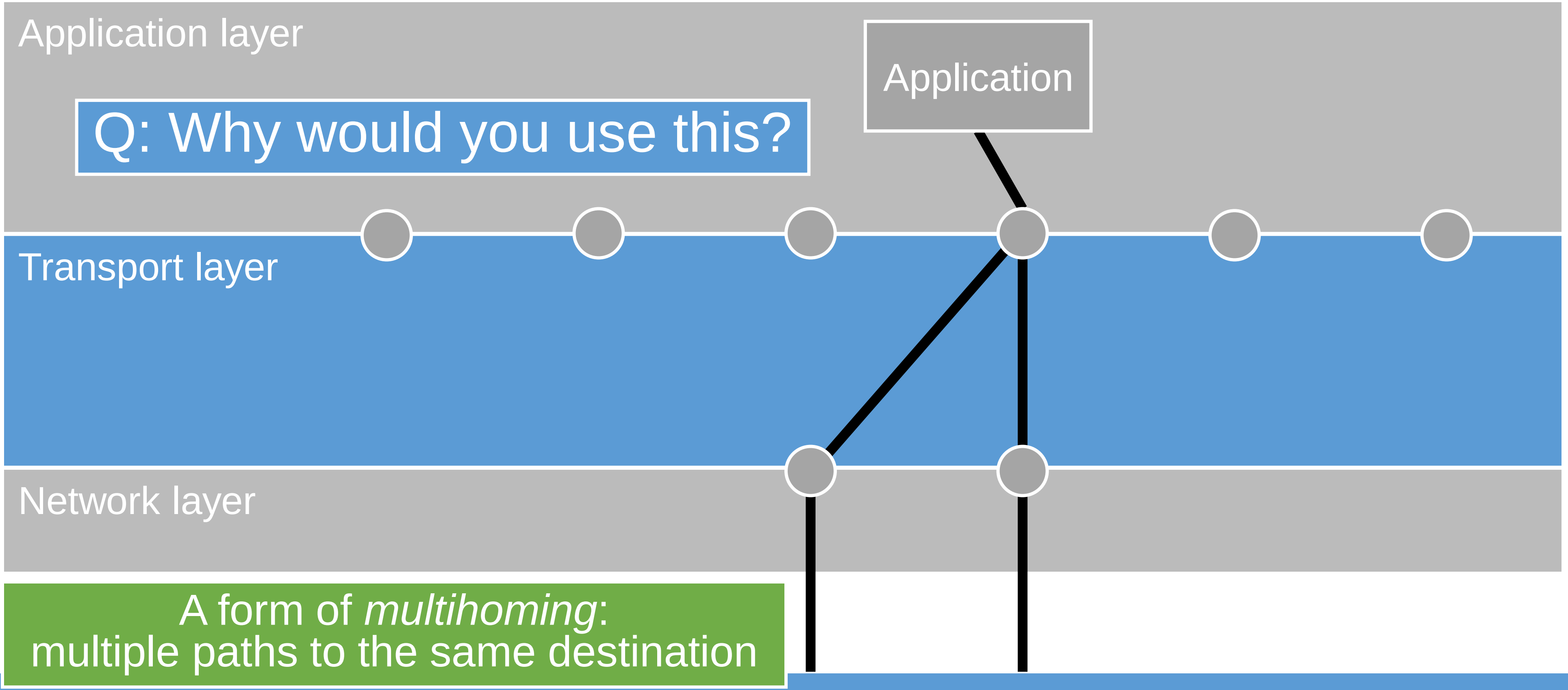


Q: Why not port 80?

Q: How does incoming connection know the correct port?

Server ports typically hardcoded!

Inverse multiplexing: One transport connection over multiple network connections



Network Address Translation (NAT)

Q: How to send something back to a_s?

a_s = source address.

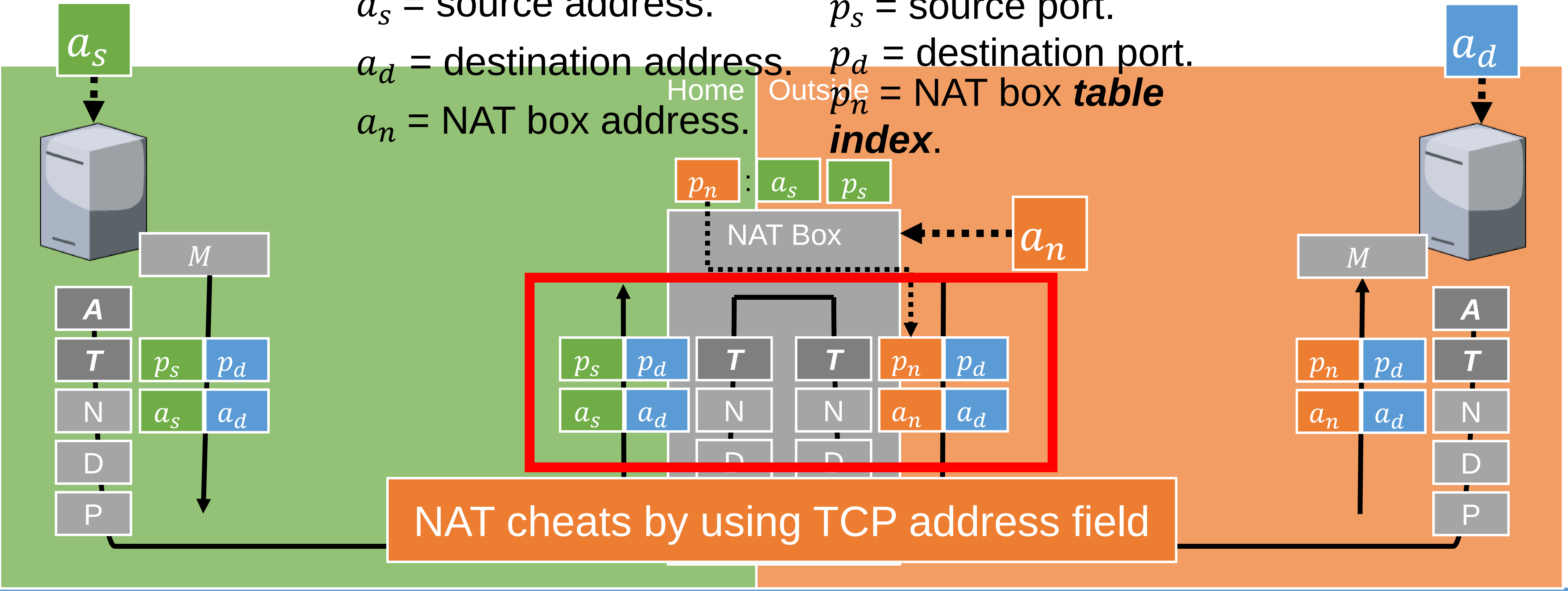
a_d = destination address.

a_n = NAT box address.

p_s = source port.

p_d = destination port.

p_n = NAT box ***table index***.



NAT cheats by using TCP address field

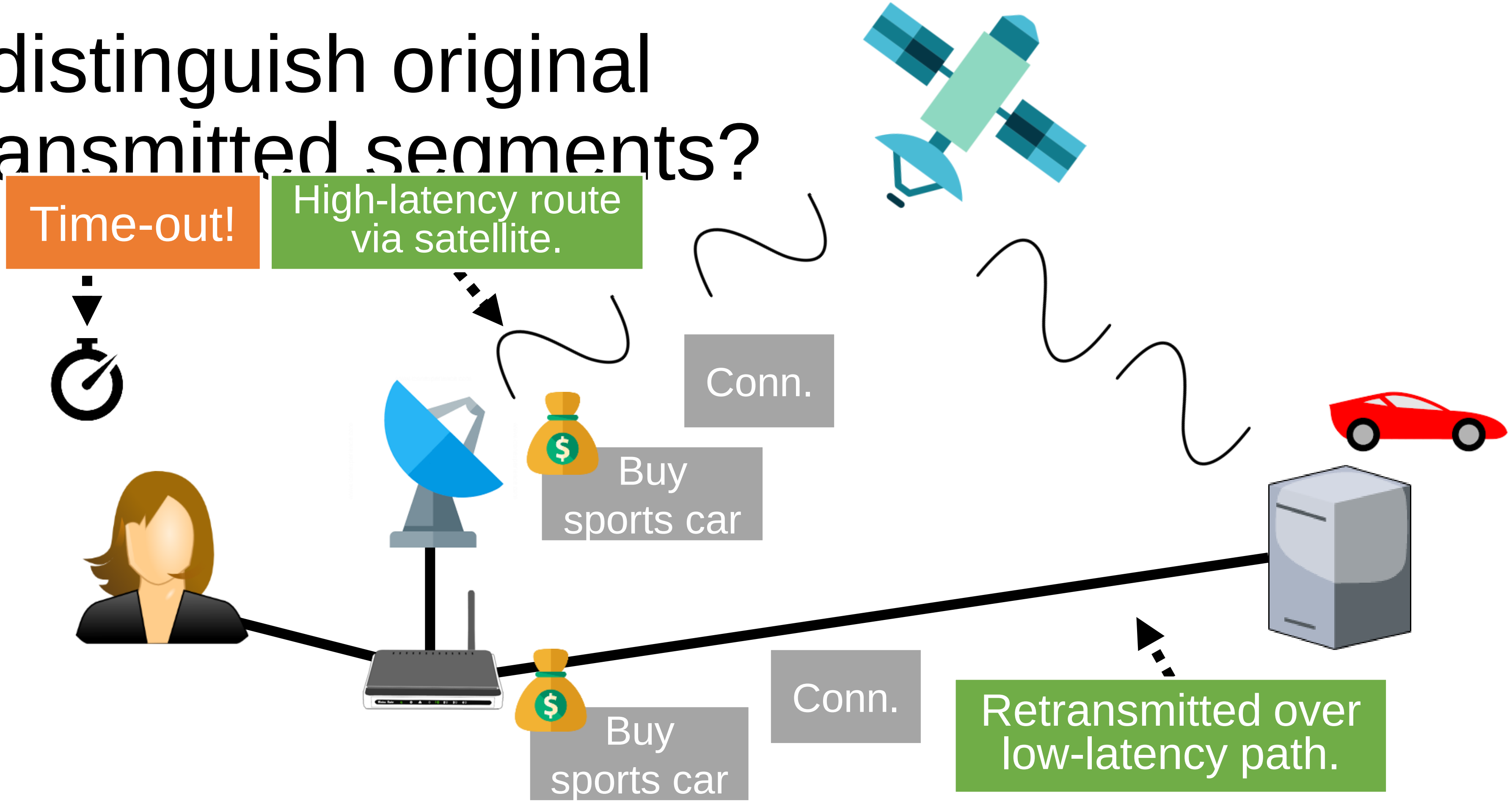
Roadmap: Transport Layer

1. Transport layer responsibilities and challenges
2. **Connection establishment and release**
   1. **Connection establishment**
   2. Connection release
3. Revisiting reliable delivery and flow control
4. Congestion control and bandwidth allocation
5. TCP and UDP

Connection Establishment

Q: Don’t the lower layers solve this problem?

How to distinguish original
and retransmitted segments?



Connection establishment using sequence numbers

If a segment comes in with a sequence number that we have already seen, we discard it.

Q: Can you think of a subproblem we need to solve?

1. How do we ensure that there are never ***multiple*** packets with ***the same*** sequence number?
2. If a machine crashes and reboots, what sequence number should it choose?

Connection establishment using sequence numbers

1. We use the packet ***hop limit*** to remove old packets. After time T, sequence numbers safe to wrap around.
2. We use a ***time-of-day clock*** to decide which sequence number to choose. Keeps working when host crashes.

Sequence Number Limits Performance

x bit sequence number

y bytes per second sending rate

Sequence number wraps around after $\frac{2^x}{y}$ seconds

Sequence number that reappears within T seconds is retransmission

Sequence number that reappears later is new segment

Maximum sending rate:

$\frac{2^x}{T}$ Bps (bytes per second)

000
001
010
011
100
101
110
111
000
001
010
011
100
101
110
111

Sequence Number Limits Performance

32 bit sequence number

Sequence number that reappears within 128 seconds is retransmission

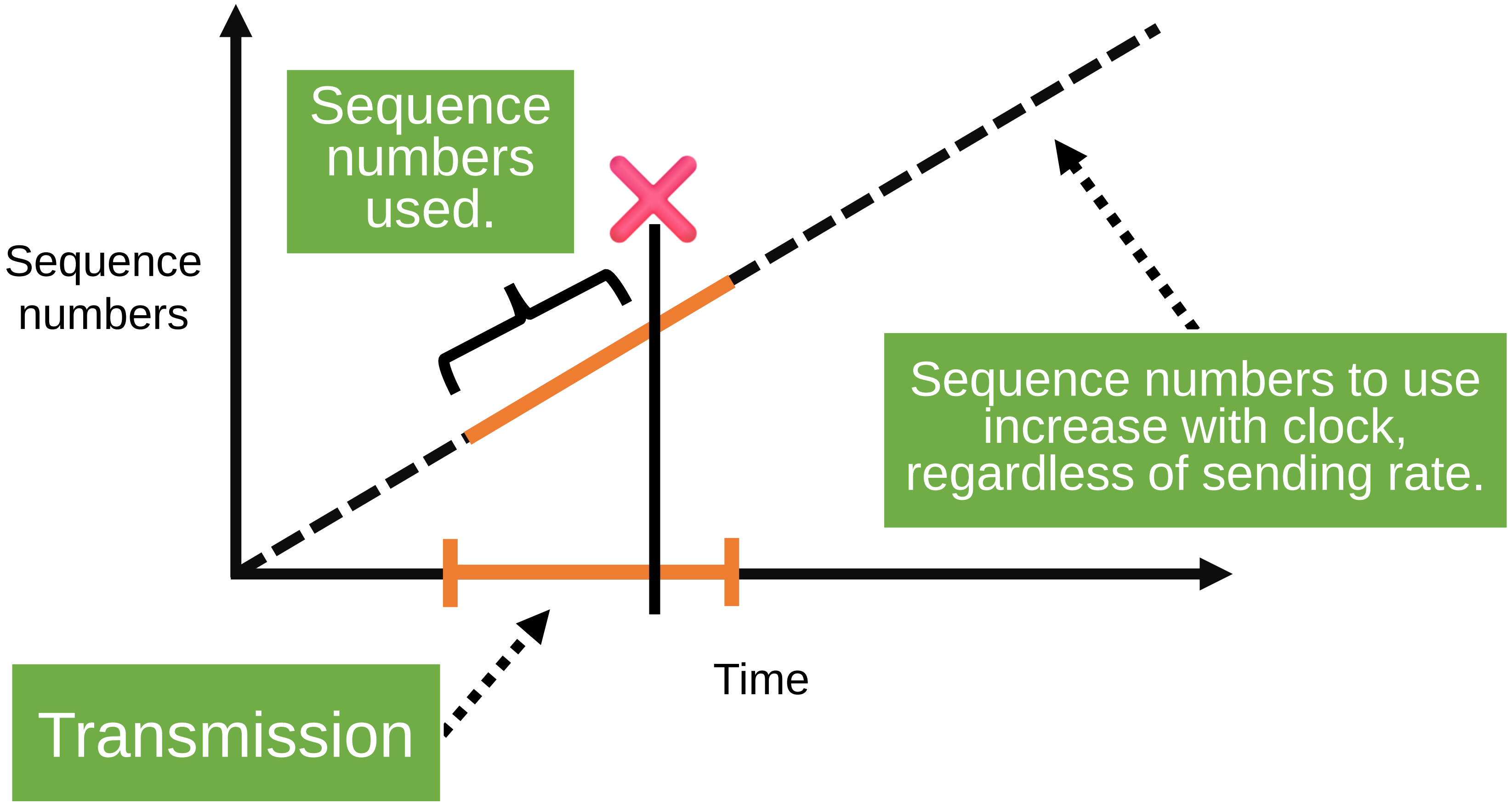
Q: What is the maximum sending rate?

$$\frac{2^x}{T} = \frac{2^{32}}{128} = 2^{25} = 32 \text{ MiB/s}$$

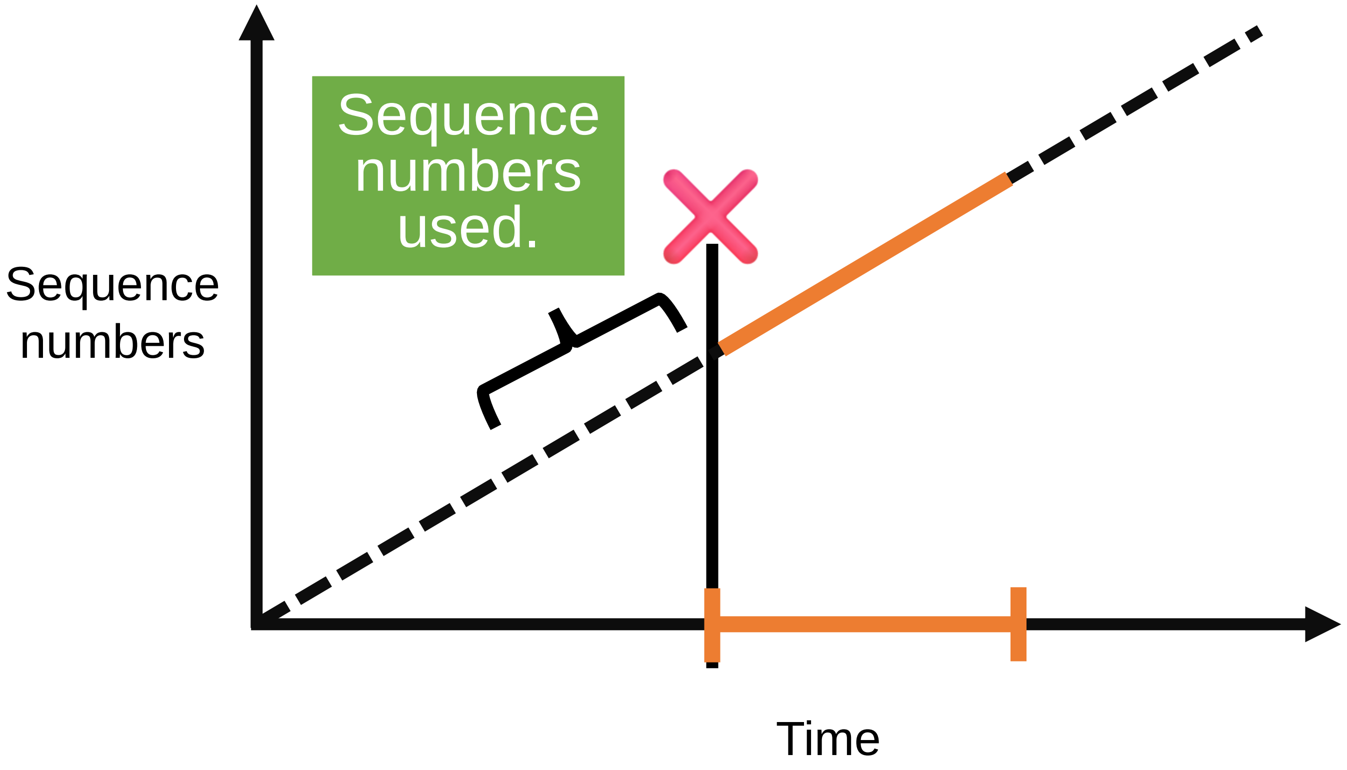
$$1 \text{ MiB} = 2^{20} \text{ bytes} = 1{,}048{,}576 \text{ bytes}$$

000
001
010
011
100
101
110
111
000
001
010
011
100
101
110
111

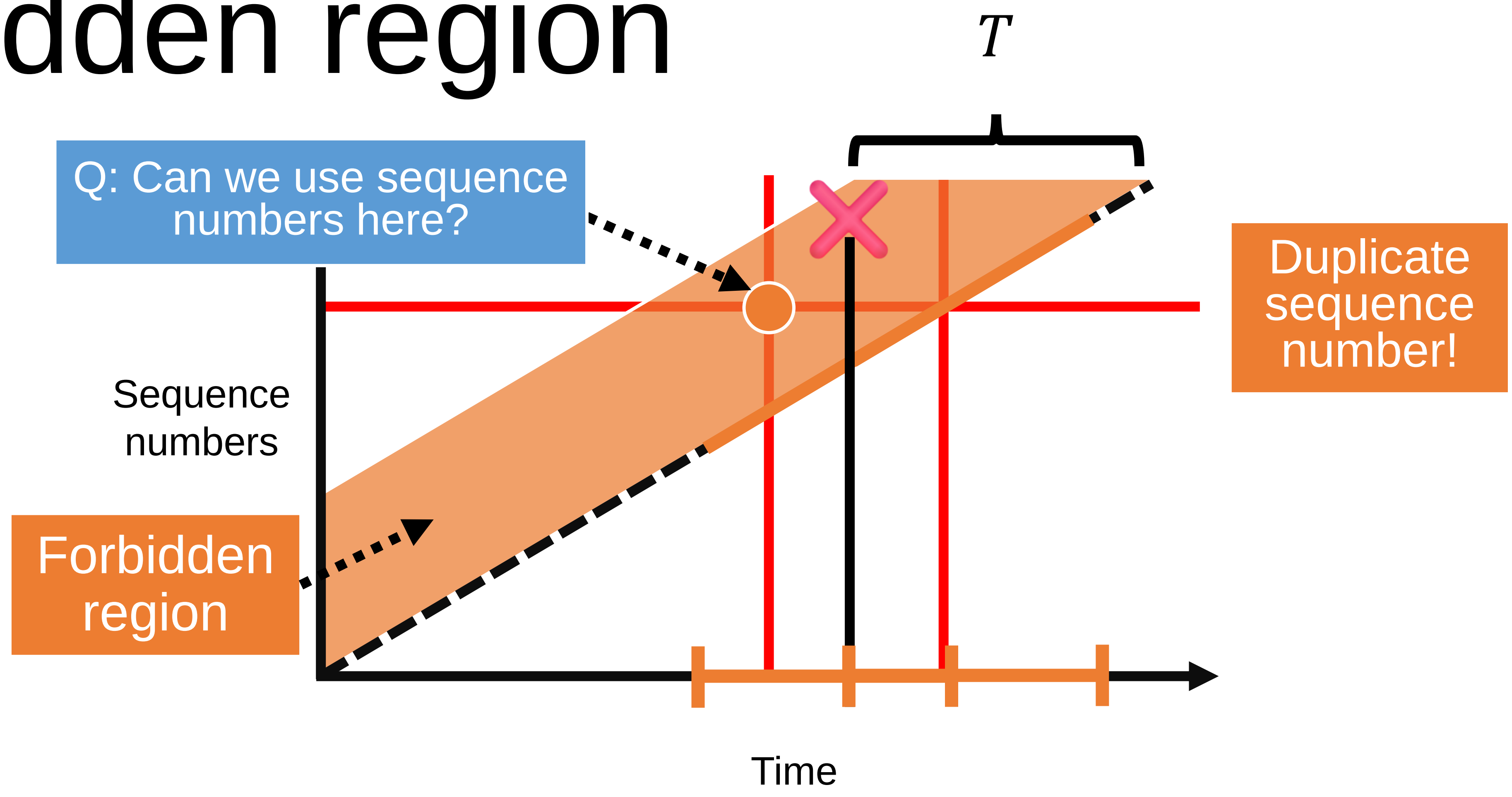
Clock-based sequence numbers



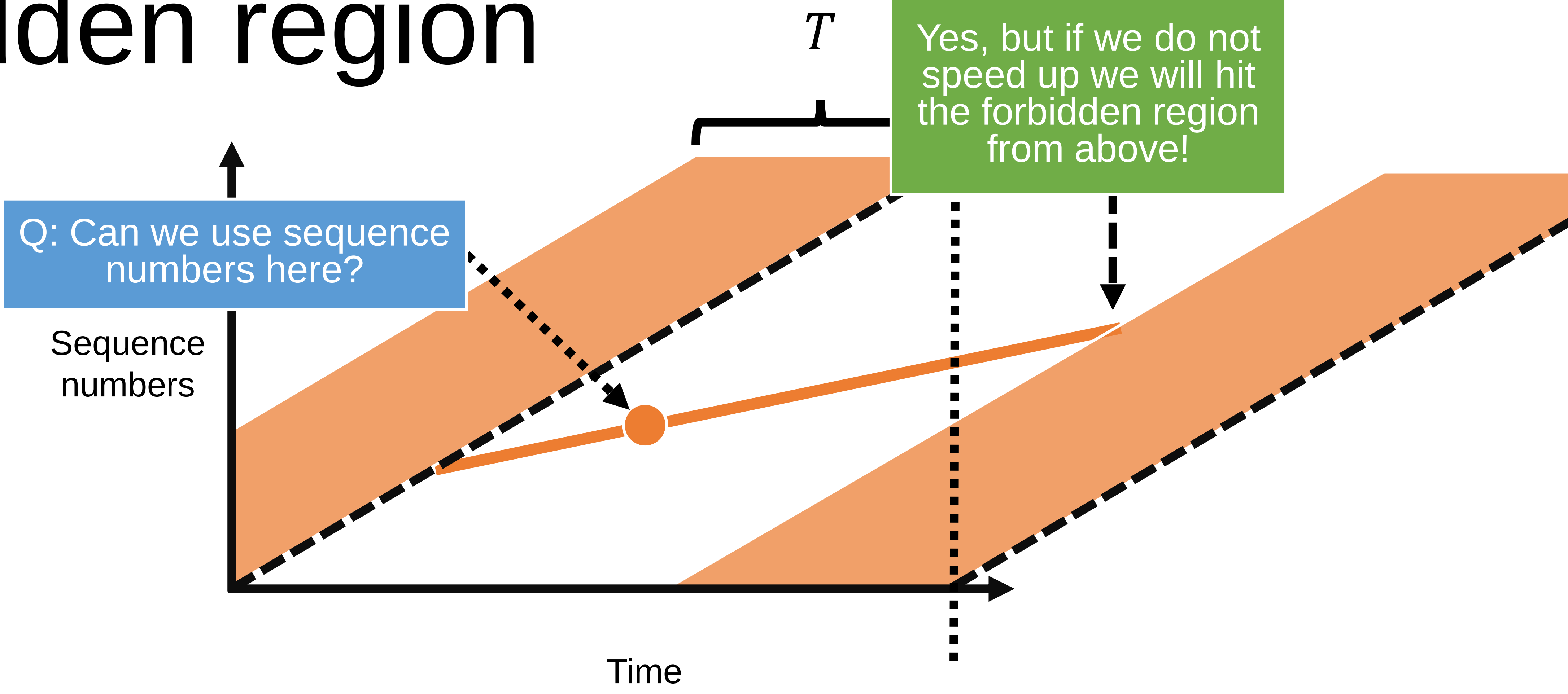
Clock-based sequence numbers



Clock-based sequence numbers forbidden region



Clock-based sequence numbers forbidden region

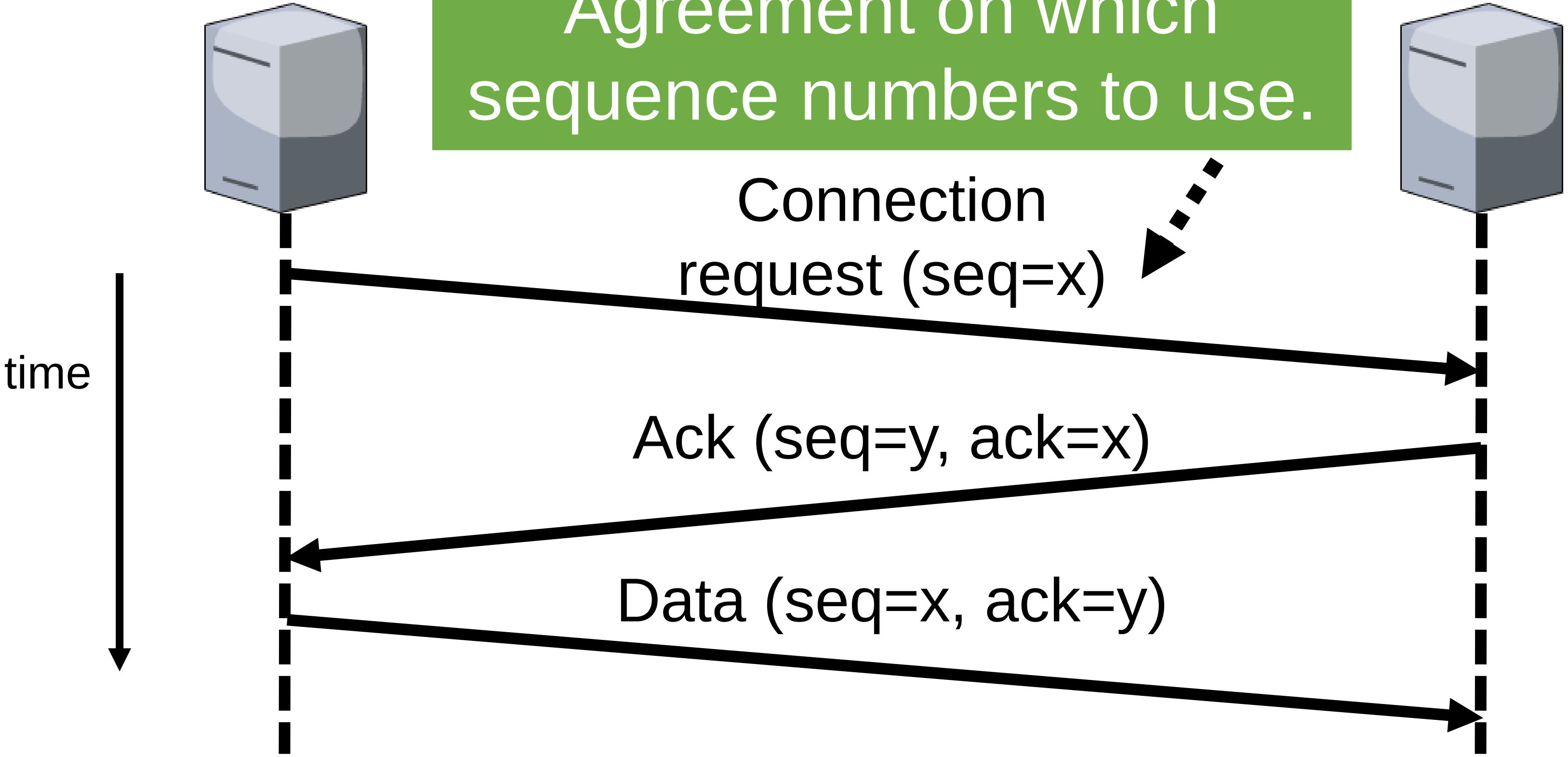


Q: Can a receiver always detect delayed duplicates?

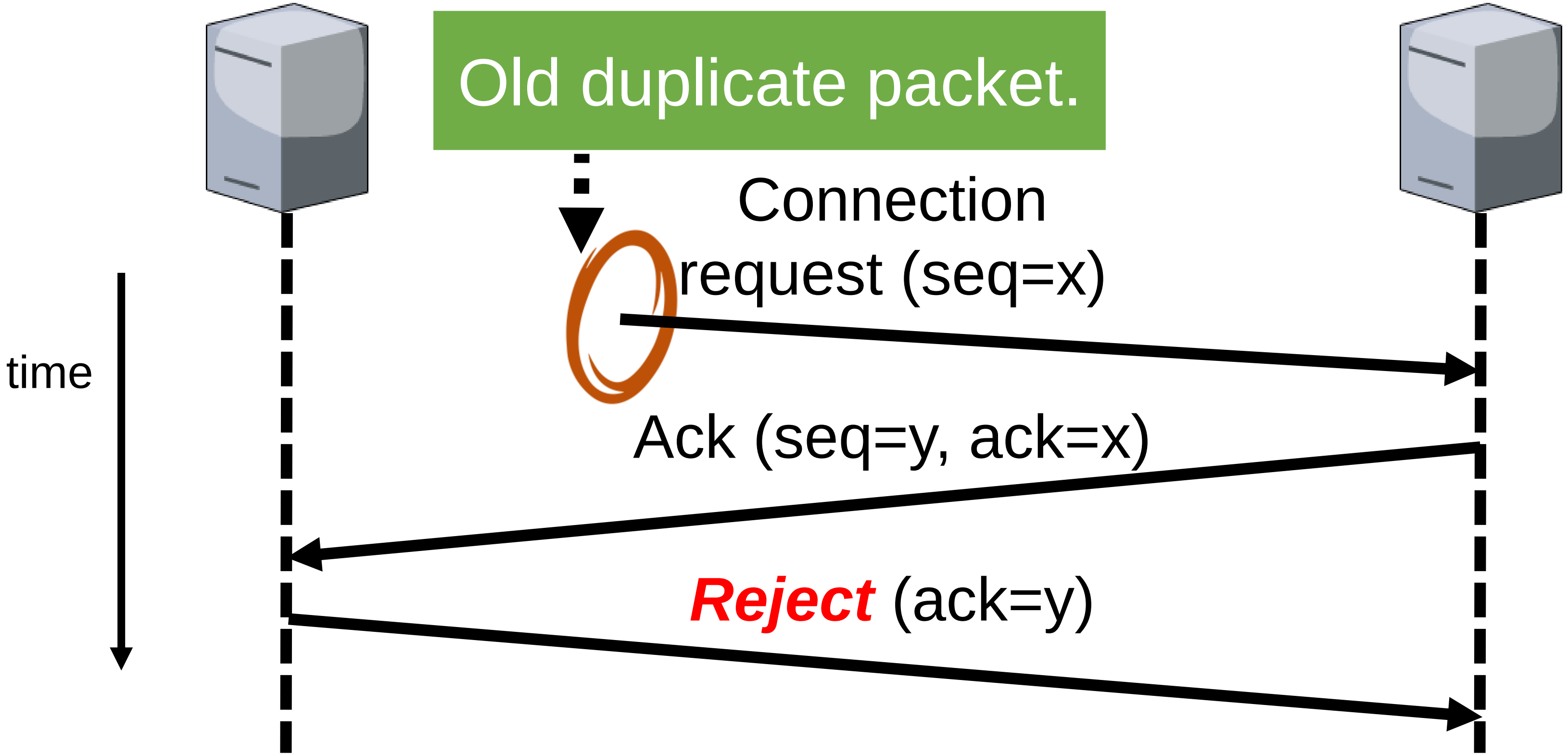
Three-way handsha

TCP uses a (slightly different) three-way handshake!

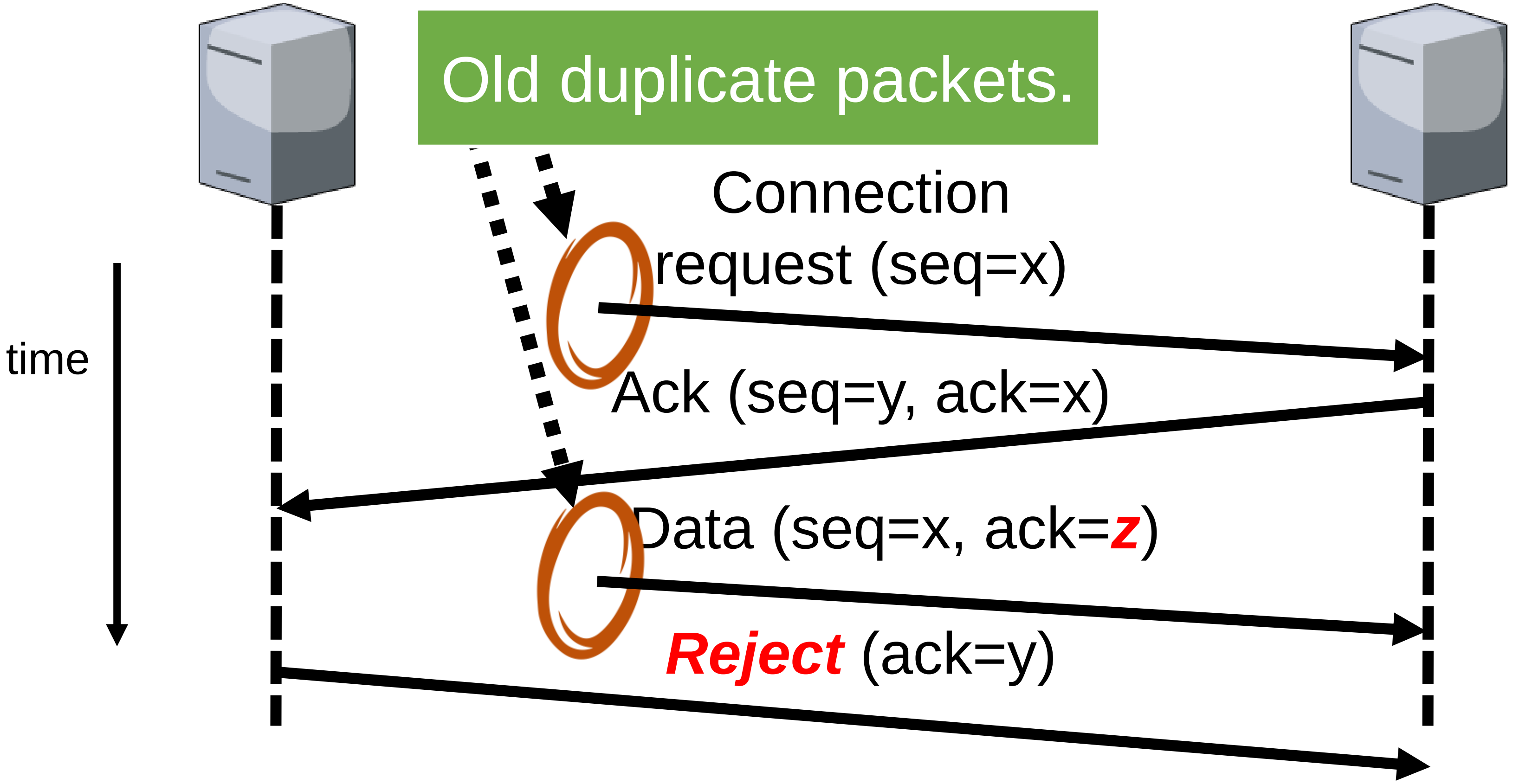
Agreement on which sequence numbers to use.



Three-way handshake handles duplicates



Three-way handshake handles duplicates



Roadmap: Transport Layer

1. Transport layer responsibilities and challenges
2. **Connection establishment and release**
    1. Connection establishment
    2. **Connection release**
3. Revisiting reliable delivery and flow control
4. Congestion control and bandwidth allocation
5. TCP and UDP

Connection release

When the exchange is complete, the connection should be closed.

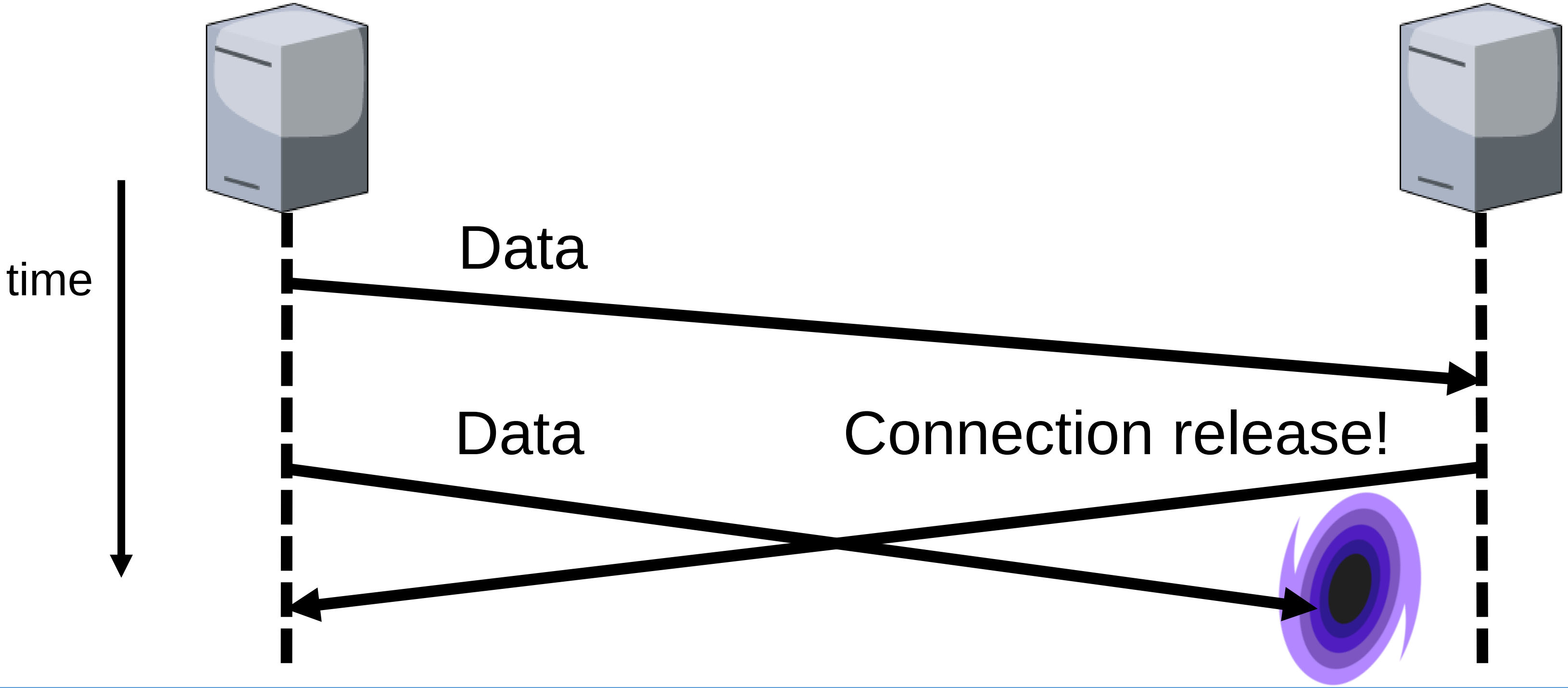
Two approaches:

1. Asymmetric disconnect.
2. Symmetric disconnect.

Asymmetric connection release

Connection ended by either participant without agreement.

May result in data loss!

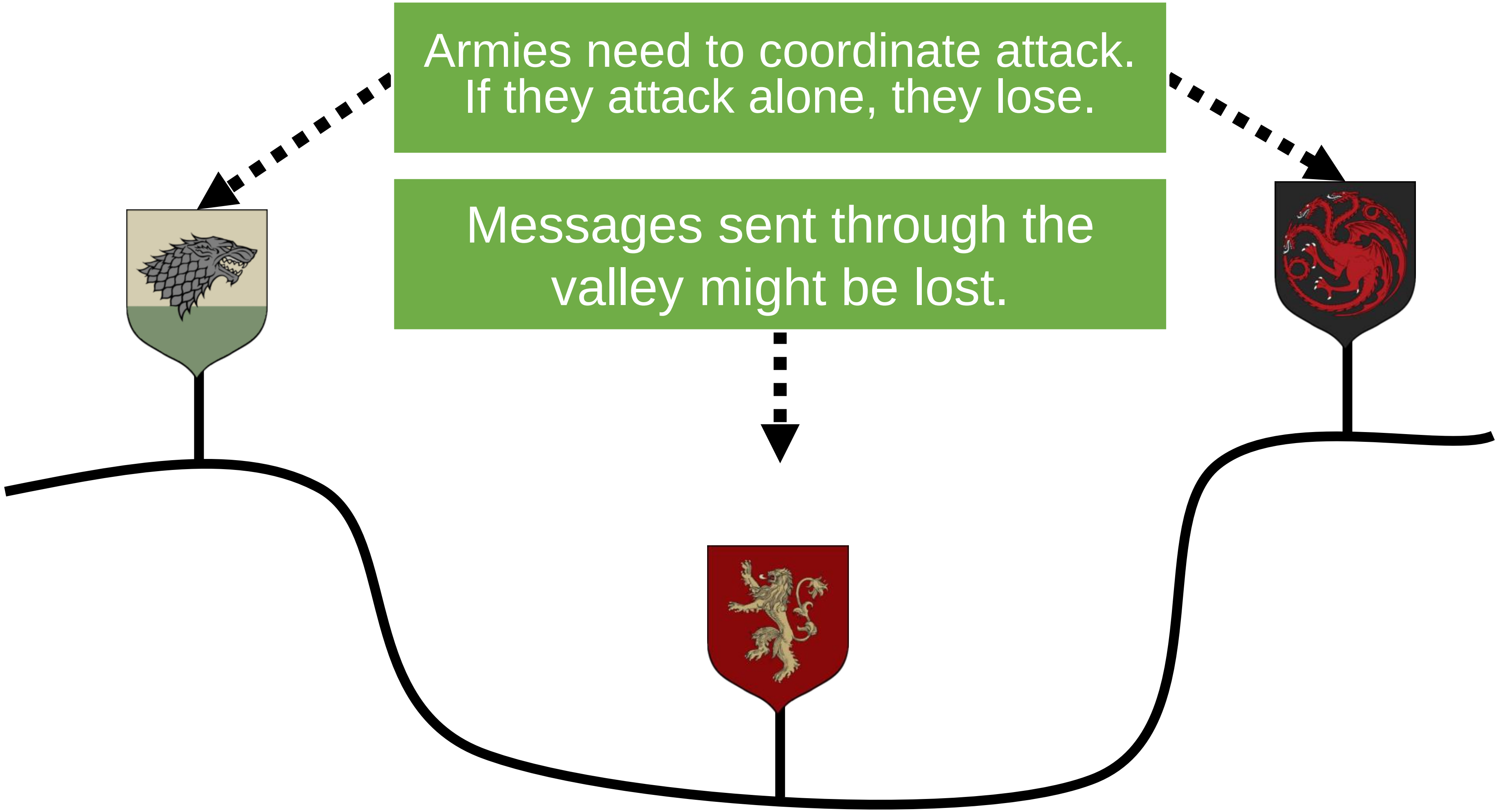


Symmetric connection release

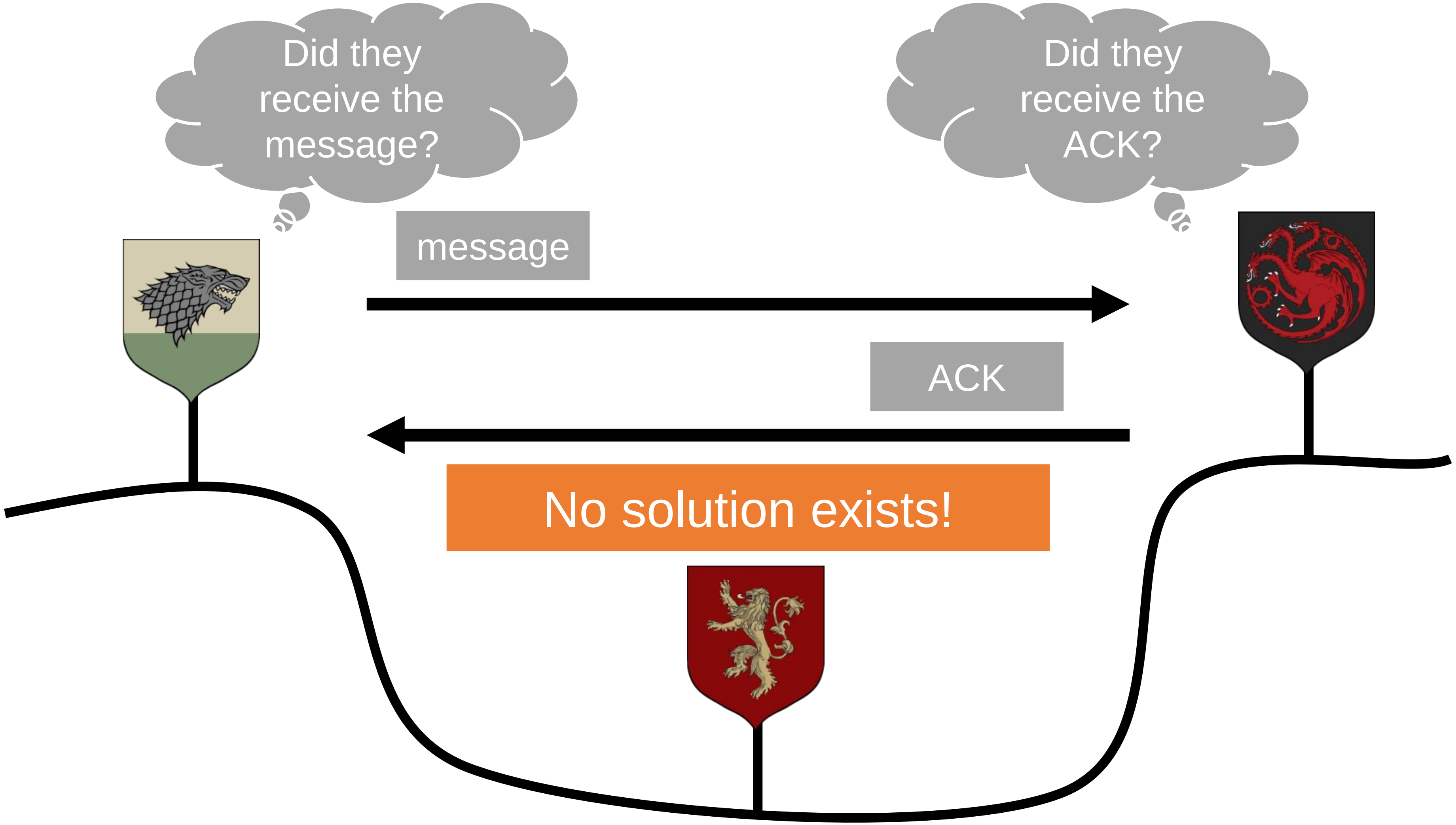
Participants agree to end connection.

More difficult than it sounds!

The two armies problem

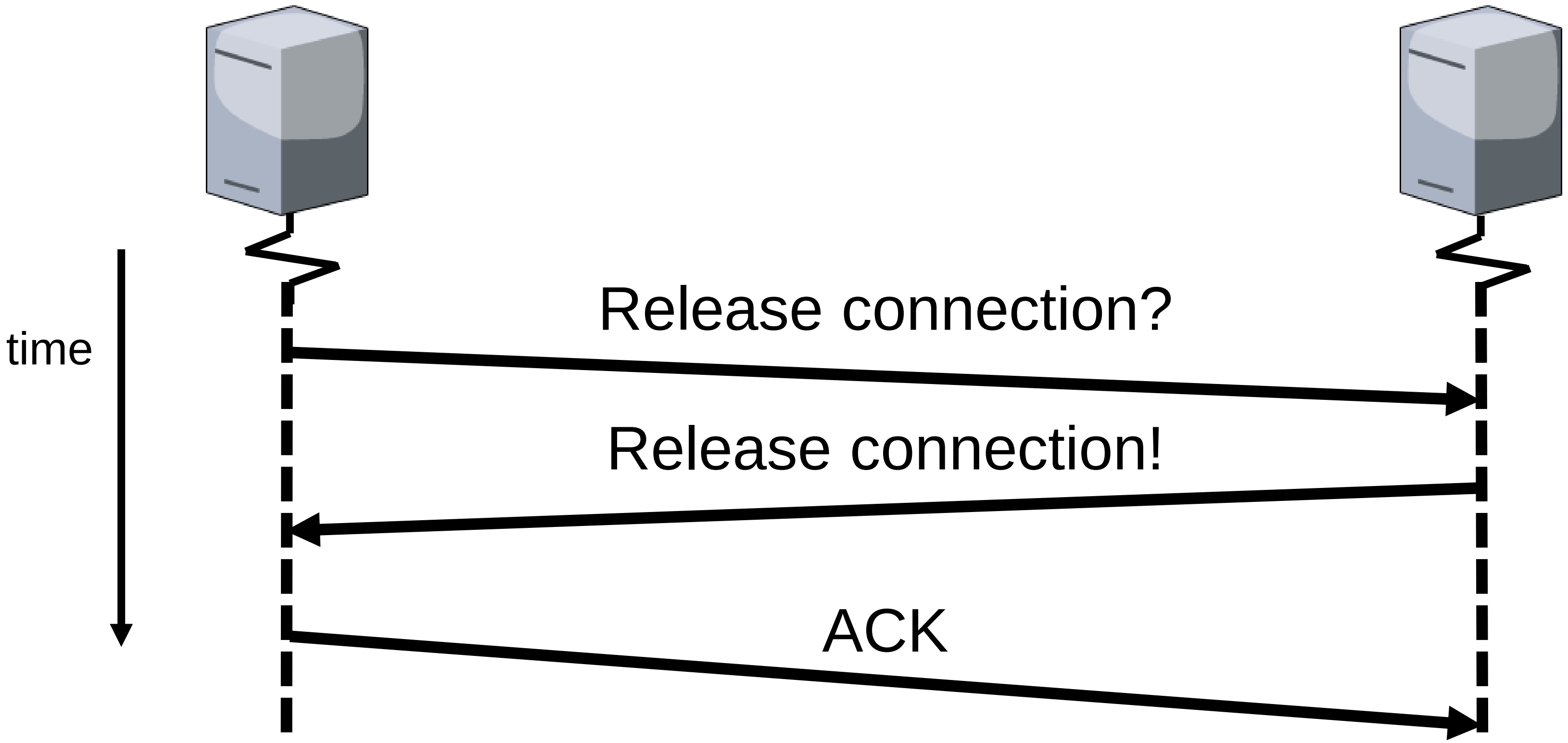


The two armies problem



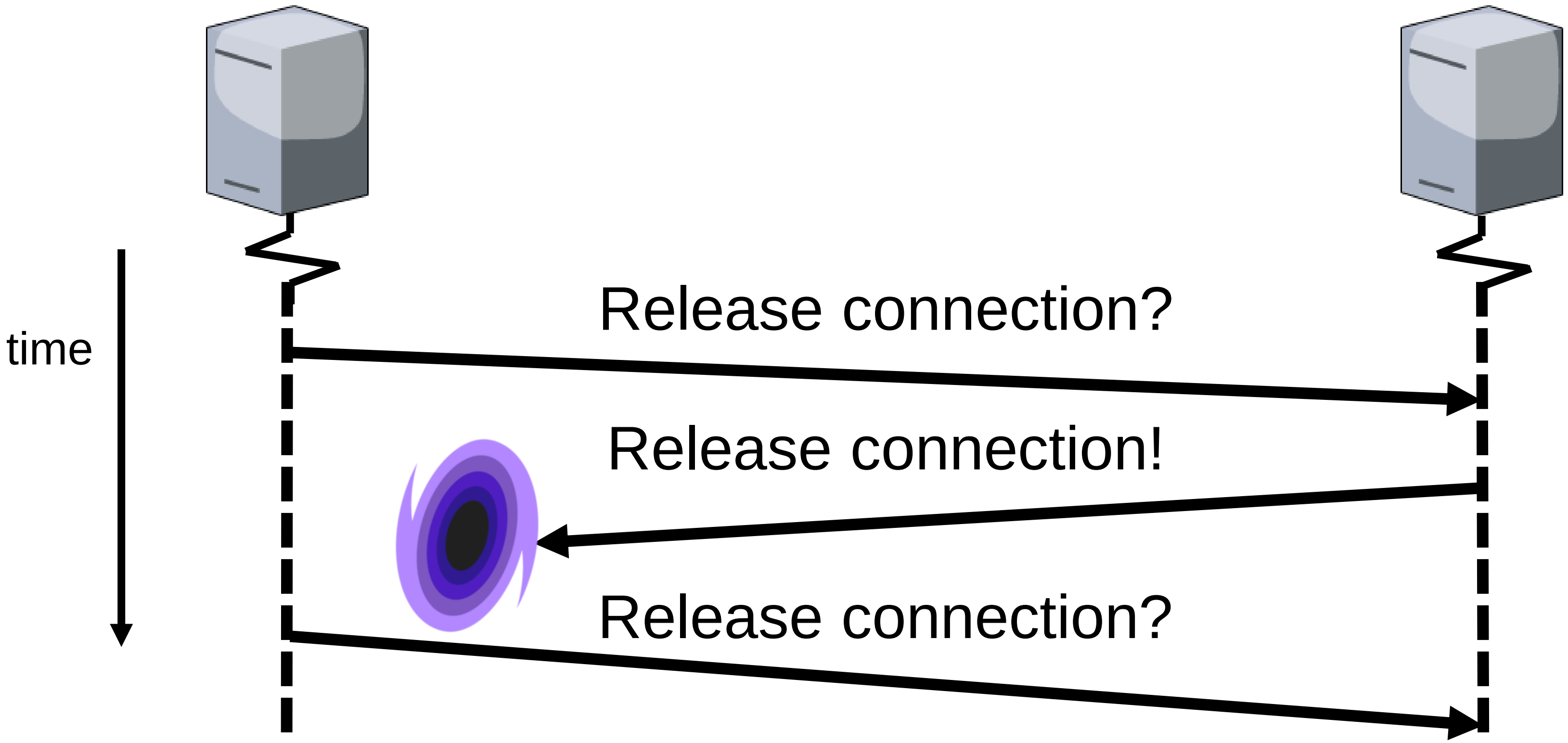
Symmetric connection release

Participants agree to end connection.



Symmetric connection release

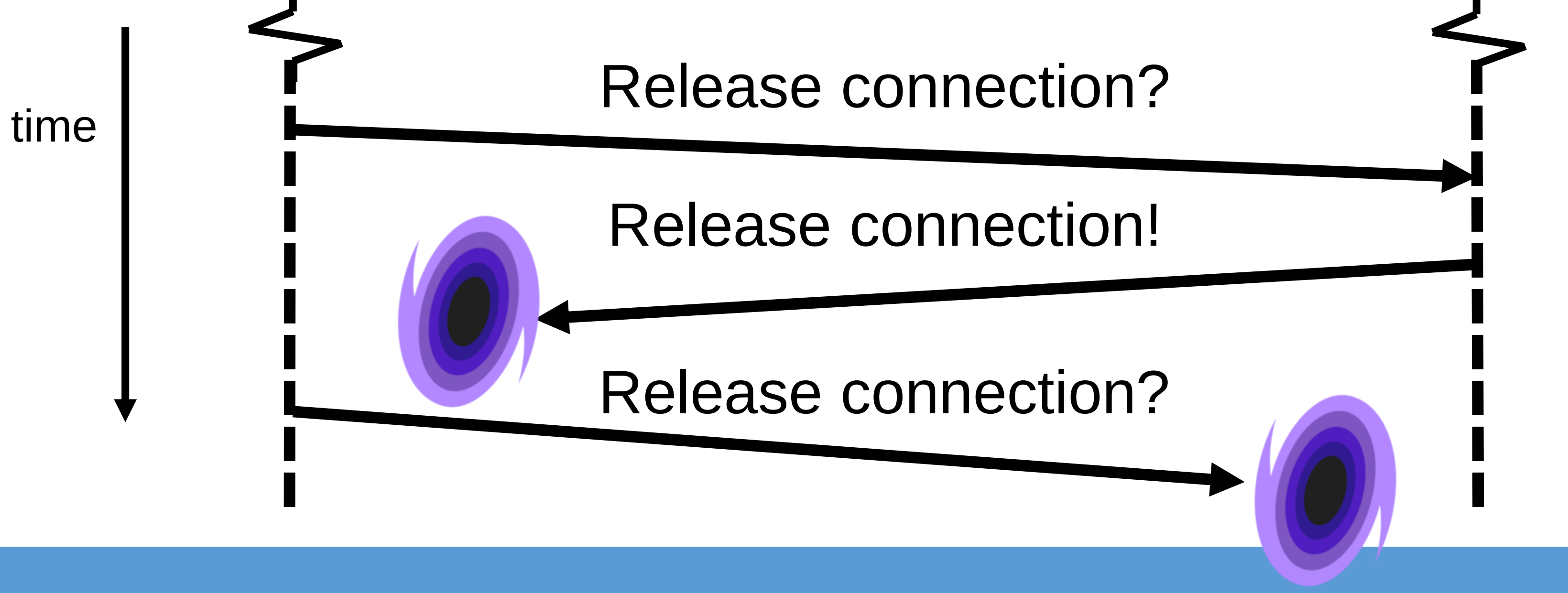
Participants agree to end connection.



Symmetric connection release

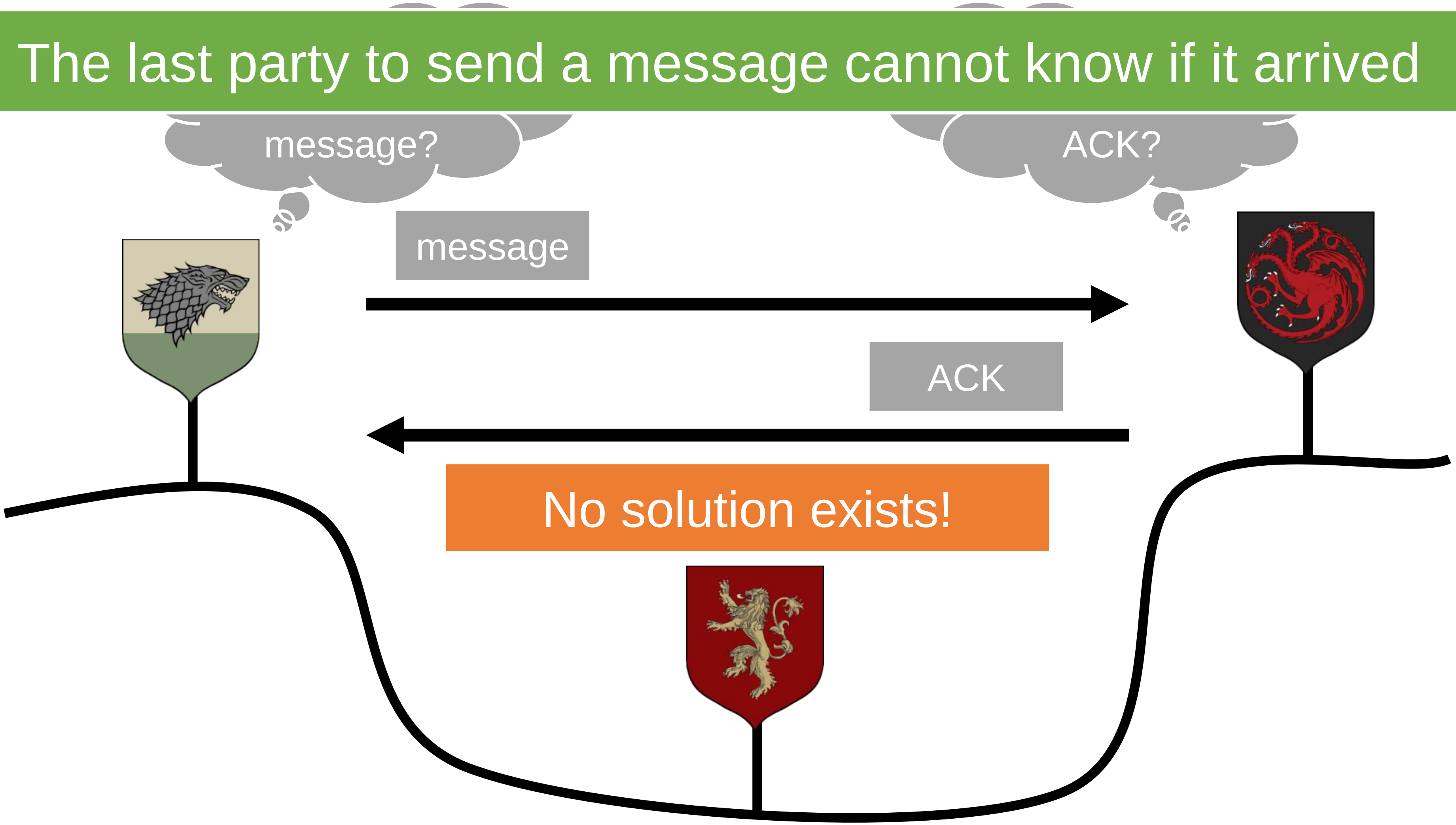
Participants agree to end connection.

Q: How do we solve this in practice?



The two armies problem

The last party to send a message cannot know if it arrived



No solution exists!