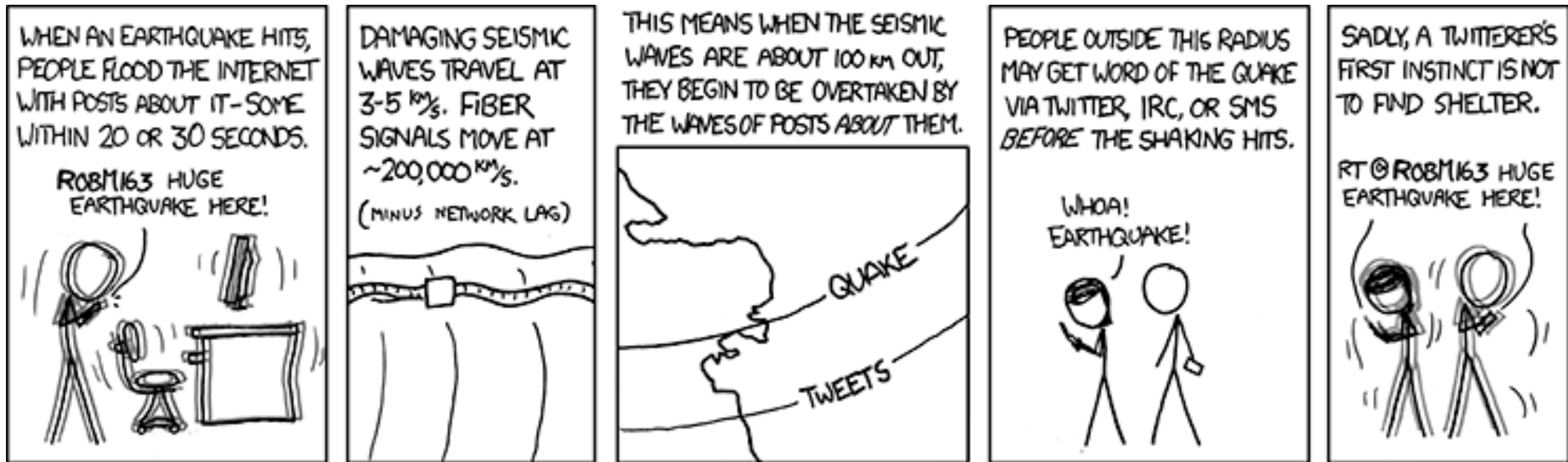


xkcd.com



End To End Protocols

COS 460 & 540

End to End Protocols



This section is about

Process to Process
communications.

or the **how** applications can talk
to each other.

Requirements

- **Guarantee Delivery**
- **Deliver in order**
- **Deliver at most one copy**
- **Any size**
- **Flow control & Synchronization**

The Network May...

- Drop messages
- Reorder messages
- Deliver duplicate copies
- Limit message size
- Delay messages

End To End Protocols

The network provides a **best-effort** service



This layer needs to provide
high level services

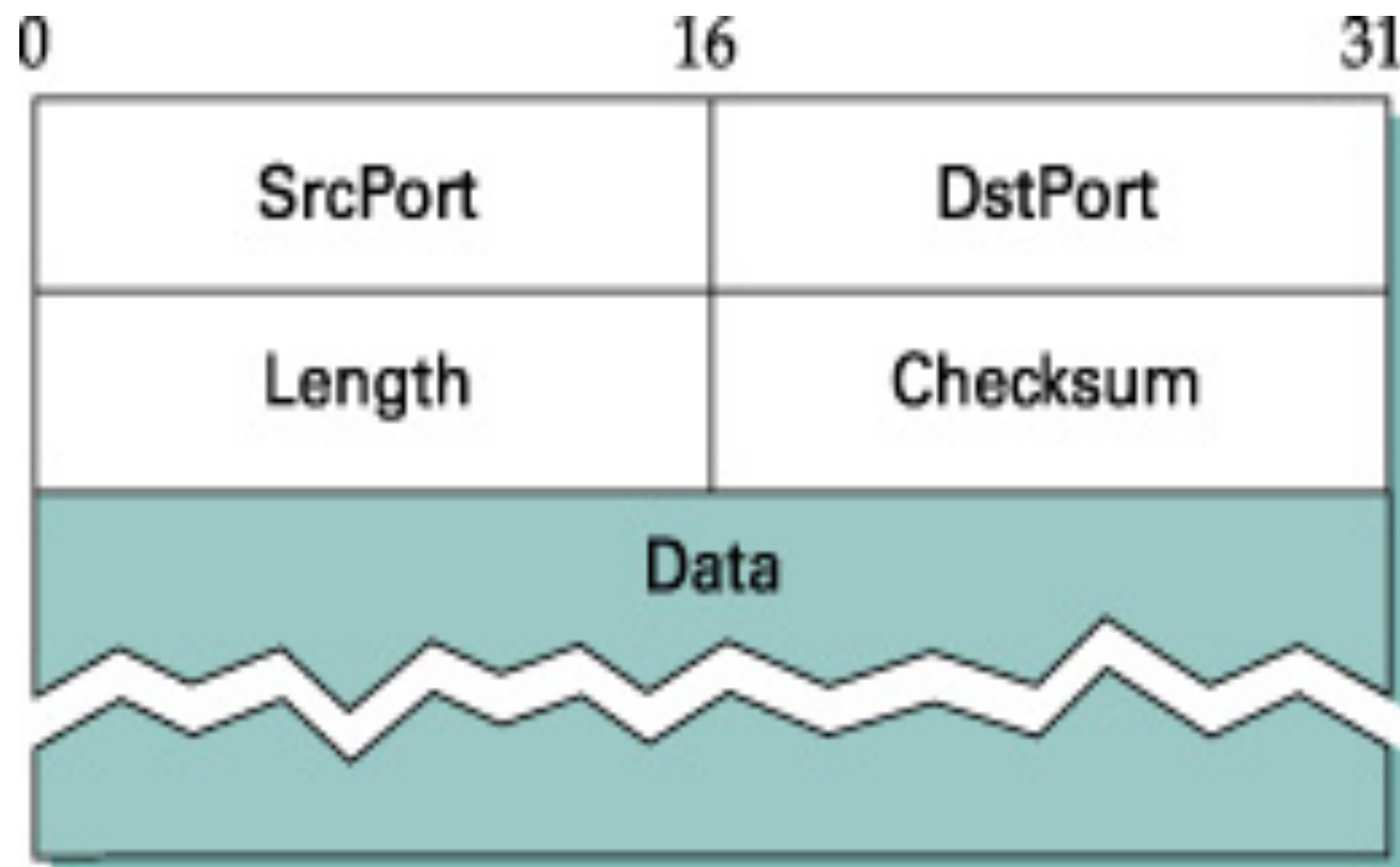
End To End Protocols

- ★ Simple Demultiplexer (UDP)
- Reliable Stream (TCP)
- Remote Procedure Call (RPC)
- Real-time Applications (RTP)
- Performance

User Datagram Protocol (UDP)

- Thin layer over Network Layer
(unreliable, best-effort service)
- Simple demultiplexing; Adds **Process Address** or **port**
- Checksum (packet correctness)

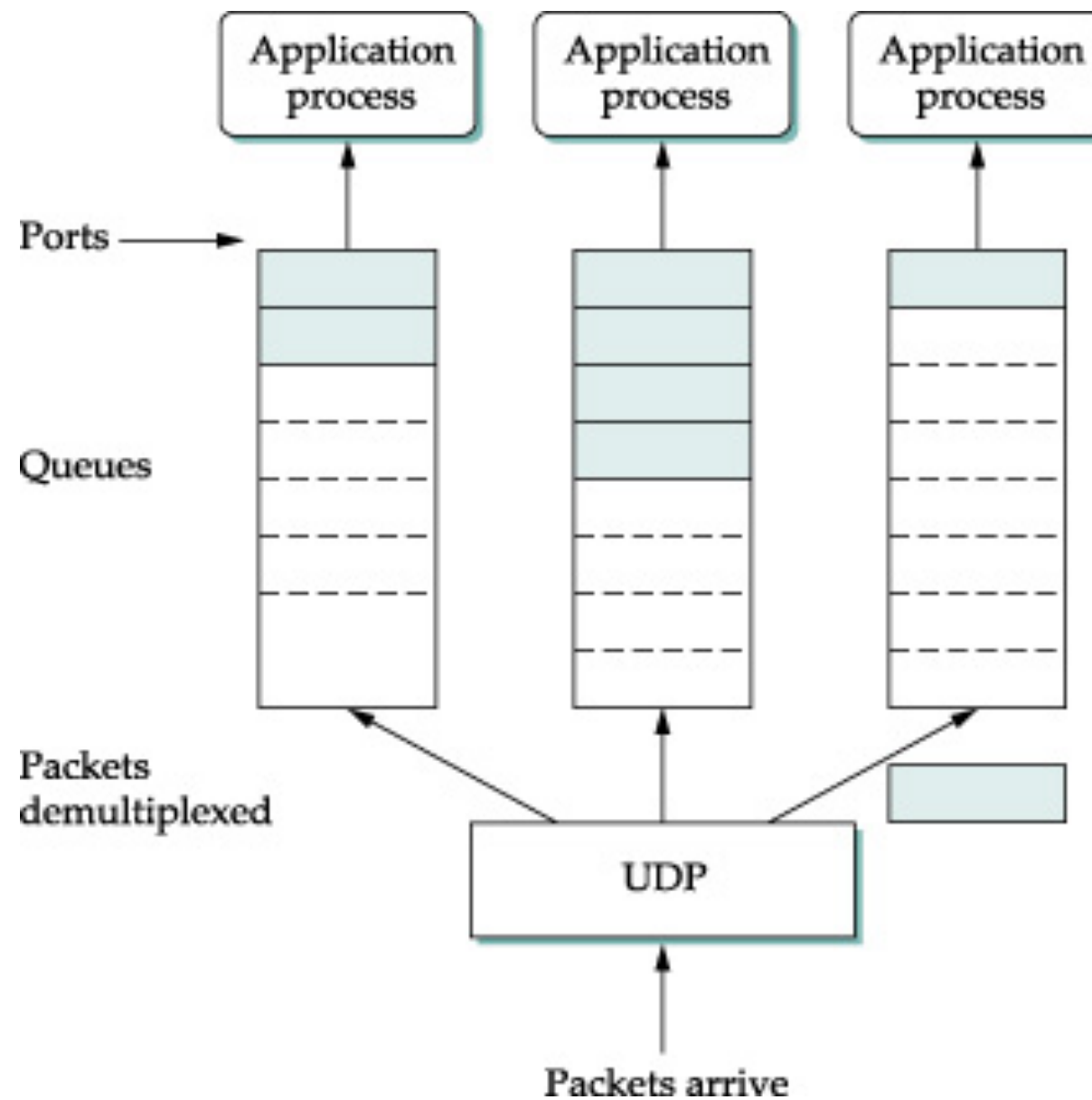
UDP



Ports

- 16 bit port number = 64,536 ports
- Unique to a host (IP address)
 - for example: 192.168.1.2:80
- *well-known* ports
 - 22, 25, 80, ...
- <1,024 usually requires administrator

UDP Demultiplexing



Socket Pairs

Every datagram has a unique “pairing”

- source ip address
- source port
- destination ip address
- destination port

End To End Protocols

- ✓ Simple Demultiplexer (UDP)
- ★ Reliable Stream (TCP)
 - Remote Procedure Call (RPC)
 - Real-time Applications (RTP)
 - Performance

Reliable Byte Stream

Provide a reliable, in-order, at most once delivery mechanism with flow control and synchronization.

... sounds easy right?

Transmission Control Protocol (TCP)

- Byte-oriented
- Connection-oriented
- Reliable, guaranteed delivery
- In-order delivery
- full-duplex channel

End-To-End Issues

- Logical connections over the great unknown network
- No “single path,” each packet is routed individually.
- Solve by using Sliding Window Protocol with connection setup and teardown

End-To-End Issues

- Round Trip Time (RTT) varies widely
 - even over short-lived connections
- Solve by using *adaptive retransmission*

End-To-End Issues

- Packets are likely to get delivered out of order
- TTL removes packets from the network
- Old packets suddenly show up later
- Solve using Sliding Window w/sequence numbers

End-To-End Issues

- Bandwidth at endpoints and links between them may be vastly different
- Solve by using a flow control mechanism

End-To-End Issues

- The Sender may be able to generate large amounts of data
- Intermediate links can become congested
- Solve using congestion control mechanisms

Segmentation

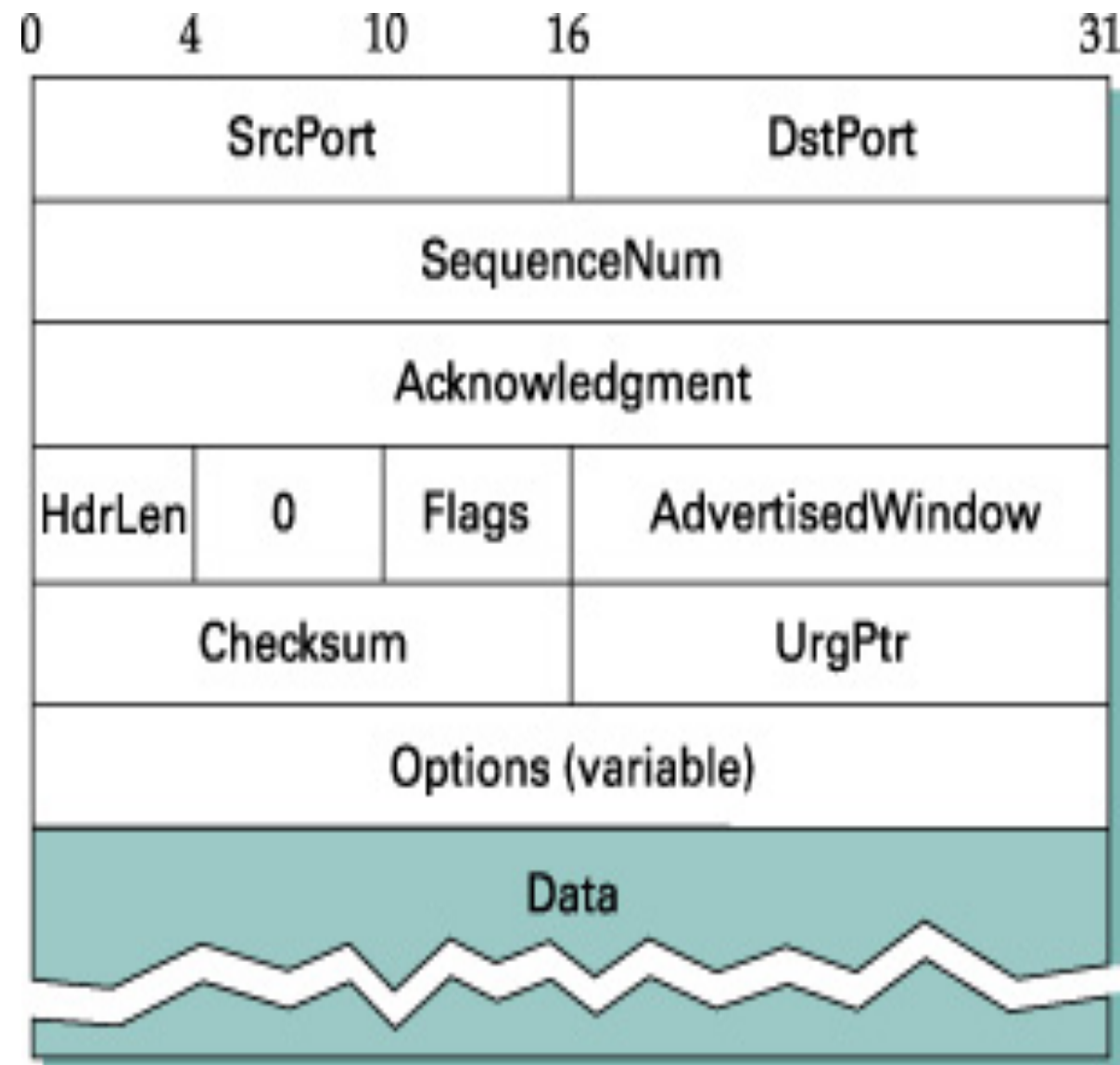
TCP may be a ***byte-stream*** protocol...

...but the network is ***packet based***.

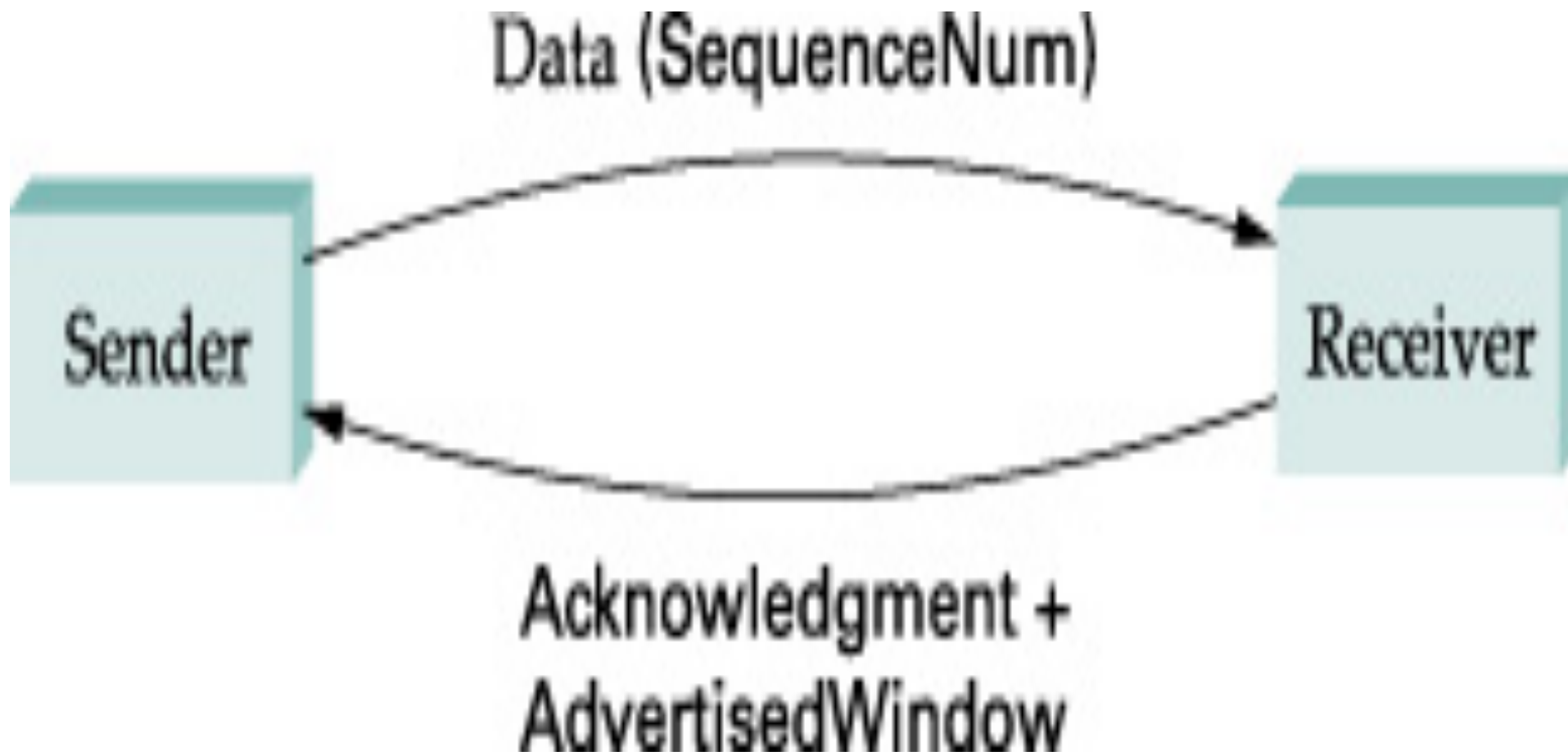
TCP Segment Format

- TCP is byte-oriented
 - buffers bytes for transmission
 - send when full or flushed
- Send bytes in **segments**
- **Acknowledgments** and **windows**

TCP Segment Format



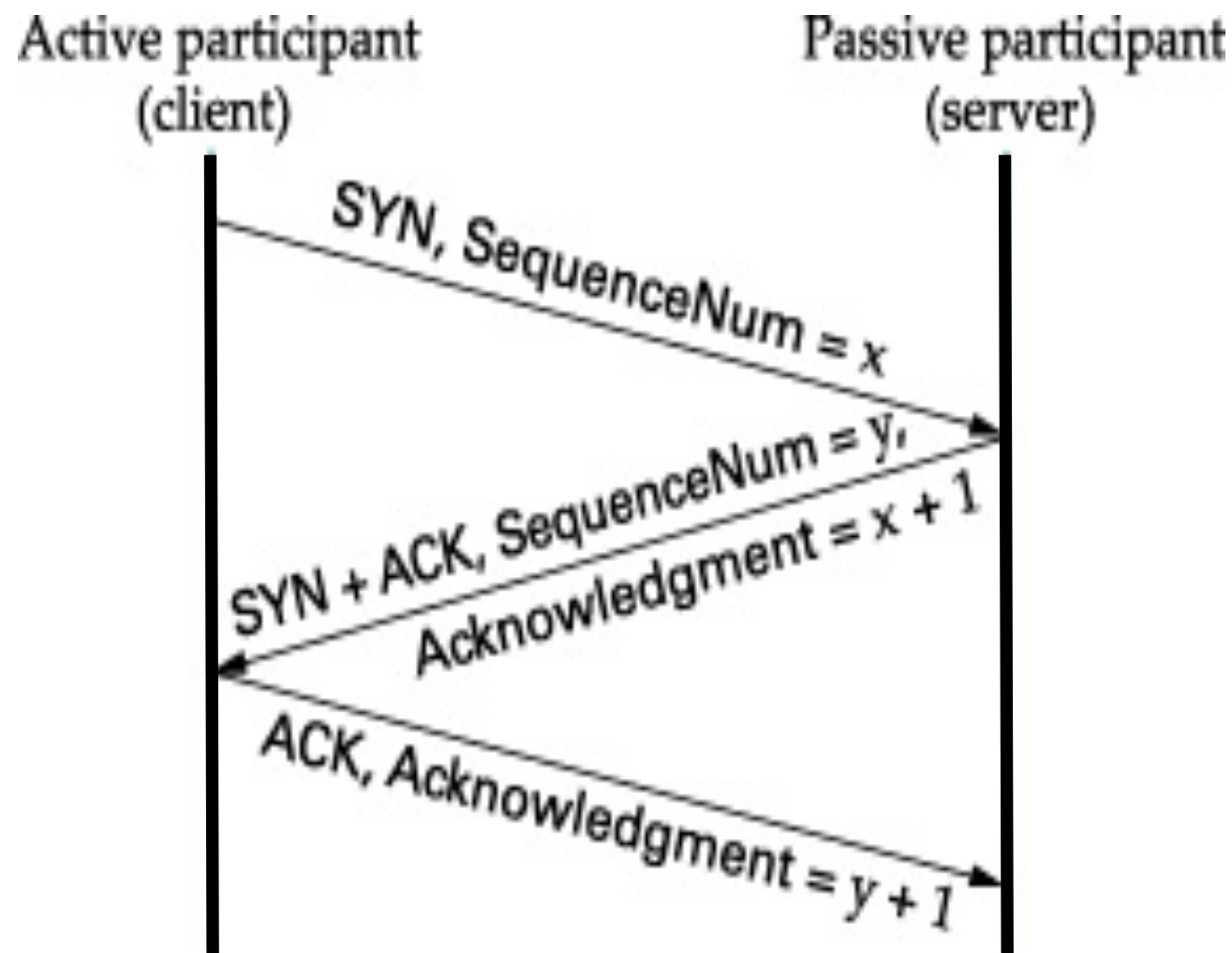
TCP Flow



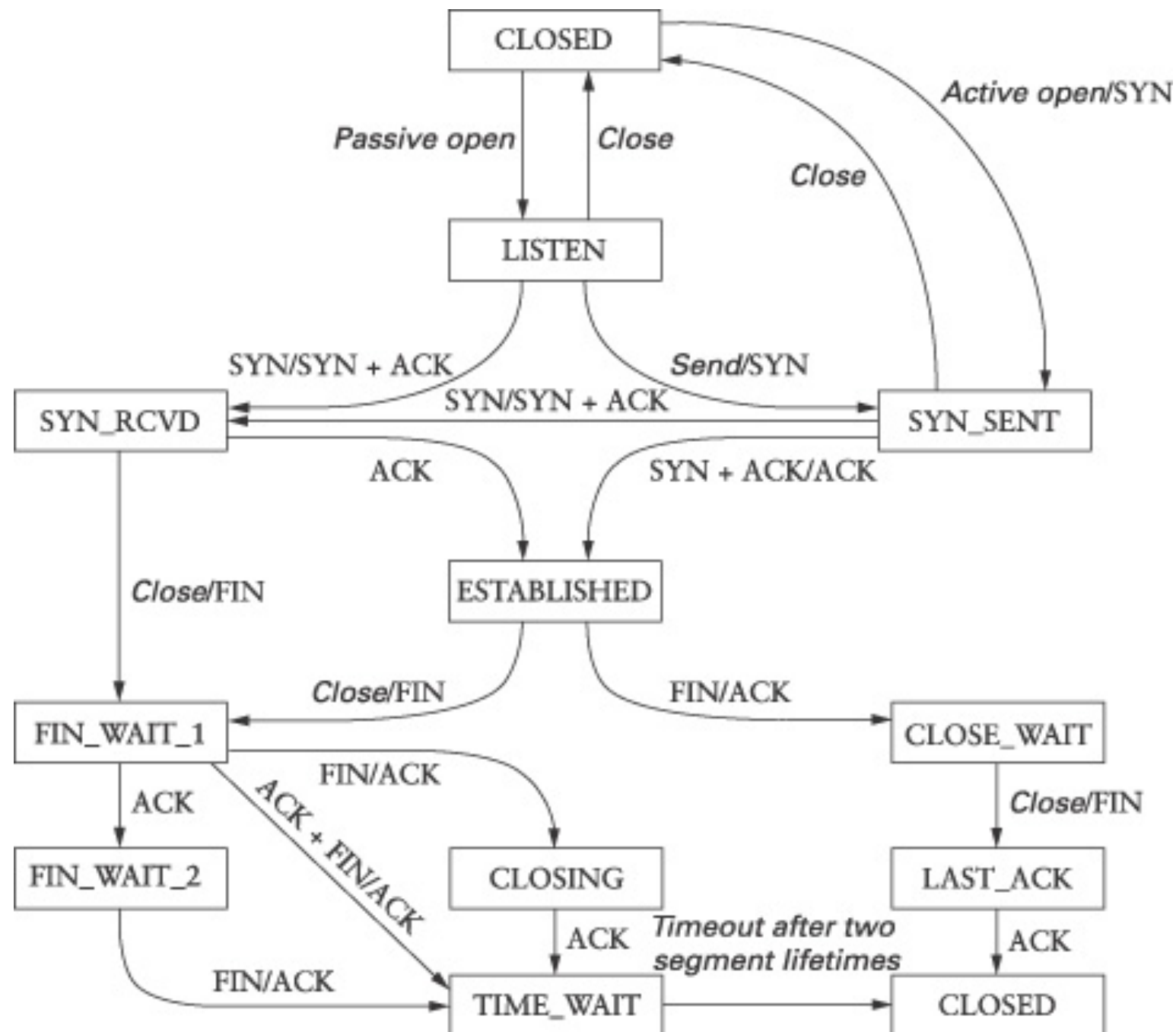
TCP Segment

- Two way communication
- SequenceNum = first byte in segment
- Acknowledgement = next seq. expected
- AdvertisedWindow = buffer size left

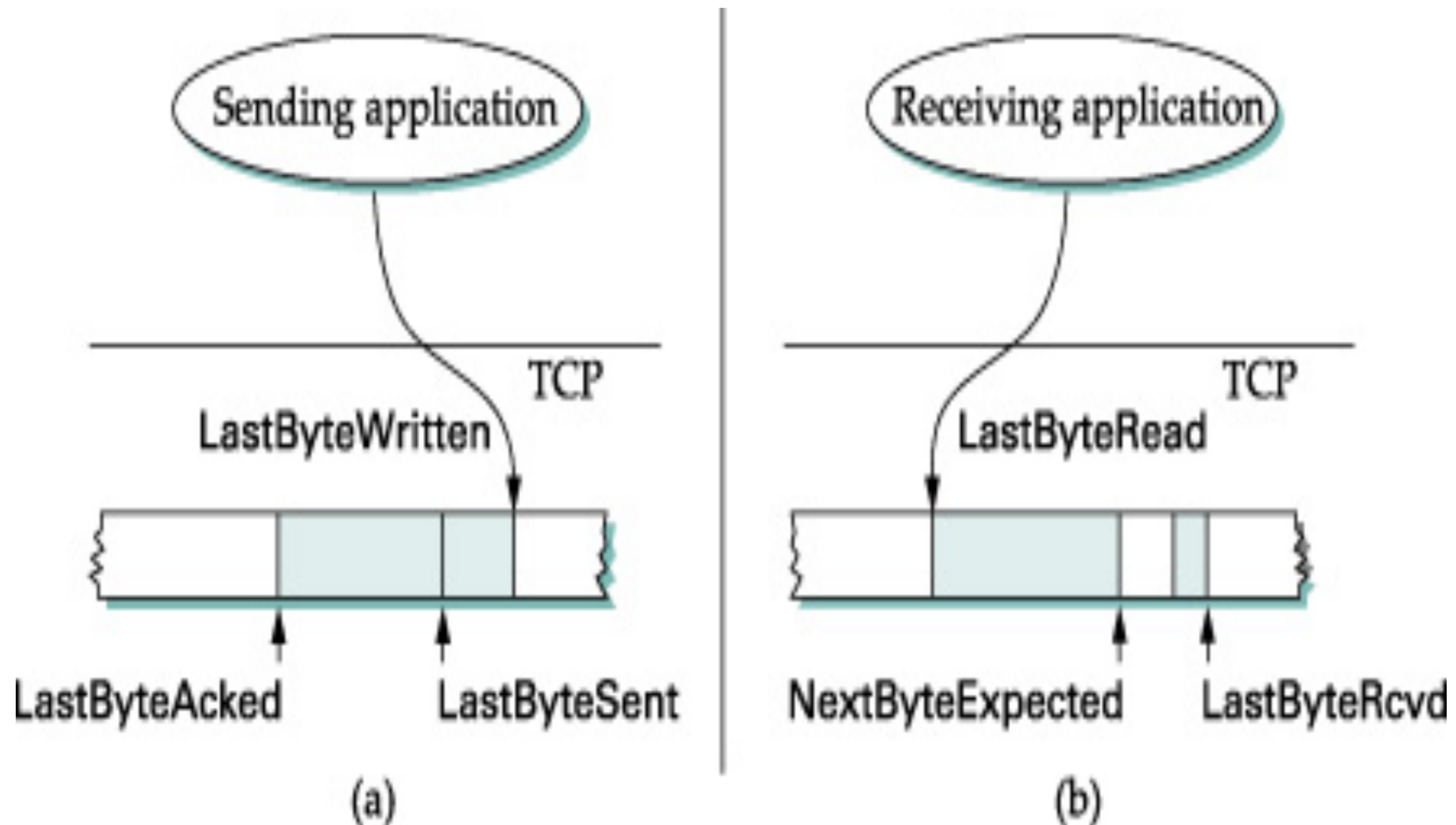
Connection Establishment



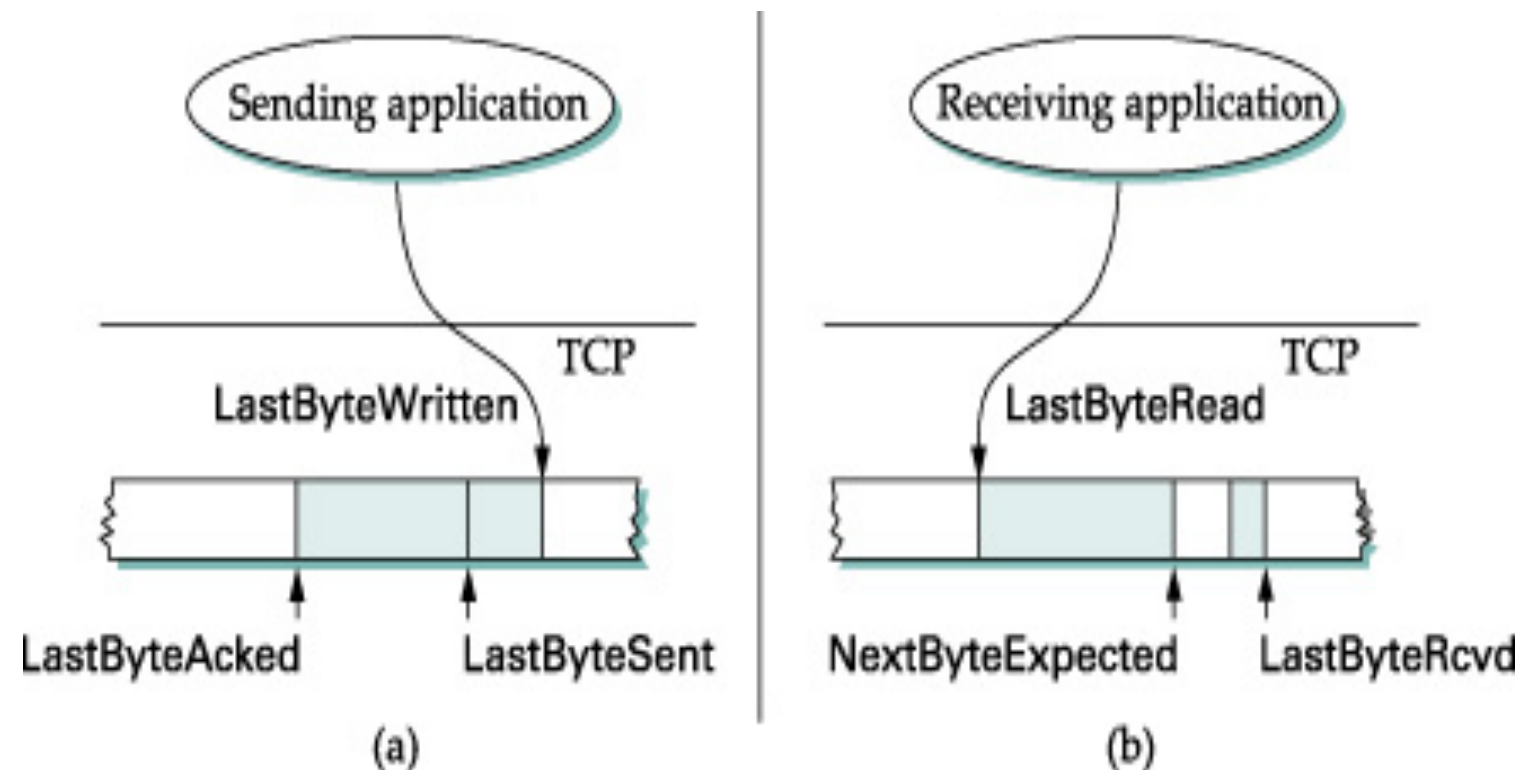
TCP Lifecycle



TCP Sliding Window



TCP Flow Control



Advertised Window

$$\text{LastByteRcvd} - \text{LastByteRead} \leq \text{MaxBuff} =$$
$$\text{MaxB} - ((\text{NextByte} - 1) - \text{LastByteRead})$$

Advertised Window = 0

Sender “pings” with 1 byte segments

Triggering Transmission

- When segment is full ($MSS = MTU$)
- *push* operation
- Timeout expires

Record Boundaries

How would we send a set of database records?

- <name, address, phone,...>, <...>, ...
- UDP = 1 record per datagram

Record Boundaries

In TCP we could....

- Encode special characters, recall ETX?
- Encode the dataset, e.g. XML
- Urgent data flag
- push operation

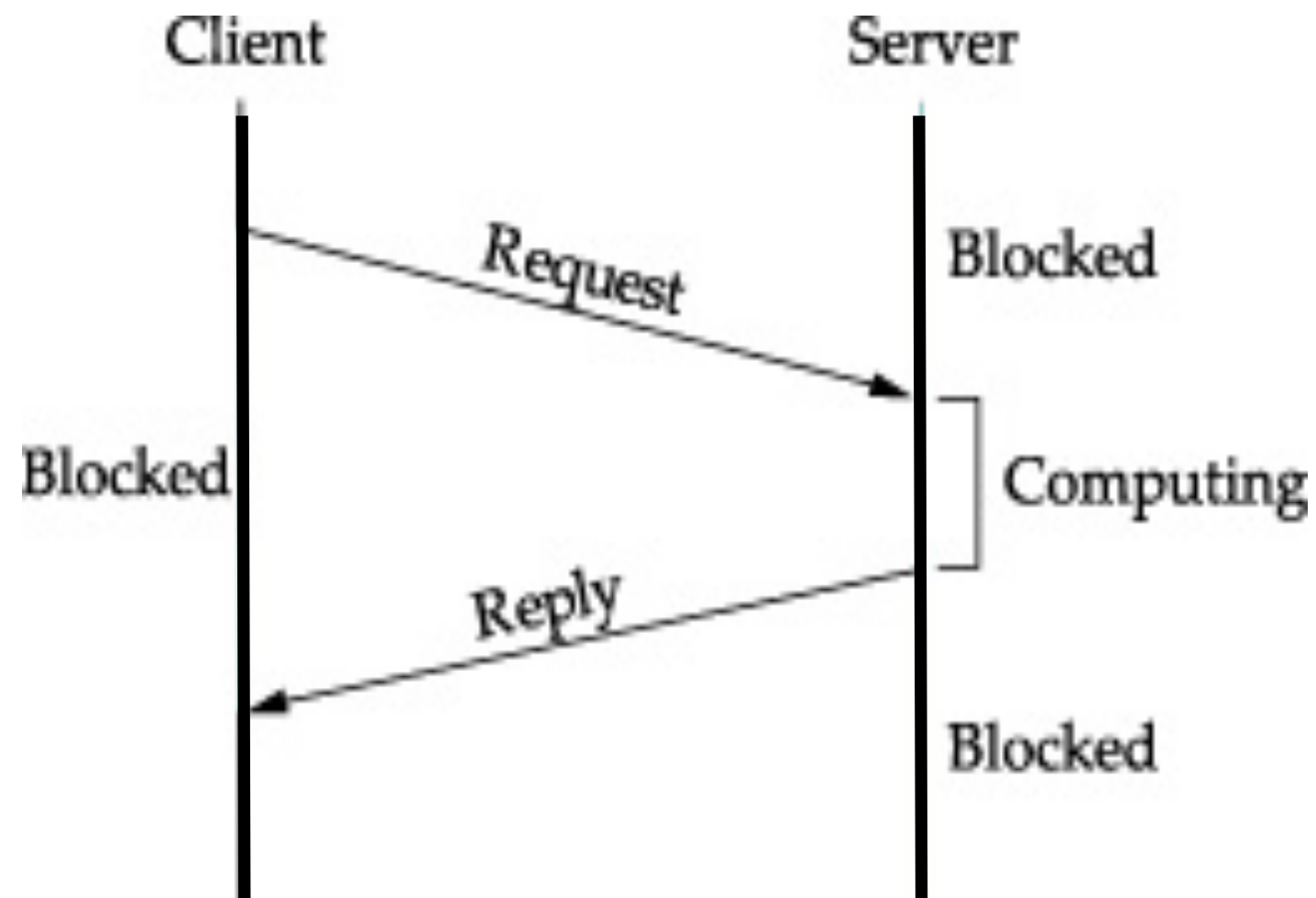
End To End Protocols

- ✓ Simple Demultiplexer (UDP)
- ✓ Reliable Stream (TCP)
- ★ Remote Procedure Call (RPC)
 - Real-time Applications (RTP)
 - Performance

Remote Procedure Call

- Request/Reply Protocol
- Different from UDP?
 - Reliable
 - Sequencing

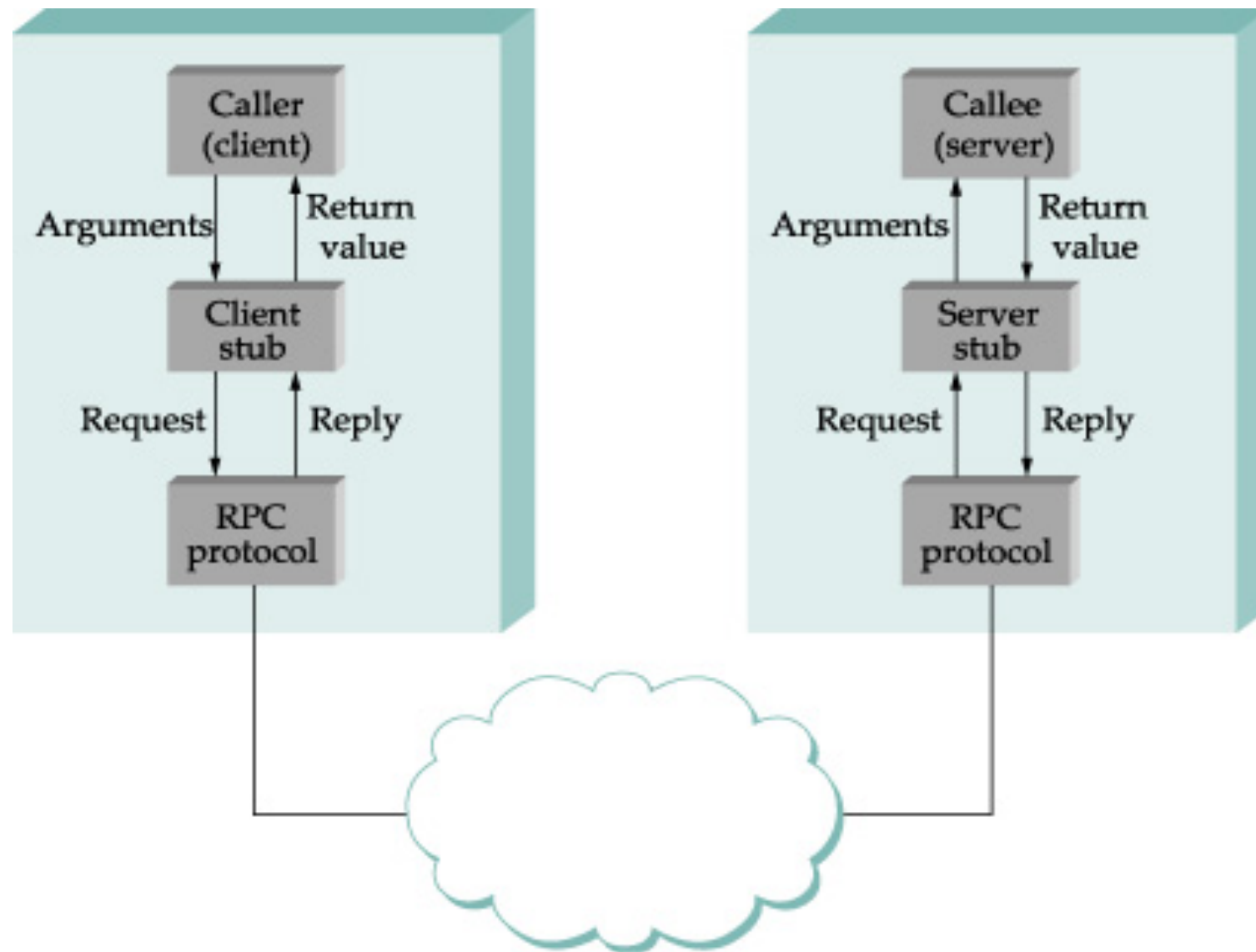
Request Reply



RPC Stack

- Client-Side
 - Generates “stubs”
 - Developer calls stubs
- Server-Side
 - Generates “server”
 - Server calls developers routines

RPC Stack



Serializing Data

- byte, word, character
- string
- array
- tree
- ... more complex data structures

RPC Questions

- RPC over UDP or over TCP or over IP?
- Textual data representation or binary
 - e.g. XML vs XDR

Other RPC-like Protocols

- Common Object Request Broker (CORBA)
- Distributed Component Object Module (DCOM)
- Remote Method Invocation (RMI)
- Simple Object Access Protocol (SOAP)

End To End Protocols

- ✓ Simple Demultiplexer (UDP)
- ✓ Reliable Stream (TCP)
- ✓ Remote Procedure Call (RPC)
- ★ Real-time Applications (RTP)
- Performance

Real-Time Applications

- Encodings... (mp3, aac, etc..)
- Timing of data at receiver
 - Synchronization
- Packet loss
- Frame boundaries

Real-Time Applications

- TCP
 - Has it's own reliable transmission
 - Does not “know” the data
- UDP
 - Does very little, works well

End To End Protocols

- ✓ Simple Demultiplexer (UDP)
- ✓ Reliable Stream (TCP)
- ✓ Remote Procedure Call (RPC)
- ✓ Real-time Applications (RTP)
- ★ Performance

End To End Protocols

- ✓ Simple Demultiplexer (UDP)
- ✓ Reliable Stream (TCP)
- ✓ Remote Procedure Call (RPC)
- ✓ Real-time Applications (RTP)
- ✓ Performance

End

End To End Protocols