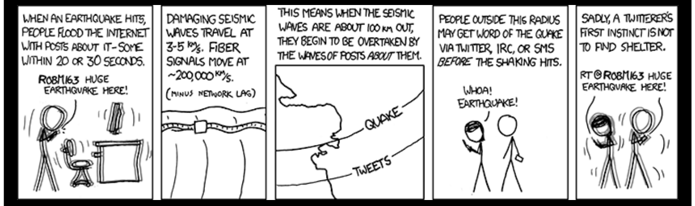
1

## xkcd.com


WHEN AN EARTHQUAKE HITS, PEOPLE FLOOD THE INTERNET WITH POSTS ABOUT IT–SOME WITHIN 20 OR 30 SECONDS.
ROBM163 HUGE EARTHQUAKE HERE!
DAMAGING SEISMIC WAVES TRAVEL AT 3-5 KM/S. FIBER SIGNALS MOVE AT ~200,000 KM/S.
(MINUS NETWORK LAG)
THIS MEANS WHEN THE SEISMIC WAVES ARE ABOUT 100 KM OUT, THEY BEGIN TO BE OVERTAKEN BY THE WAVES OF POSTS ABOUT THEM.
QUAKE
TWEETS
PEOPLE OUTSIDE THIS RADIUS MAY GET WORD OF THE QUAKE VIA TWITTER, IRC, OR SMS BEFORE THE SHAKING HITS.
WHOA! EARTHQUAKE!
SADLY, A TWITTERER'S FIRST INSTINCT IS NOT TO FIND SHELTER.
RT @ROBM163 HUGE EARTHQUAKE HERE!

2

# End To End Protocols

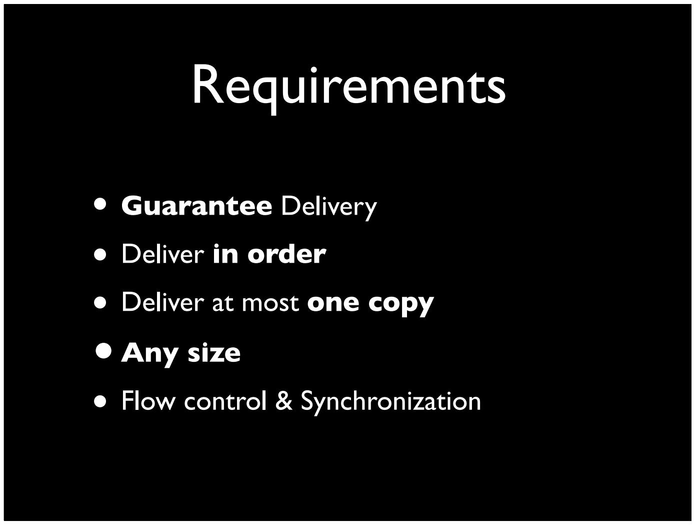
COS 460 & 540

3

## End to End Protocols

This section is about **Process to Process** communications.

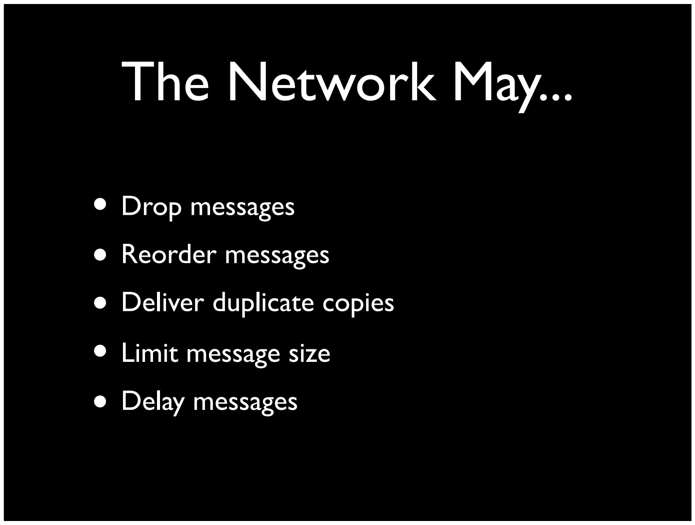
or the **how** applications can talk to each other.



4

## Requirements

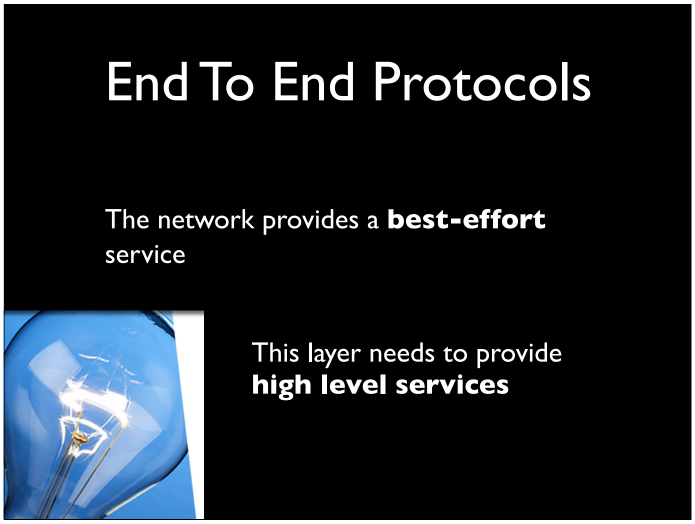
- **Guarantee** Delivery
- Deliver **in order**
- Deliver at most **one copy**
- **Any size**
- Flow control & Synchronization



5

## The Network May...

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



6

## End To End Protocols

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

This layer needs to provide **high level services**

7

# End To End Protocols

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

8

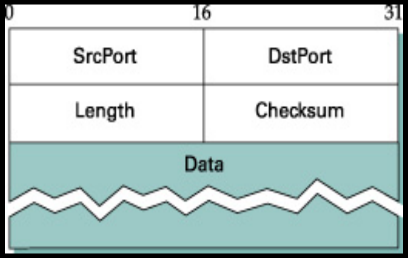
# User Datagram Protocol (UDP)

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

– port is indirect addressing of process. Process id could be used but is not ubiquitous.

9

# UDP



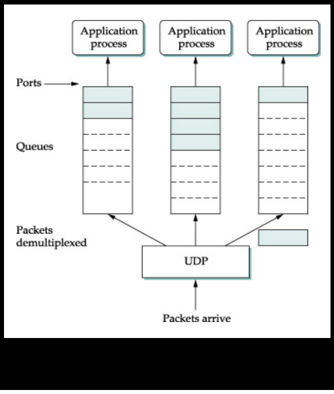
# Ports

- 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

10

– Well known port may be used just for initial connection; then move to another port for communications.
– A port mapper service may also be used

# UDP Demultiplexing



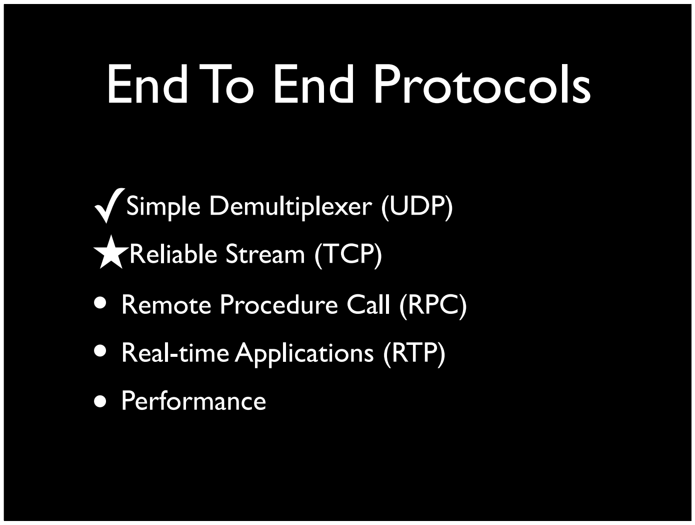
11

# Socket Pairs

Every datagram has a unique “pairing”

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

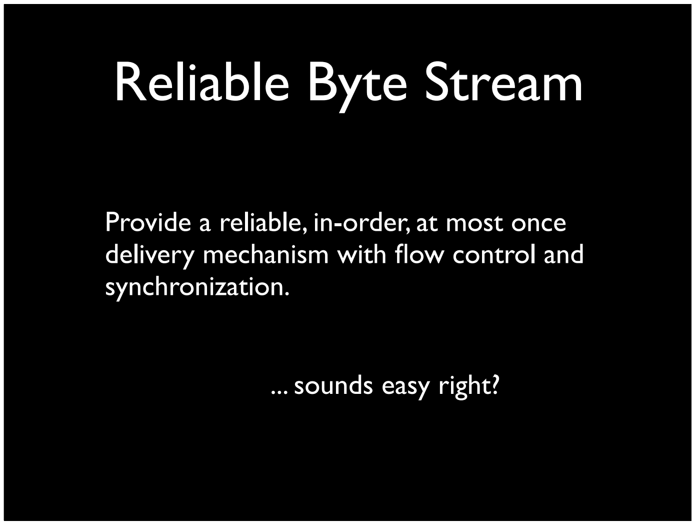
12



13

# End To End Protocols

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

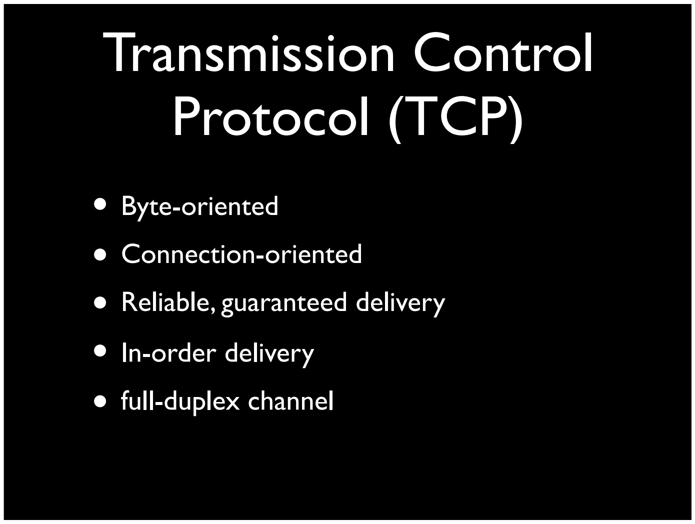


14

# Reliable Byte Stream

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

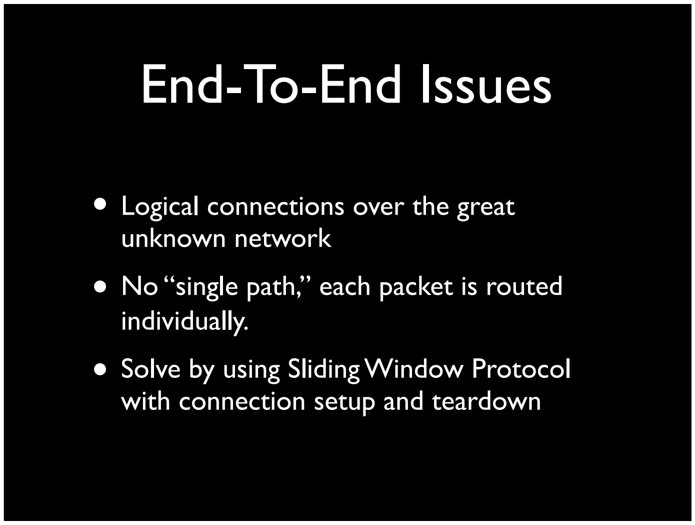
... sounds easy right?



15

# Transmission Control Protocol (TCP)

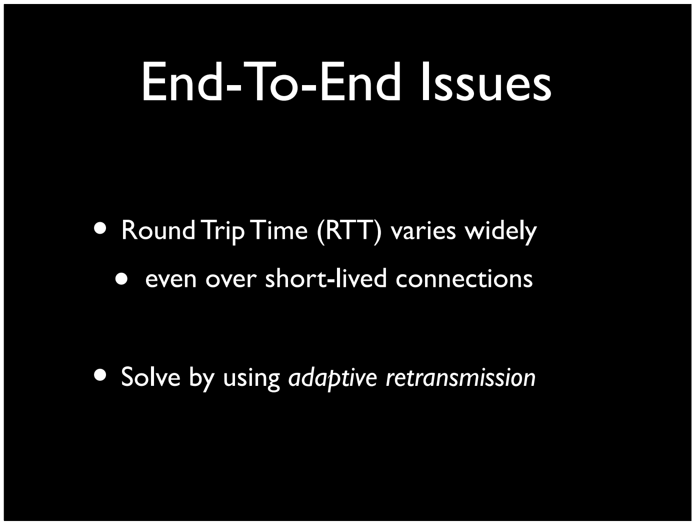
- Byte-oriented
- Connection-oriented
- Reliable, guaranteed delivery
- In-order delivery
- full-duplex channel



16

## End-To-End Issues

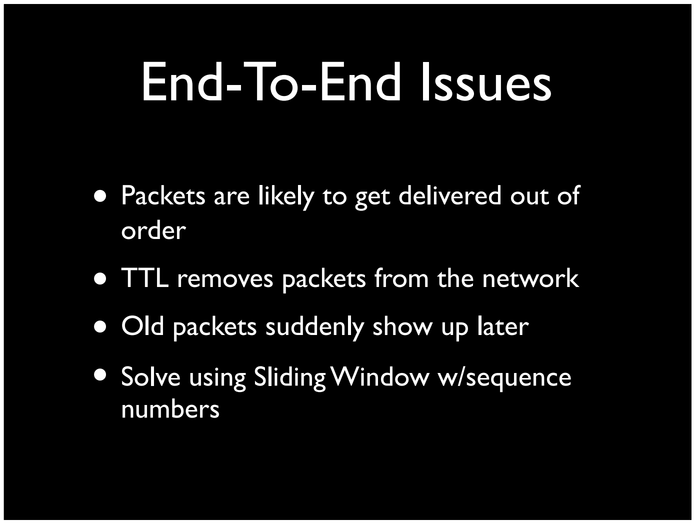
- Logical connections over the great unknown network
- No "single path," each packet is routed individually.
- Solve by using Sliding Window Protocol with connection setup and teardown



17

## End-To-End Issues

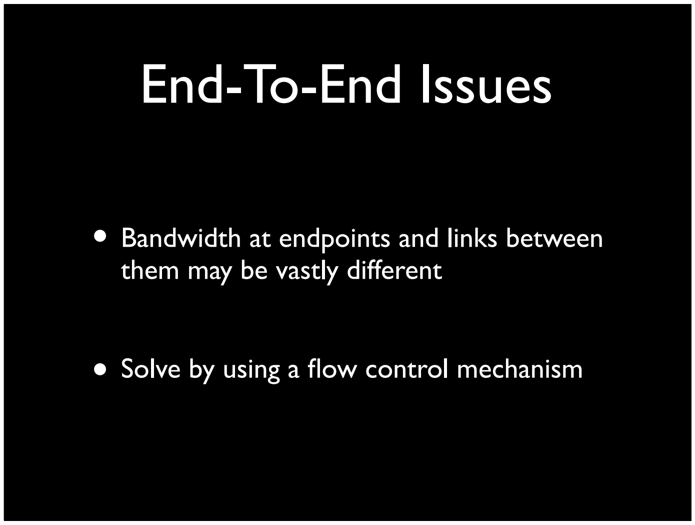
- Round Trip Time (RTT) varies widely
  - even over short-lived connections
- Solve by using *adaptive retransmission*



18

## End-To-End Issues

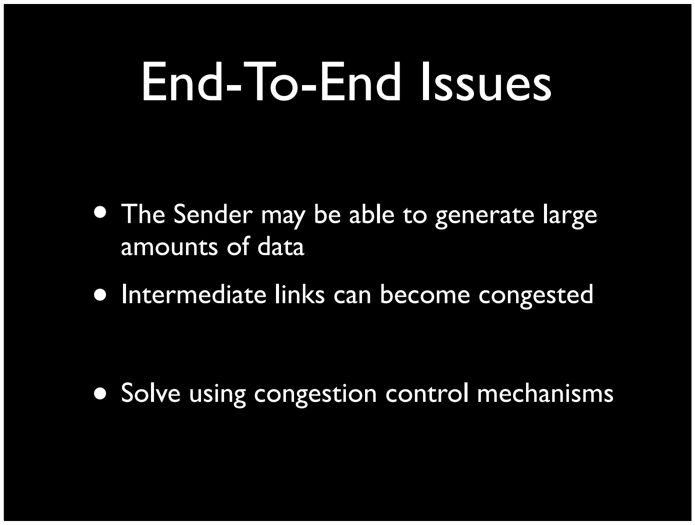
- Packets are likely to get delivered out of order
- TTL removes packets from the network
- Old packets suddenly show up later
- Solve using Sliding Window w/sequence numbers



## End-To-End Issues

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

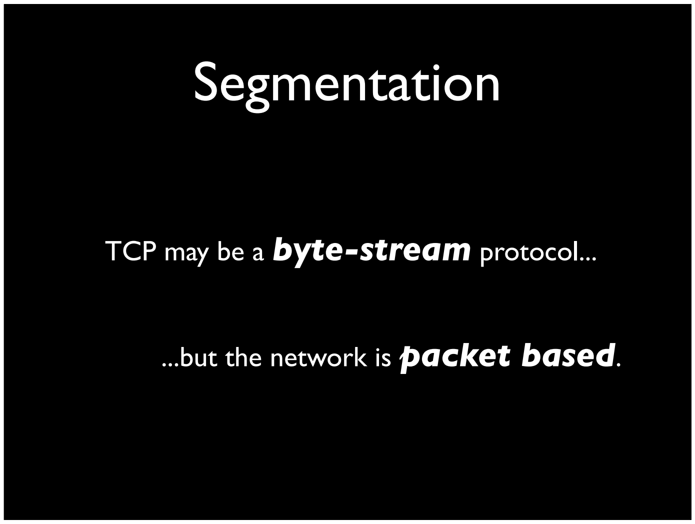
19



## 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

20



## Segmentation

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

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

21

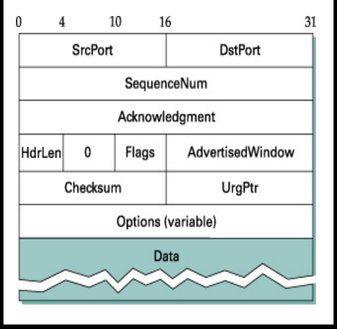
22

# TCP Segment Format

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

23

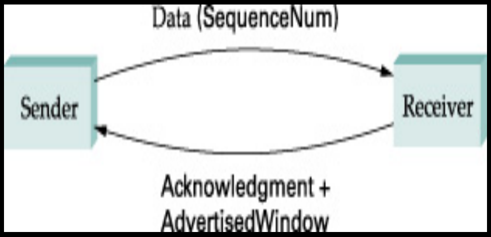
# TCP Segment Format



| 0 | 4 | 10 | 16 31 |
|---|---|---|---|
| SrcPort | | | DstPort |
| SequenceNum | | | |
| Acknowledgment | | | |
| HdrLen | 0 | Flags | AdvertisedWindow |
| Checksum | | | UrgPtr |
| Options (variable) | | | |
| Data | | | |

24

# TCP Flow



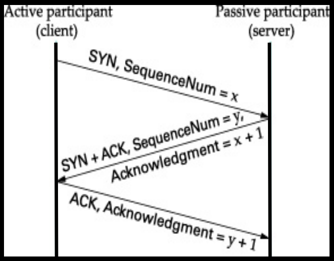
25

# TCP Segment

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

26

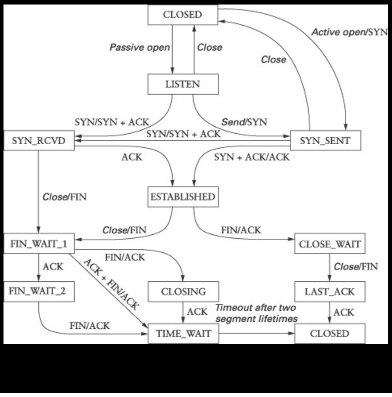
# Connection Establishment

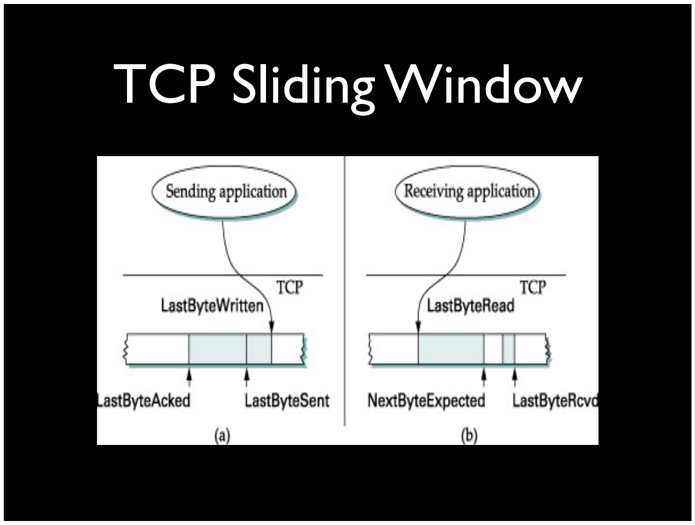

Active participant (client)
Passive participant (server)
SYN, SequenceNum = x
SYN + ACK, SequenceNum = y,
Acknowledgment = x + 1
ACK, Acknowledgment = y + 1

Three Way Handshake

27

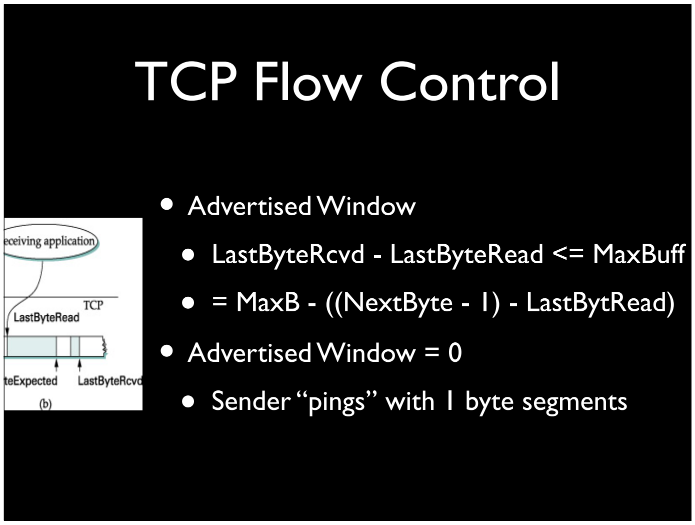
# TCP Lifecycle





# TCP Sliding Window

28

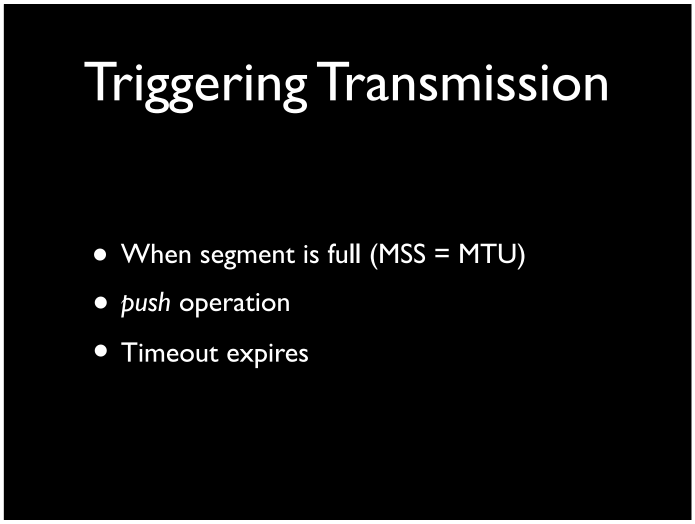


# TCP Flow Control

eceiving application
TCP
LastByteRead
teExpected
LastByteRcvd
(b)

- Advertised Window
  - LastByteRcvd - LastByteRead <= MaxBuff
  - = MaxB - ((NextByte - 1) - LastBytRead)
- Advertised Window = 0
  - Sender “pings” with 1 byte segments

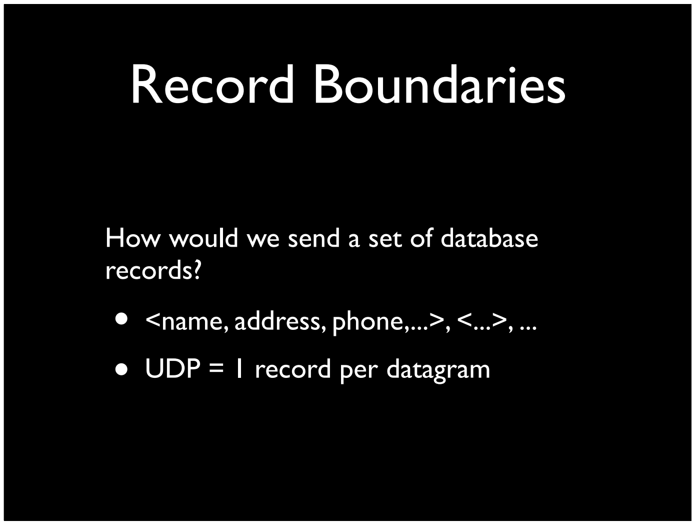
29



# Triggering Transmission

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

30

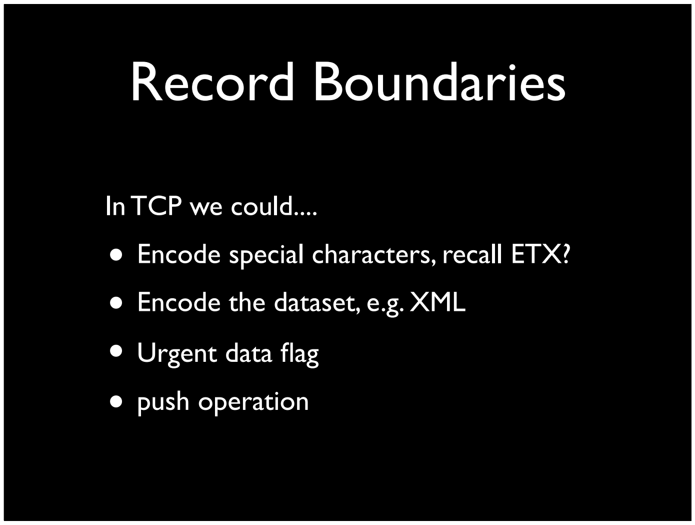


31

## Record Boundaries

How would we send a set of database records?

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

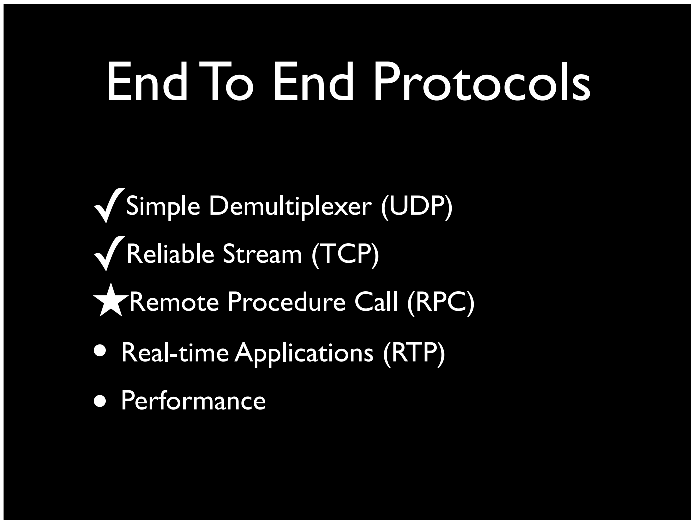


32

## Record Boundaries

In TCP we could....

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



33

## End To End Protocols

✓Simple Demultiplexer (UDP)

✓Reliable Stream (TCP)

★Remote Procedure Call (RPC)

- Real-time Applications (RTP)
- Performance

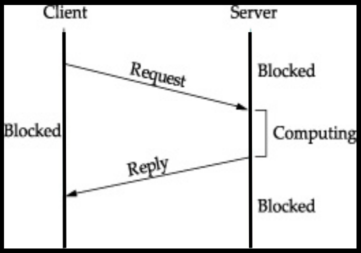
34

# Remote Procedure Call

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

35

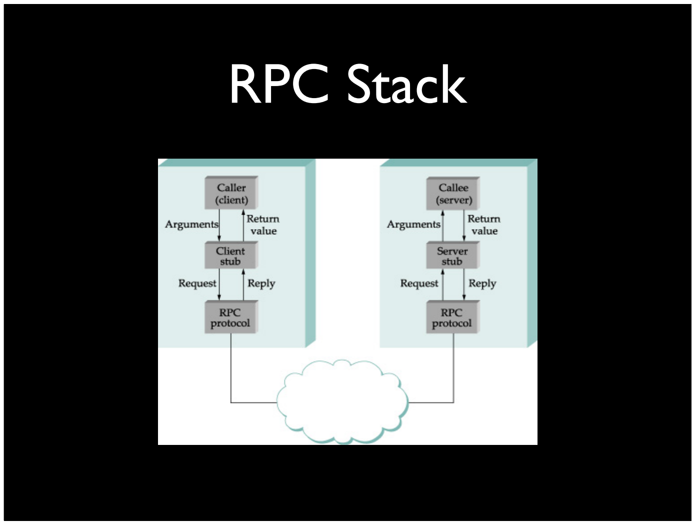
# Request Reply



36

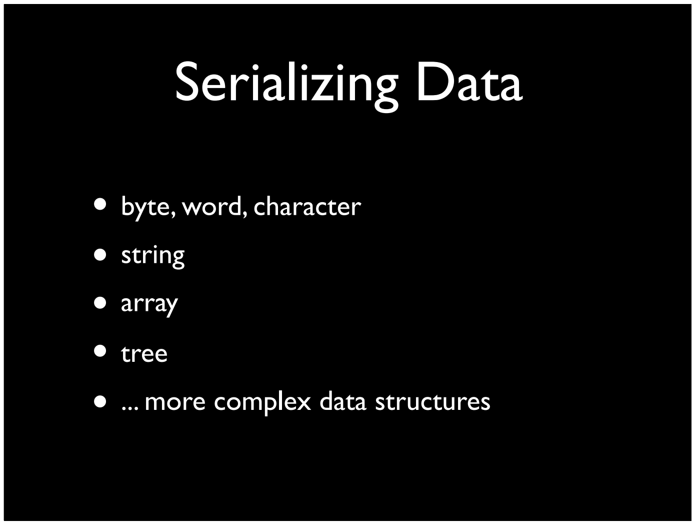
# RPC Stack

- Client-Side
  - Generates "stubs"
  - Developer calls stubs
- Server-Side
  - Generates "server"
  - Server calls developers routines



# RPC Stack

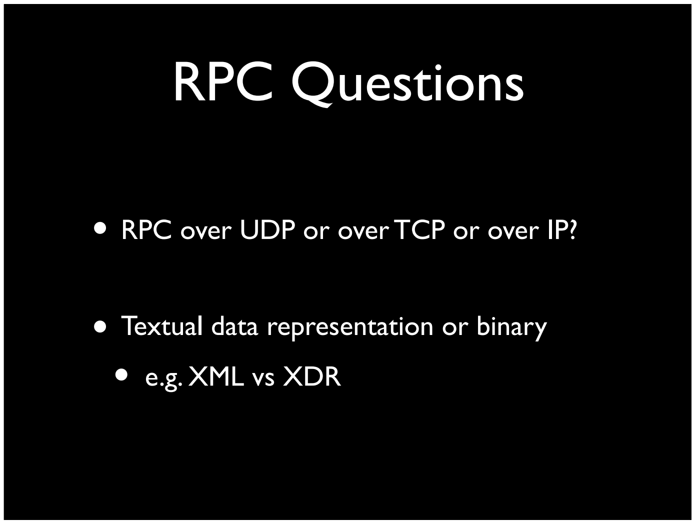
37



# Serializing Data

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

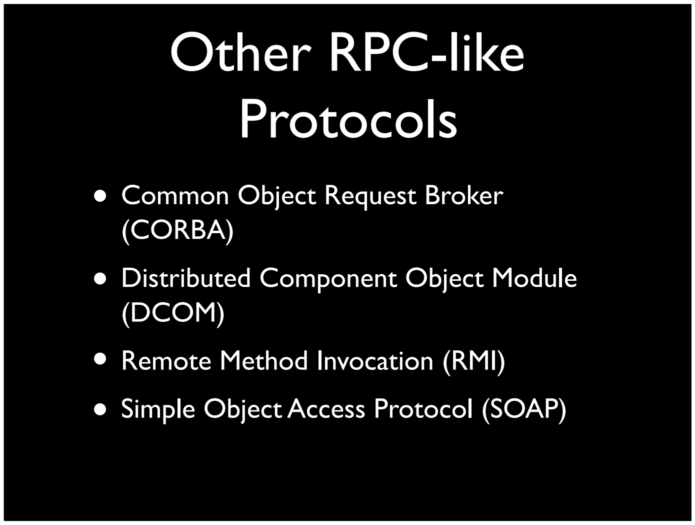
38



# RPC Questions

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

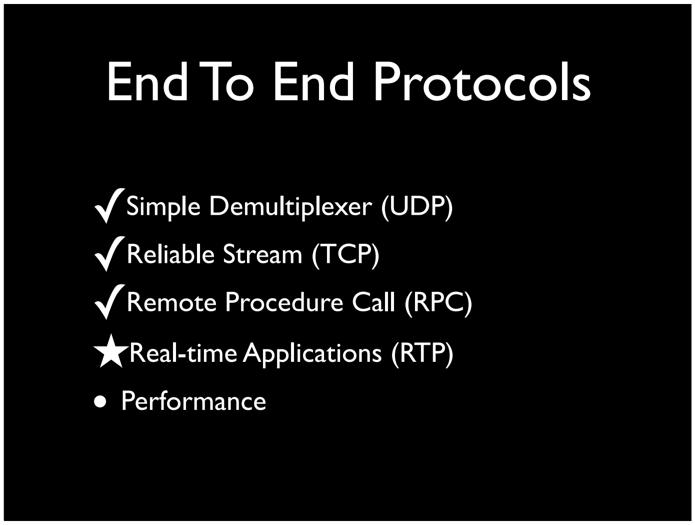
39



40

# Other RPC-like Protocols

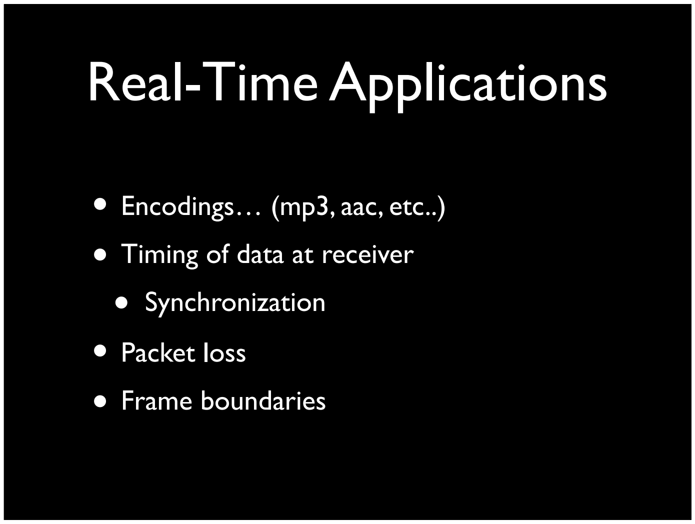
- Common Object Request Broker (CORBA)
- Distributed Component Object Module (DCOM)
- Remote Method Invocation (RMI)
- Simple Object Access Protocol (SOAP)



41

# End To End Protocols

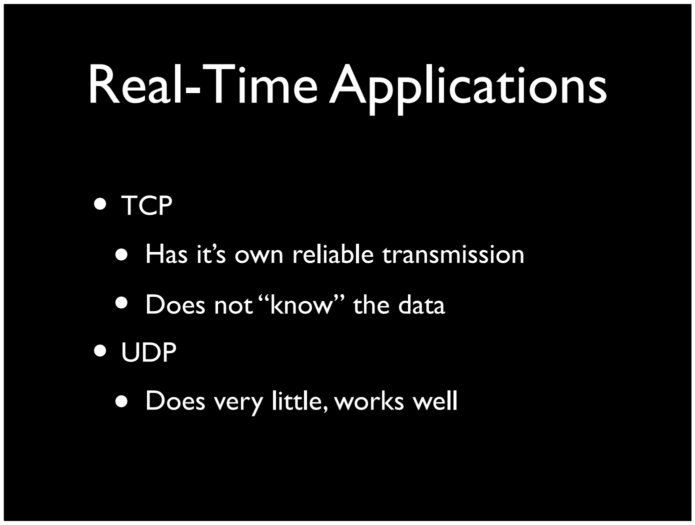
✓ Simple Demultiplexer (UDP)
✓ Reliable Stream (TCP)
✓ Remote Procedure Call (RPC)
★ Real-time Applications (RTP)
- Performance



42

# Real-Time Applications

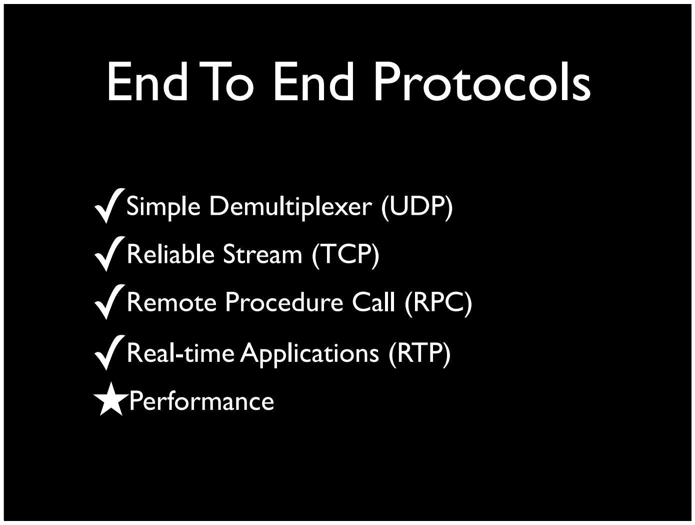
- Encodings… (mp3, aac, etc..)
- Timing of data at receiver
  - Synchronization
- Packet loss
- Frame boundaries



43

## Real-Time Applications

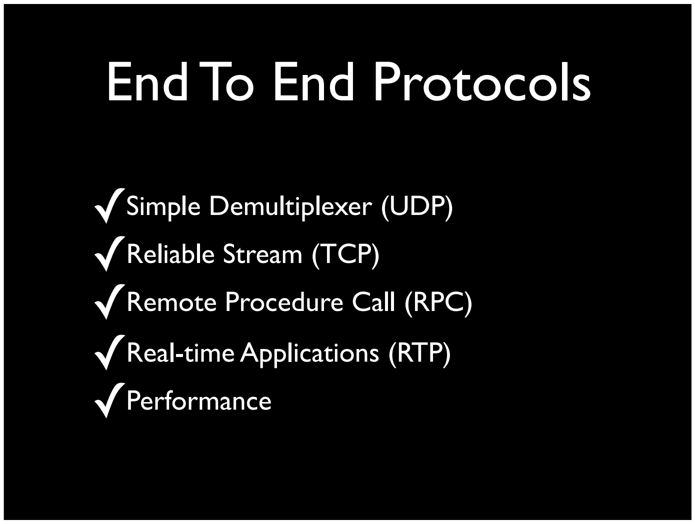
- TCP
  - Has it’s own reliable transmission
  - Does not “know” the data
- UDP
  - Does very little, works well



44

## End To End Protocols

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



45

## End To End Protocols

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

# End

End To End Protocols