CS244a: An Introduction to Computer Networks

Handout 6: The Transport Layer, Transmission Control Protocol (TCP), and User Datagram Protocol (UDP)

Nick McKeown

Professor of Electrical Engineering and Computer Science, Stanford University

nickm@stanford.edu
http://www.stanford.edu/~nickm

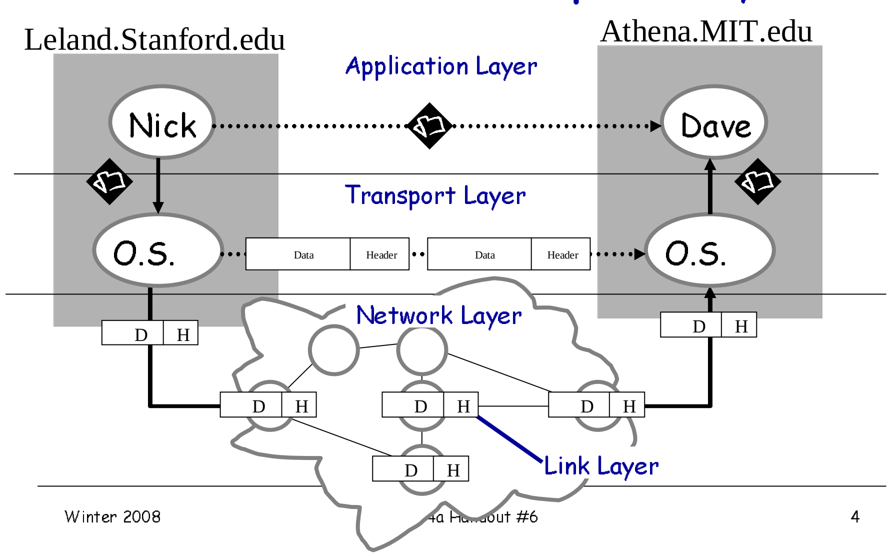
Outline

- The Transport Layer
- The TCP Protocol
  - TCP Characteristics
  - TCP Connection setup
  - TCP Segments
  - TCP Sequence Numbers
  - TCP Sliding Window
  - Timeouts and Retransmission
  - (Congestion Control and Avoidance)
- The UDP Protocol

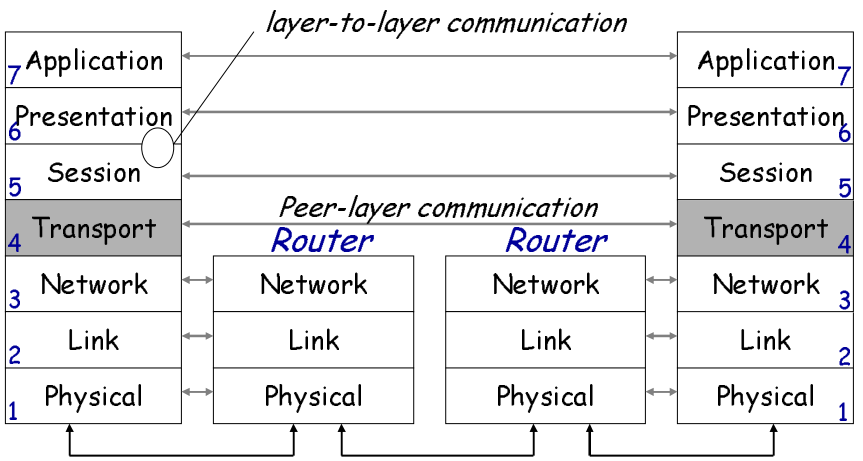
The Transport Layer

- What is the transport layer for?
- What characteristics might it have?
  - Reliable delivery
  - Flow control
  - ...

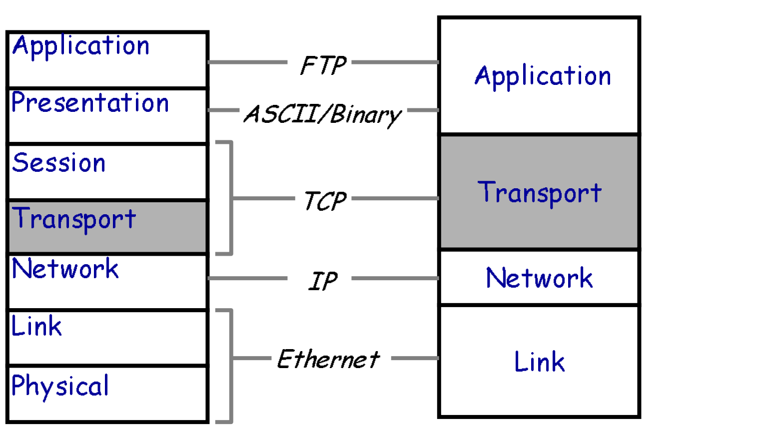
Review of the transport layer



Layering: The OSI Model



Layering: Our FTP Example



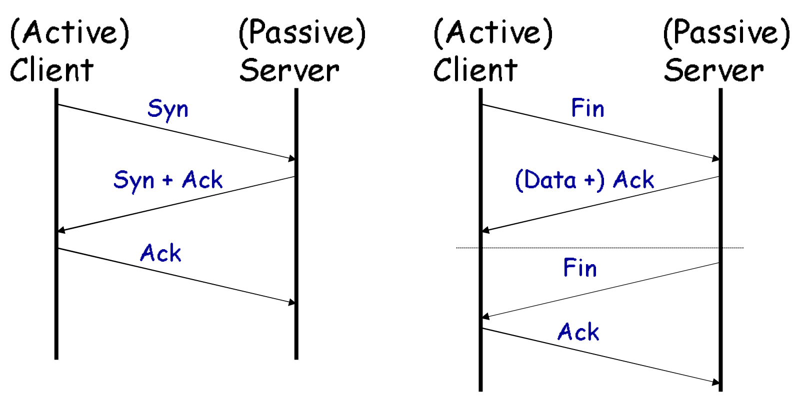
The 7-layer OSI Model

The 4-layer Internet model

TCP Characteristics

- TCP is *connection-oriented.*
  - 3-way handshake used for connection setup.
- TCP provides a *stream-of-bytes* service.
- TCP is reliable:
  - Acknowledgements indicate delivery of data.
  - Checksums are used to detect corrupted data.
  - Sequence numbers detect missing, or mis-sequenced data.
  - Corrupted data is retransmitted after a timeout.
  - Mis-sequenced data is re-sequenced.
  - (Window-based) Flow control prevents over-run of receiver.
- TCP uses *congestion control* to share network capacity among users. We'll study this in the next lecture.

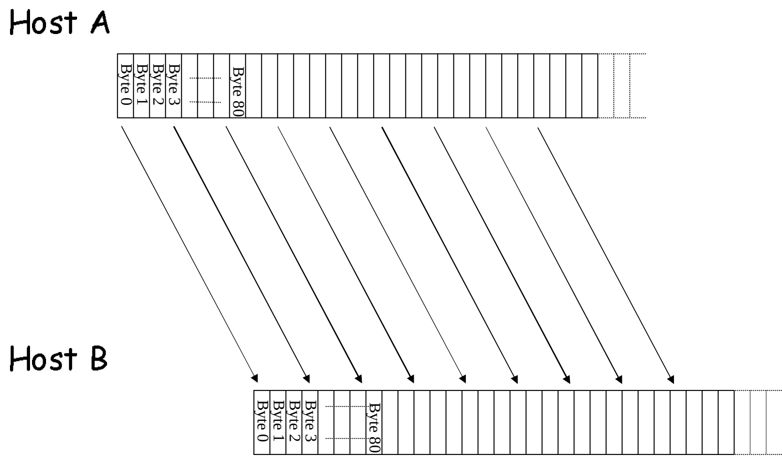
TCP is connection-oriented



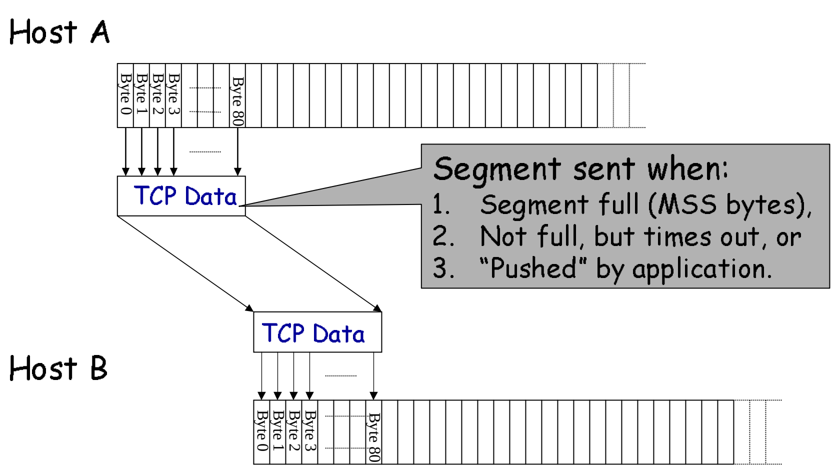
Connection Setup
3-way handshake

Connection Close/Teardown
2 x 2-way handshake

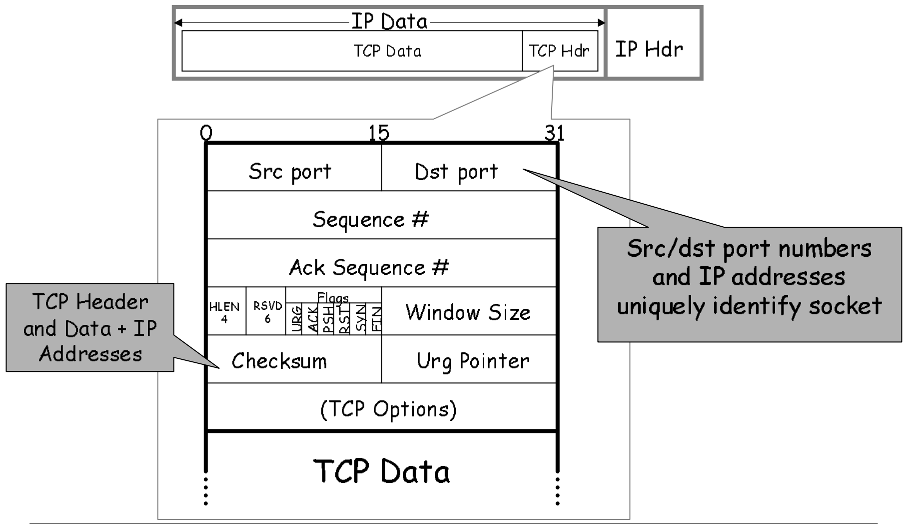
TCP supports a "stream of bytes" service



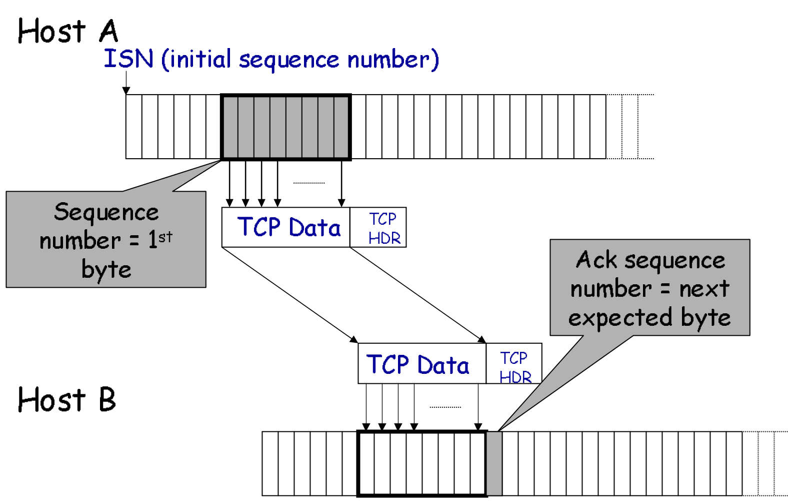
…which is emulated using TCP "segments"



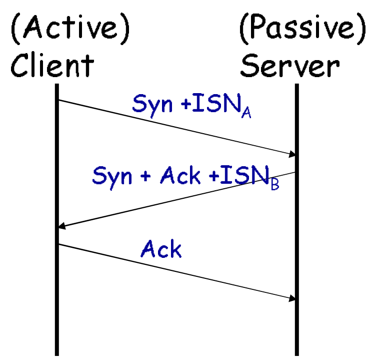
The TCP Segment Format



Sequence Numbers



Initial Sequence Numbers

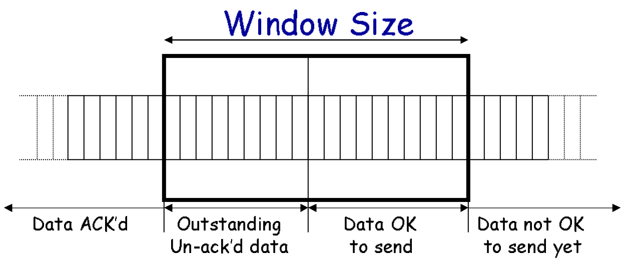


Connection Setup
3-way handshake

TCP Sliding Window

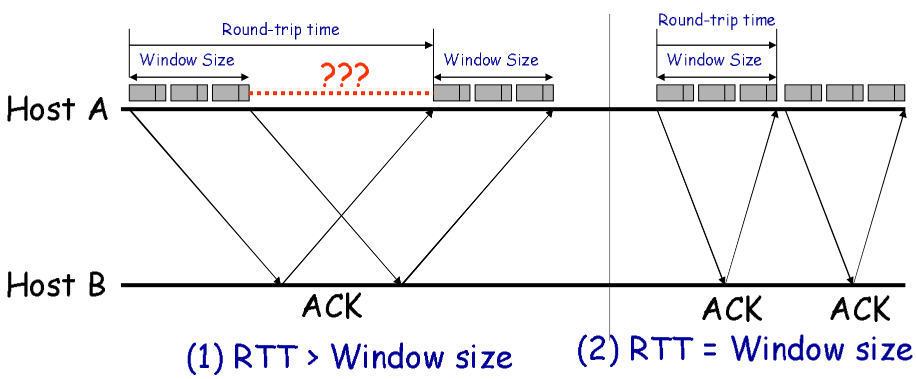
- How much data can a TCP sender have outstanding in the network?
- How much data should TCP retransmit when an error occurs? Just selectively repeat the missing data?
- How does the TCP sender avoid over-running the receiver's buffers?

TCP Sliding Window

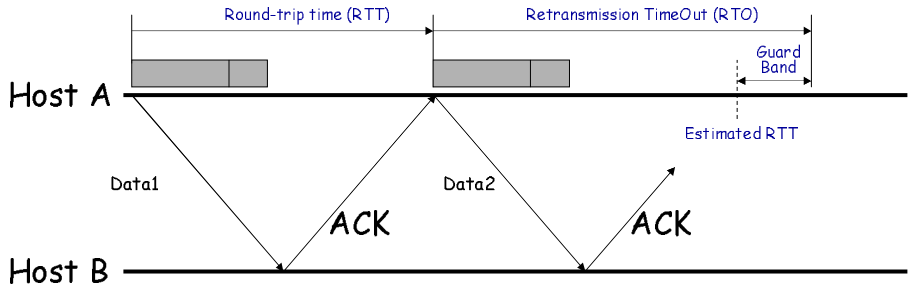


- Window is meaningful to the *sender*.
- Current window size is "advertised" by receiver (usually 4k – 8k Bytes when connection set-up).
- TCP's Retransmission policy is "Go Back N".

TCP Sliding Window



TCP: Retransmission and Timeouts



TCP uses an adaptive retransmission timeout value:

Congestion, Changes in Routing — RTT changes frequently

TCP: Retransmission and Timeouts

Picking the RTO is important:

- Pick a values that's too big and it will wait too long to retransmit a packet,
- Pick a value too small, and it will unnecessarily retransmit packets.

The original algorithm for picking RTO:

1. $\text{EstimatedRTT}_k = \alpha \, \text{EstimatedRTT}_{k-1} + (1 - \alpha)\, \text{SampleRTT}$
2. RTO = 2 * EstimatedRTT

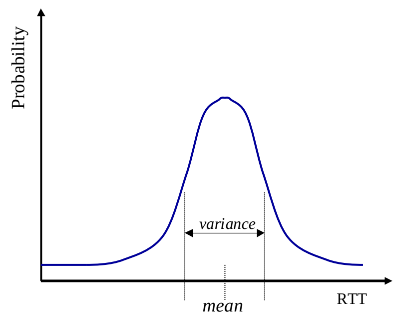
Determined empirically

Characteristics of the original algorithm:

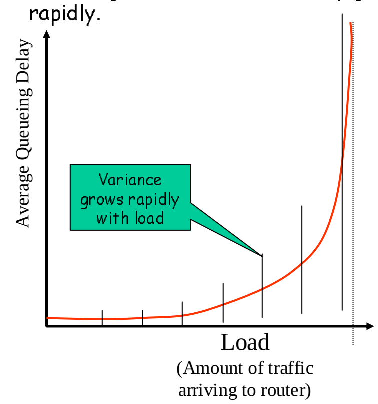
- Variance is assumed to be fixed.
- But in practice, variance increases as congestion increases.

TCP: Retransmission and Timeouts

- There will be some (unknown) distribution of RTTs.
- We are trying to estimate an RTO to minimize the probability of a false timeout.



- Router queues grow when there is more traffic, until they become unstable.
- As load grows, variance of delay grows rapidly.



TCP: Retransmission and Timeouts

<u>Newer Algorithm includes estimate of variance in RTT:</u>

Same as before

- Difference = SampleRTT - EstimatedRTT
- $\text{EstimatedRTT}_k = \text{EstimatedRTT}_{k-1} + (\delta * \text{Difference})$
- Deviation = Deviation + δ*(|Difference| - Deviation)

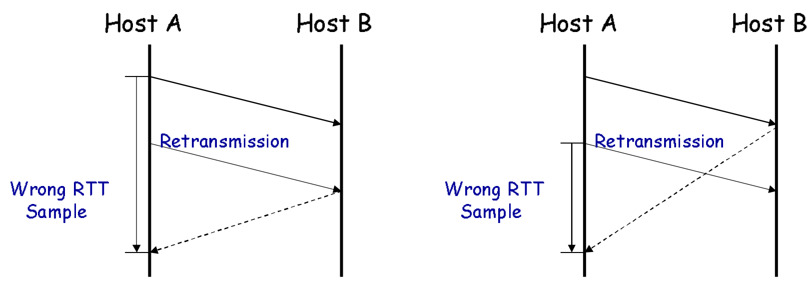
- RTO = μ * EstimatedRTT + ϕ * Deviation

  $\mu \approx 1$

  $\phi \approx 4$

TCP: Retransmission and Timeouts

Karn's Algorithm



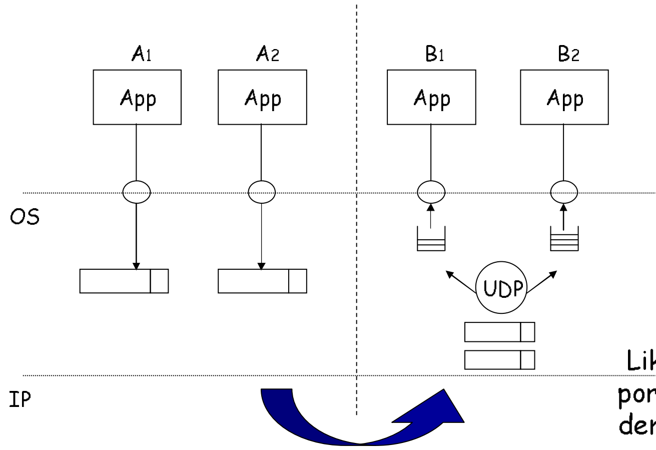
Problem:
How can we estimate RTT when packets are retransmitted?
Solution:
On retransmission, don't update estimated RTT (and double RTO).

User Datagram Protocol (UDP)

Characteristics

- UDP is a *connectionless datagram service.*
  - There is no connection establishment: packets may show up at any time.
- UDP packets are self-contained.
- UDP is unreliable:
  - No acknowledgements to indicate delivery of data.
  - Checksums cover the header, and only optionally cover the data.
  - Contains no mechanism to detect missing or mis-sequenced packets.
  - No mechanism for automatic retransmission.
  - No mechanism for flow control, and so can over-run the receiver.

User-Datagram Protocol (UDP)



Like TCP, UDP uses port number to demultiplex packets

User-Datagram Protocol (UDP)

Packet format

By default, only covers the header.

SRC port	DST port
checksum	length
DATA	

- Why do we have UDP?
  - It is used by applications that don't need reliable delivery, or
  - Applications that have their own special needs, such as streaming of real-time audio/video.