CIS 457 - Data Communications

Nathan Bowman

Images taken from Kurose and Ross book

ACKs in TCP

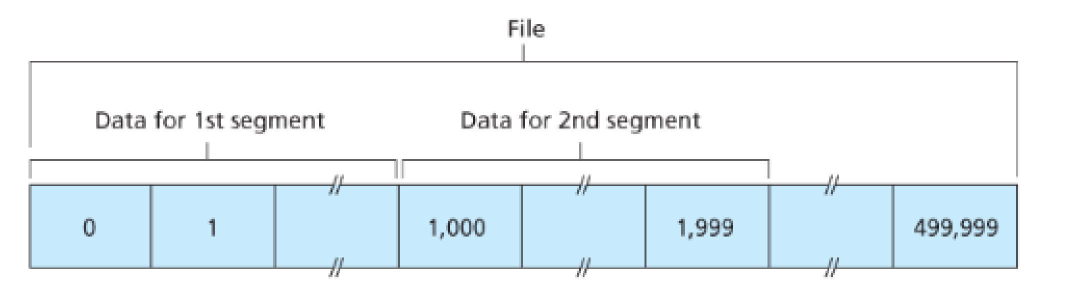
TCP views data as stream of bytes that is

- unstructured
- ordered

So, TCP does not use sequence numbers to refer to particular segment

Instead, sequence numbers refer to bytes in the stream

Sequence number field in TCP header gives byte number of first byte in segment


File
Data for 1st segment
Data for 2nd segment
0
1
1,000
1,999
499,999

Recall that TCP connection is bi-directional

One stream of data flows from Host A -> Host B,
another stream flows from Host B -> Host A

Each stream numbers bytes individually

Sequence numbers on A -> B packets completely
independent of sequence numbers on B -> A packets

Assume A sends to B and B ACKs

ACK field in TCP header from B shows *next* byte B expects from A

This is cumulative acknowledgement -- every byte before number specified in ACK has been received

In GBN, our previous example with cumulative acknowledgements, receiver would not buffer out-of-order packets

TCP specification does not mandate one way or the other

Most TCP implementations have receiver buffer out-of-order packets in order to save network bandwidth

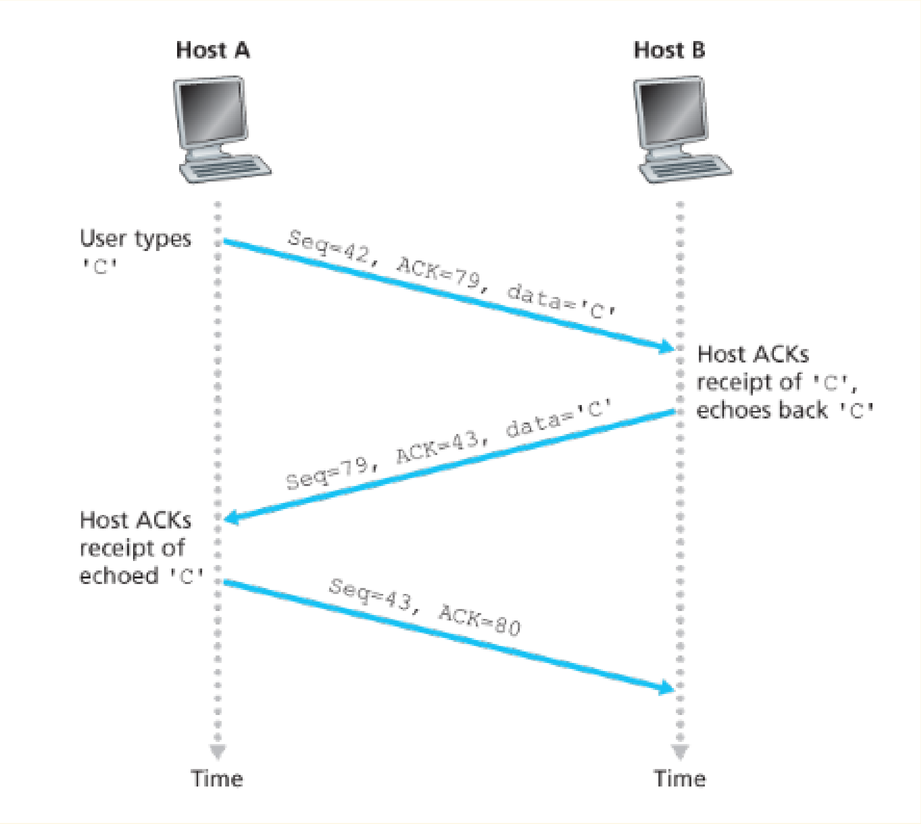
TCP endpoints try to be efficient with number of packets sent

If B has information to send A and also wants to ACK A recent packet from A, it can do so in single segment

B sets ACK bit in header, specifies correct ACK number, then sends message as usual

Use as example Telnet protocol

In example, Telnet server simply echoes back what was sent to it by client


Host A
Host B
User types 'C'
Seq=42, ACK=79, data='C'
Host ACKs receipt of 'C', echoes back 'C'
Seq=79, ACK=43, data='C'
Host ACKs receipt of echoed 'C'
Seq=43, ACK=80
Time
Time

To deal with packet loss, TCP will resend packet after certain amount of time without ACK

Determining good amount of time to wait is easier with good estimate of round-trip time (RTT)

Estimate RTT by sampling

Because RTT fluctuates over time, continuously sampling RTT for various packets

Generally sample for just one packet at a time

Results in one new sample per RTT

Samples of RTT time are noisy, so TCP averages them

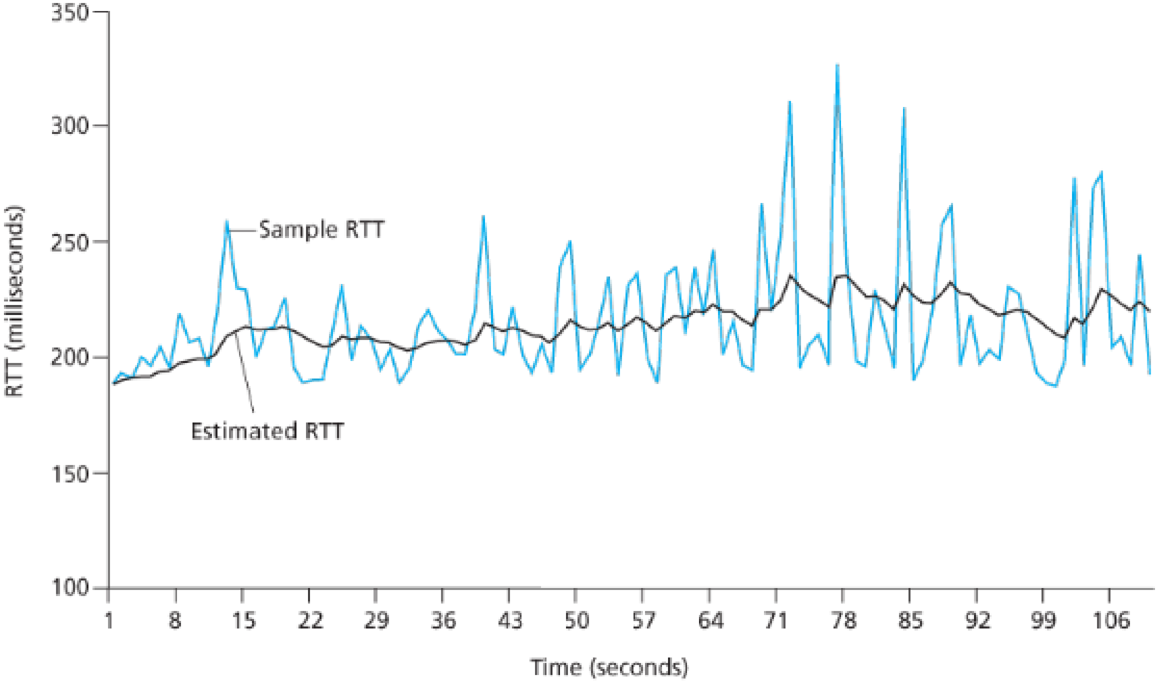
```
ExpectedRTT = (1 - a)*ExpectedRTT + a*SampleRTT
```

Value of a changes behavior of average -- generally chosen to be 1/8

Formula requires some initial ExpectedRTT to get started -- one minute is common choice

Computed according to given formula, ExpectedRTT is exponentially-weighted moving average of SampleRTT

Most recent samples count for most, and weight of older samples decreases exponentially over time



RTT (milliseconds)
350
300
250
200
150
100
Sample RTT
Estimated RTT
1
8
15
22
29
36
43
50
57
64
71
78
85
92
99
106
Time (seconds)

When determining whether packet has been lost based on time, also helpful to know how much RTT usually deviates from average

Deviation from expected time also computed as exponentially-weighted moving average

```
DevRTT = (1 - b)*DevRTT + b*|SampleRTT - ExpectedRTT|
```

Usually choose b = 1/4

Timeout value is then

```
timeout = ExpectedRTT + 4*DevRTT
```