Chapter 5
TCP Sliding Window

Networking

CS 3470, Section 1

Sliding Window

- Remember this? What was it useful for?

Sliding Window Revisited

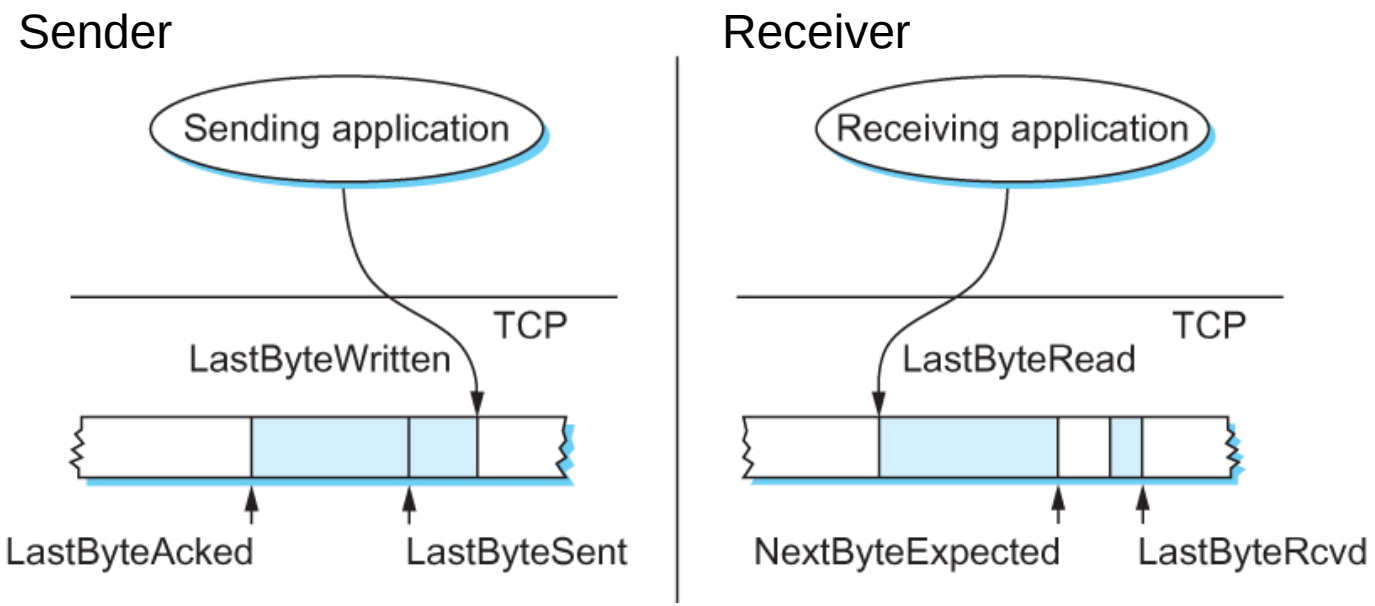
- TCP's variant of the sliding window algorithm, which serves several purposes:
  - (1) it guarantees the reliable delivery of data,
  - (2) it ensures that data is delivered in order, and
  - (3) it enforces flow control between the sender and the receiver.

(Same as Chapter 2 for (1) and (2), but adds flow control.)

Sliding Window Revisited

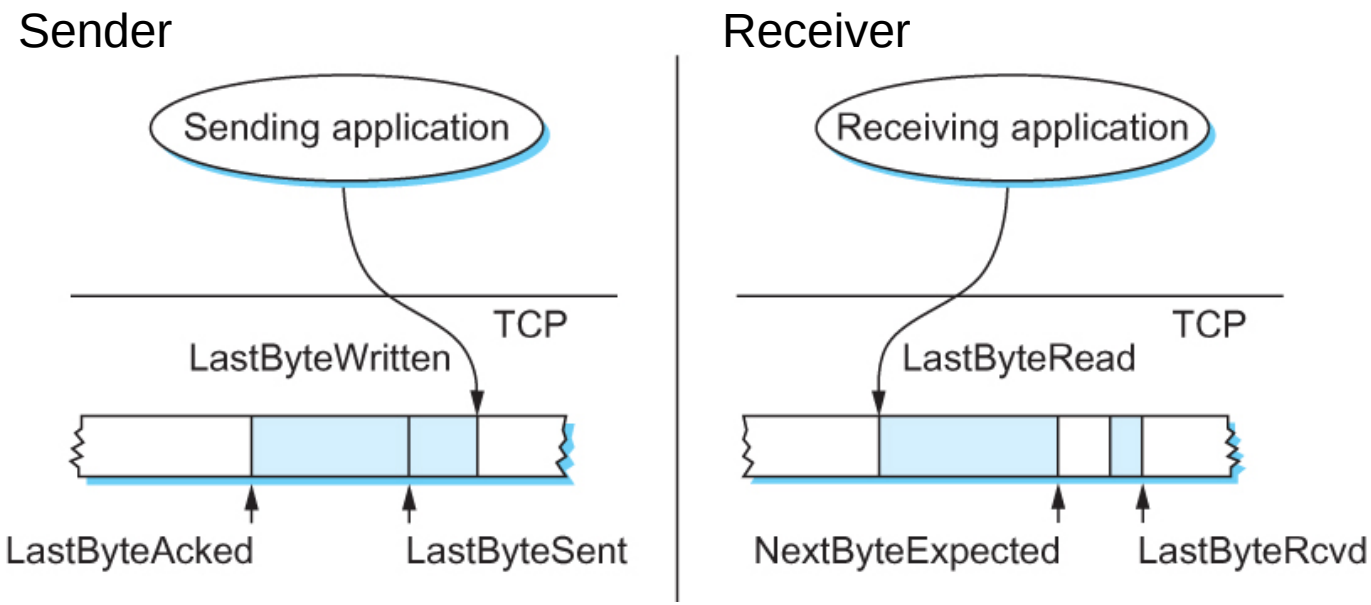
- Rather than having fixed-size sliding window, receiver ***advertises*** a window size to the sender
  - ***AdvertisedWindow*** field in TCP header
- Sender is limited to having no more than AdvertisedWindow bytes of unACK'ed data at any given time
  - Receiver selects this value based on amount of memory allocated to connection

Sliding Window Revisited



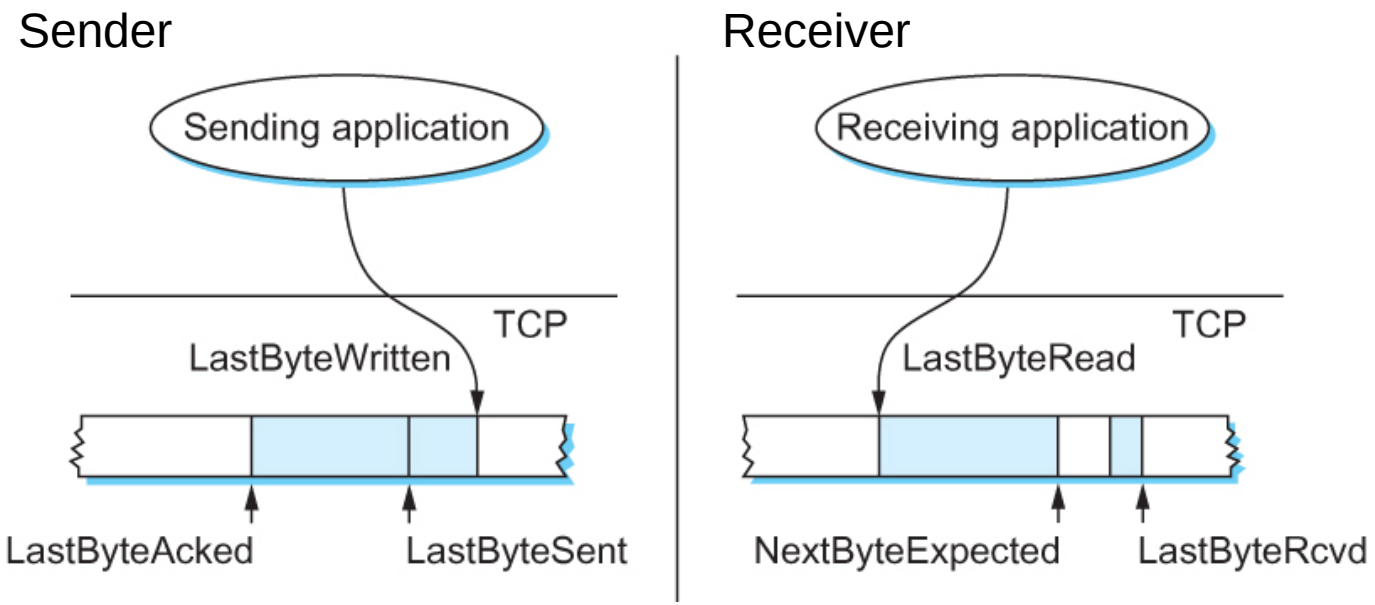
- Sender side
  - LastByteAcked <= LastByteSent
  - LastByteSent <= LastByteWritten

Sliding Window Revisited



- Receiver side
  - Less intuitive because of the problem of out-of-order delivery

Sliding Window Revisited



- Receiver side
  - LastByteRead < NextByteExpected
  - NextByteExpected <= LastByteRcvd +1

TCP Flow Control - Receiver

- Receiver throttles the sender by advertising a window that is no larger than the amount of data it can buffer
- Receiver advertises following window size:
  - AdvertisedWindow = MaxRcvBuffer – ((NextByteExpected -1) – LastByteRead)
  - Represents amount of free space remaining in its buffer

TCP Flow Control - Receiver

- Size of AdvertisedWindow depends on how fast local application process is consuming data
  - Window ***shrinks*** when NextByteExpected moves ahead faster than LastByteRead
  - Window ***stays open*** when NextByteExpected moves at same rate of LastByteRead

TCP Flow Control - Sender

- TCP on sender must adhere to the advertised window it gets from the receiver.
- Must ensure that
  - LastByteSent – LastByteAcked <= AdvertisedWindow

TCP Flow Control - Sender

- EffectiveWindow is amount of data that it can still send
  - EffectiveWindow = AdvertisedWindow – (LastByteSent – LastByteAcked)
- Must be > 0 to send any data

TCP Flow Control - Sender

- TCP will ***block*** sending process if it tries to send more data than what fits in the effective window

- What happens when AdvertisedWindow is 0, and sender is not allowed to send any more segments?

Sequence Number

- ***SequenceNum*** field contains the sequence number for the first byte of data carried in segment
  - Important for ??

Sequence Number

- ***SequenceNum*** field contains the sequence number for the first byte of data carried in segment
  - Important for
    - Detecting dropped packets
    - Detecting out of order packets
    - Flow control

TCP seq. #'s and ACKs

Seq. #'s:

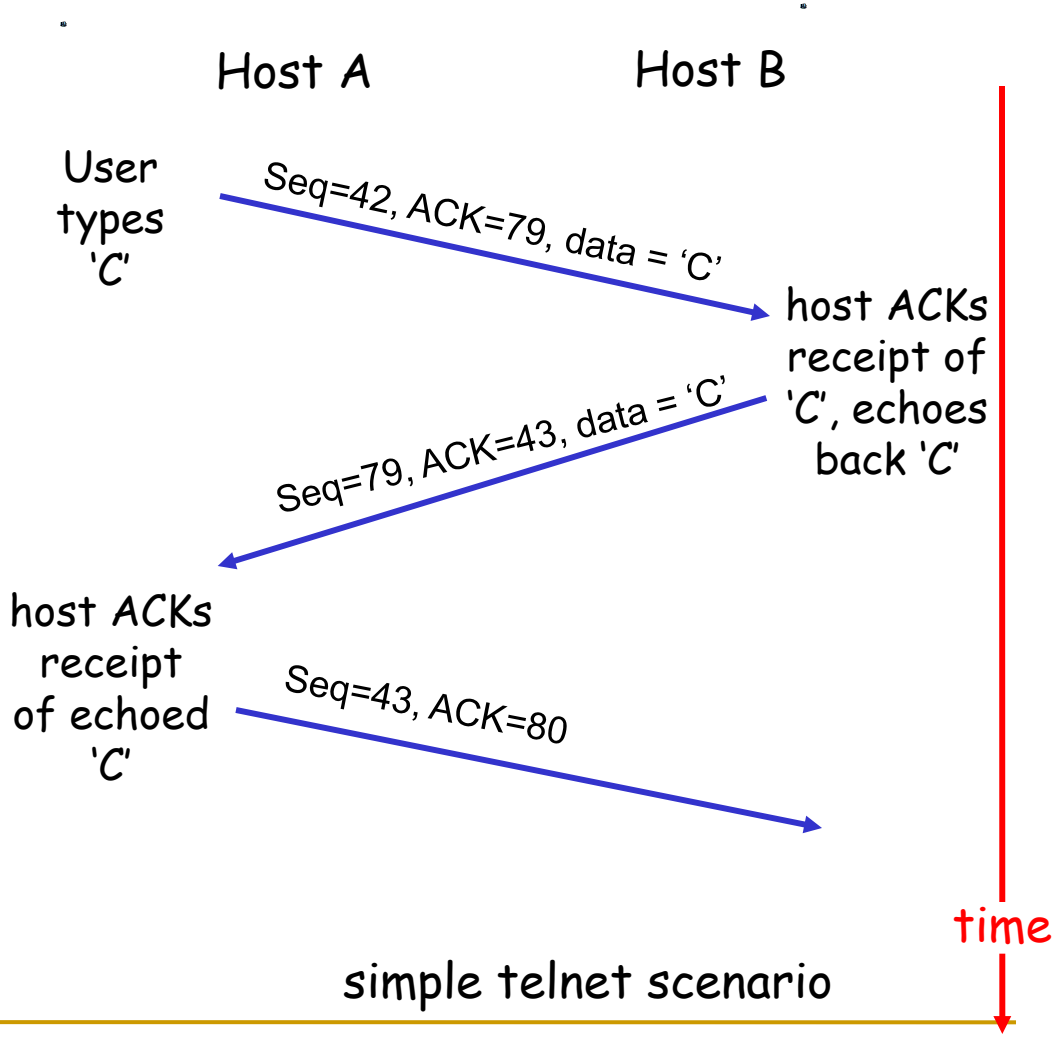
- byte stream "number" of first byte in segment's data

ACKs:

- seq # of next byte expected from other side
- cumulative ACK

Q: how receiver handles out-of-order segments

- A: TCP spec doesn't say, - up to implementer



simple telnet scenario

Protecting against Wraparound

- Relevance of the 32-bit sequence number space
  - The sequence number used on a given connection might wraparound
  - A byte with sequence number x could be sent at one time, and then at a later time another byte with the same sequence number x could be sent

Protecting against Wraparound

- Packets cannot survive in the Internet for longer than the TCP ***Maximum Segment Lifetime*** (MSL), which is 120 sec
- We need to make sure that the sequence number does not wrap around within a 120-second period of time
- Depends on how fast data can be transmitted over the Internet

Protecting against Wraparound

- How many bytes of transferred data does the 32-bit sequence number represent?

Protecting against Wraparound

- How many bytes of transferred data does the 32-bit sequence number represent?
  - 2^{32} bytes are represented
  - 4 GB of data can be sent!

Protecting against Wraparound

Bandwidth	Time until Wraparound
T1 (1.5 Mbps)	6.4 hours
Ethernet (10 Mbps)	57 minutes
T3 (45 Mbps)	13 minutes
Fast Ethernet (100 Mbps)	6 minutes
OC-3 (155 Mbps)	4 minutes
OC-12 (622 Mbps)	55 seconds
OC-48 (2.5 Gbps)	14 seconds

Time until 32-bit sequence number space wraps around.

- TCP extension is used to extend sequence number space

Figuring out Wraparound Time

- 2^{32} B / bandwidth (in Bytes)
- How long for wraparound on 2.5Gbps network (OC-48)?
  - Convert bandwidth to Bytes
    - 2.5 Gbps / 8 = 0.3125 GBps * 10^9 = 312,500,000 Bps
    - 2^{32} B / 312,500,000 Bps = 14 sec
    - (Can also use 2^{30} instead of 10^9)

Keeping the Pipe Full

- 16-bit AdvertisedWindow field must be big enough to allow the sender to keep the pipe full
- If the receiver has enough buffer space
  - The window needs to be opened far enough to allow a full delay × bandwidth product's worth of data
  - Assuming an RTT of 100 ms (typical cross-country connection in US)

Keeping the Pipe Full

Bandwidth	Delay × Bandwidth Product
T1 (1.5 Mbps)	18 KB
Ethernet (10 Mbps)	122 KB
T3 (45 Mbps)	549 KB
Fast Ethernet (100 Mbps)	1.2 MB
OC-3 (155 Mbps)	1.8 MB
OC-12 (622 Mbps)	7.4 MB
OC-48 (2.5 Gbps)	29.6 MB

Required window size for 100-ms RTT.

- Uh oh – 16 bit field only allows us to advertise a window of 64KB (2^{16} = 65536 B = 64KB)
- TCP extension is used to extend AdvertisedWindow