Computer Network I

Reti di Calcolatori I

Università di Napoli Federico II – Scuola Politecnica e delle Scienze di Base
Corso di Laurea in Informatica

Riccardo Caccavale

(riccardo.caccavale@unina.it)

UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

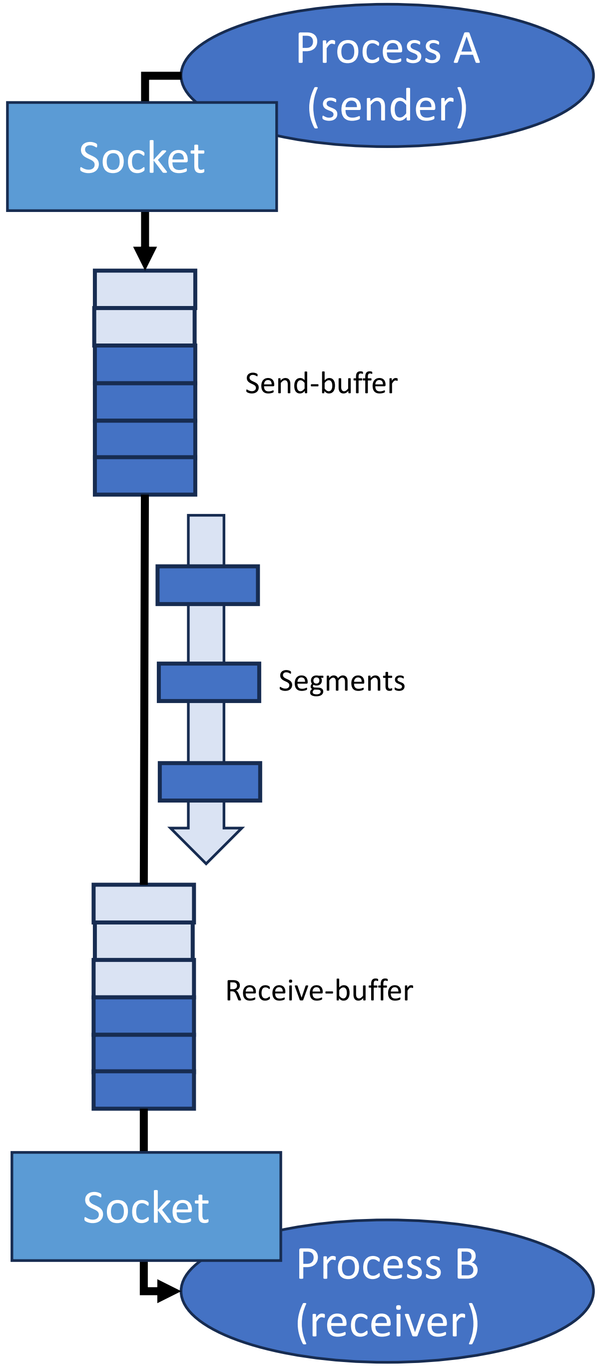
Transport Layer
TCP

- TCP with respect to UDP is:
  - **Connection-oriented**: there is a “handshake” phase before the transmission that ensures the two processes (sender and receiver) to be “connected”.
  - **Reliable**: error detection, retransmissions, acknowledgments, timers, sequence numbers are implemented.
  - **Congestion and flow aware**: there is a regulation of the transmission rate depending on the receiver performance (flow) and the network performance (congestion).

- Connection orientation makes TCP full-duplex and point-to-point:
  - **Full-duplex**: if host A is connected with B then B is connected with A.
  - **Point-to-point**: single sender and single receiver, i.e., the transfer of data from one sender to many receivers is not possible.
    - Note: UDP allows multicasting, there are ways to transfer data to multiple hosts in a single send operation is not possible.

Transport Layer
TCP Buffers

- In TCP connection sending and receiving operations strongly **rely on buffers** to be performed.
- Buffers allow us to **partially decouple** transmission times from:
  - **Application delays**.
  - **OS delays** in multiplexing demultiplexing packets.
  - **Network delays** (oscillations of network performance).
- There is also the limit of the transmission rate, so long messages could **be broken into smaller segments** that may be collected in buffers (on the sender-side).



Transport Layer
Maximum Segment Size

- The amount of **data inside a segment** is limited by the **Maximum Segment Size** (MSS):

$$MSS = MTU - header\_size$$

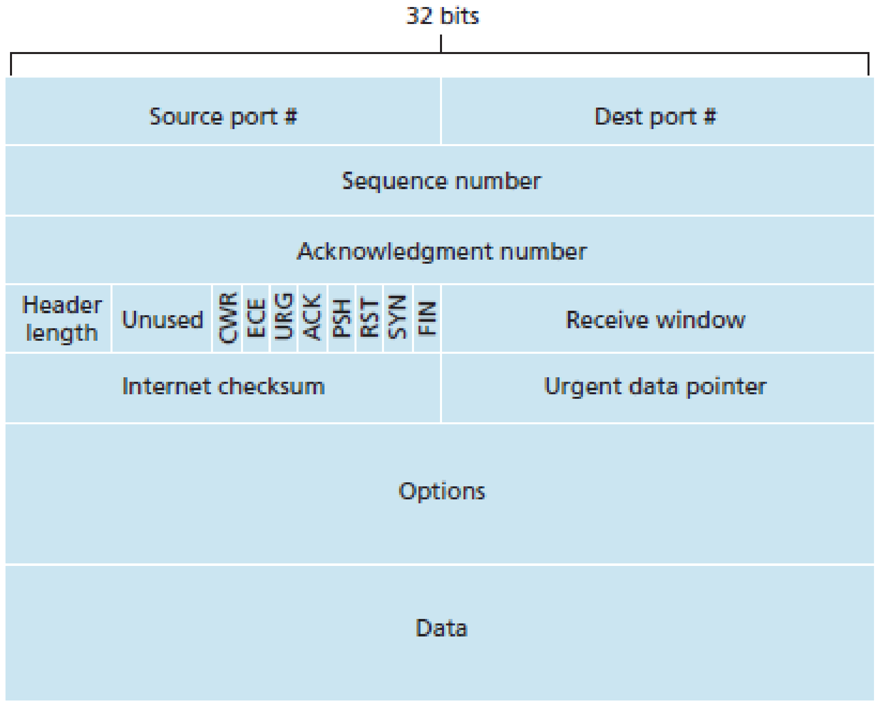
  - Where:
    - The **Maximum Transmission Unit** (MTU) is the maximum length of a frame acceptable by the link-layer (e.g., in Ethernet it is 1500 bytes).
    - The **combined header size** of TCP and IP headers (typically 20+20 bytes).

- **On sending**: the segments are created from the application data and passed down to the network layer, where they are separately encapsulated within network-layer IP datagrams to be sent.
- **On receiving**: the data is placed into the receive-buffer, the application grabs data from the buffer when ready.

Transport Layer

TCP Segment

- The TCP segment consists of **header fields** and a **data field**, the latter contains some application data **of at most MSS size**.
- The **header** contains several fields:
  - **Sequence number** (32-bit) for reliable data transfer.
  - **Acknowledgment number** (32-bit) for reliable data transfer.
  - **Receive window** (16-bit) used to indicate the number of bytes that a receiver is willing to accept (for flow control).
  - **Header length** (4-bit) specifies the number of 32-bit words contained into the header. Note that **TCP header is variable due to the options** field.
  - **Options field** (K-bit) used for optional processes such as negotiating the MSS, time-stamping, etc.
  - **Flags** (6-bit) including:
    - ACK bit indicates that **this segment is an ACK** packet (so the acknowledgment number field is in use).
    - RST, SYN, and FIN bits used for **connection setup and teardown**.
    - CWR and ECE bits used in explicit congestion notification.
    - PSH bit tells the receiver to pass the data to the application immediately.
    - URG bit indicates that the segment contains “urgent” data.
  - **Checksum** (16-bit): for integrity check.
  - **Urgent data pointer field** (16-bit) indicates the location of the last byte of the urgent part of the data.



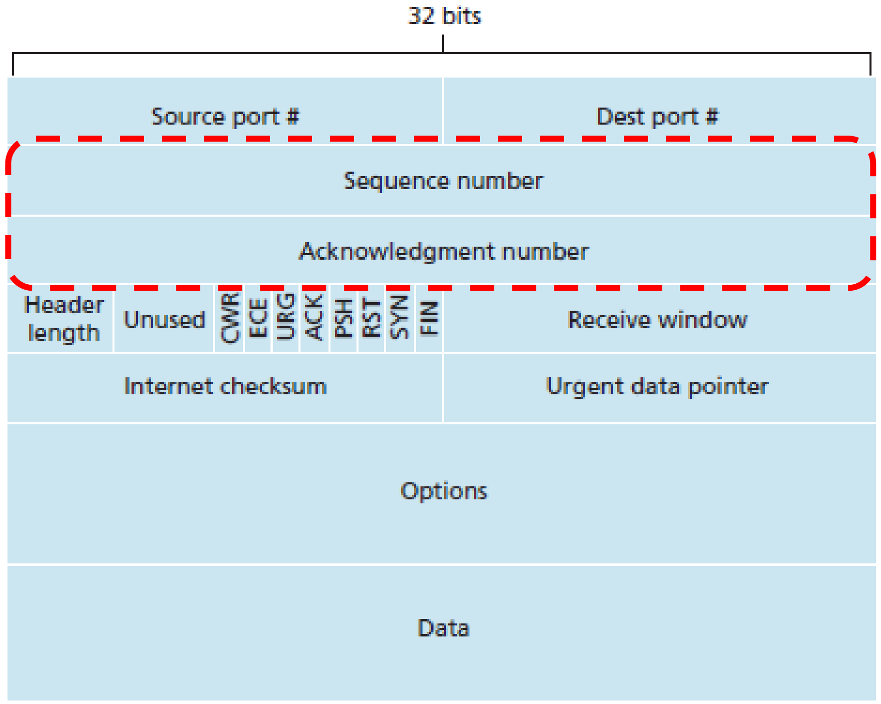
Transport Layer

TCP Sequence and Acknowledgment Numbers

- Sequence numbers and acknowledge numbers are **strictly related**.

- Sequence numbers are used not only for reliable data transfer but **also to manage segmentation**.
    - If a large data stream is transmitted from host A to host B, we need **to break it into smaller pieces** depending on the MSS.

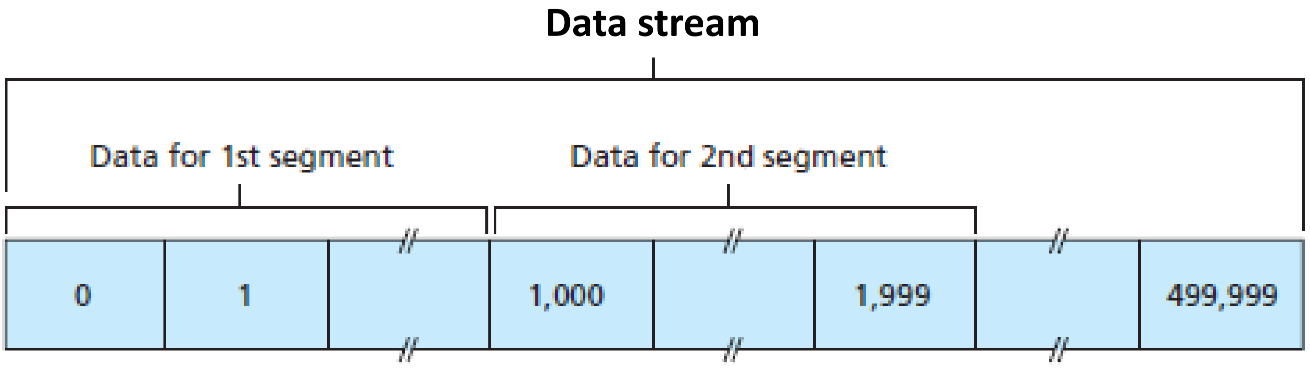
- The acknowledgment number is calculated from sequence numbers. It is conventionally the **next piece of the data stream** we are expecting.



Transport Layer

TCP Sequence and Acknowledgment Numbers

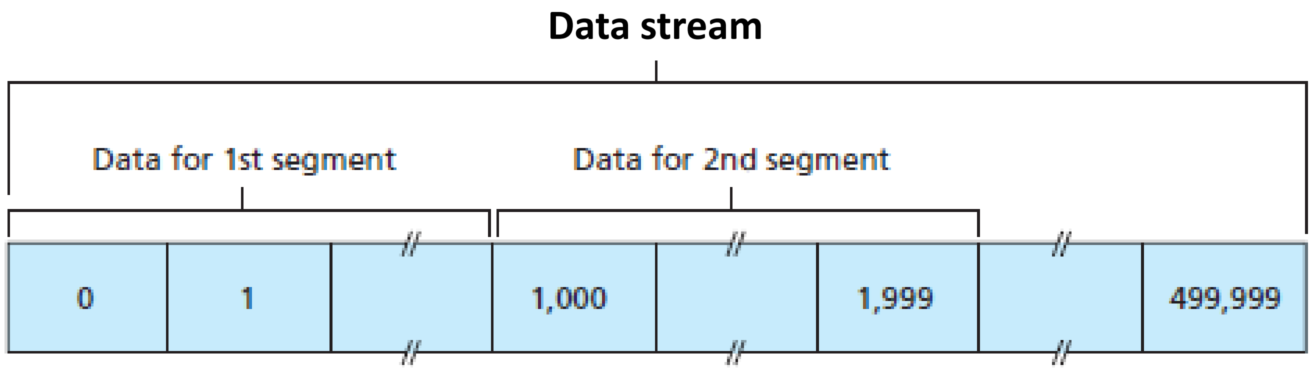
- Notice that TCP is a general-purpose transport protocol, it has no information about the data to be transmitted. From TCP's standpoint data **is just an ordered stream of bytes**.
- The **sequence number** for a segment is then the byte-stream **number of the first byte of the data** in the segment.



- In the example, assuming a **data stream of 500000 bytes** (~500KB), and a **MSS of 1000 bytes**.
- The TCP constructs **500 segments** where the first segment has sequence number 0, the second segment has sequence number 1000, the third segment has sequence number 2000, and so on.

Transport Layer

TCP Sequence and Acknowledgment Numbers



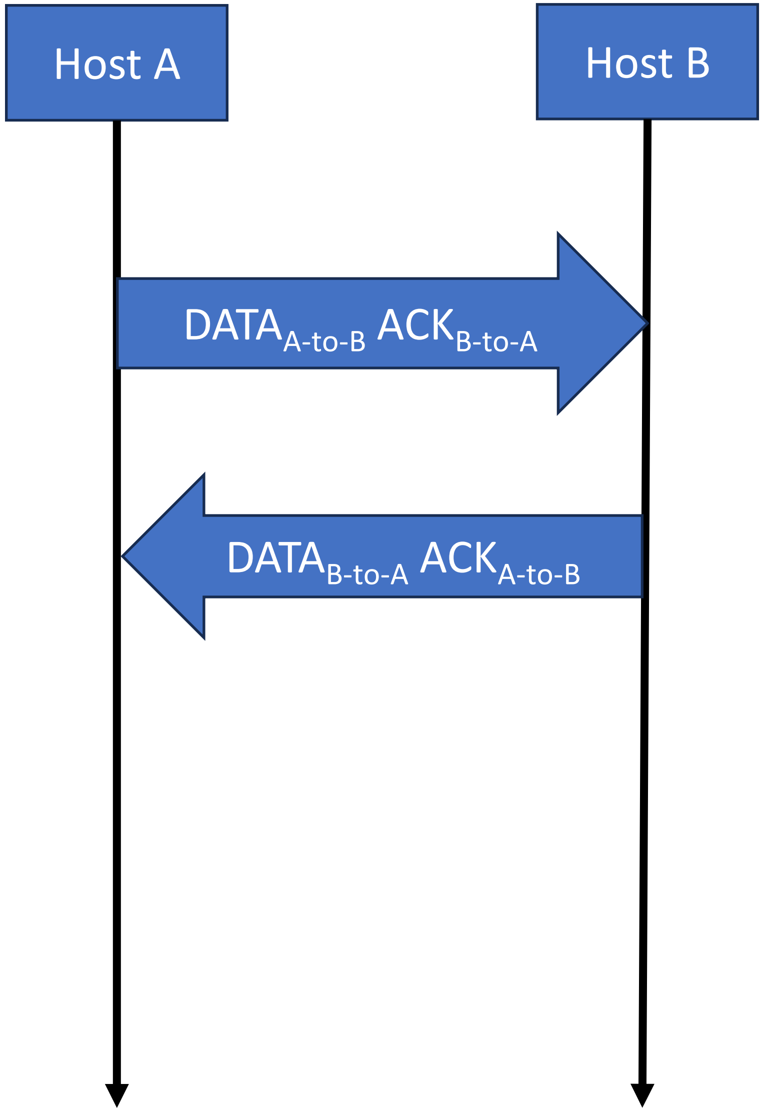
- Let’s assume this data stream to be **passed from host A to host B**.
- For simplicity, let’s **neglect previous interaction** (data exchange from connection establishment), so **we are starting from sequence number 0**:
  - The receiver will get a first segment of 1000 bytes, i.e., from **sequence number 0 to sequence number 999**.
  - The receiver will then acknowledge the transmission by setting an **acknowledgment number of 1000** (next expected byte in the stream).
  - The receiver will get the next segment of 1000 bytes, i.e., from **sequence number 1000 to sequence number 1999** and so on…

Transport Layer

TCP Sequence and Acknowledgment Numbers

- What happens if the **receiver is sending data in return**?
  - Remind: TCP communication is **full-duplex**, if 2 hosts, A and B, communicate there are also 2 flows of data to be considered for reliable data transfer: A-to-B and B-to-A.

- In this case we can **use sequence numbers and acknowledgment numbers at the same time**:
  - Segments from A have **sequence** numbers related to **A-to-B flow** and **acknowledge** numbers related to the **B-to-A flow**.
  - Segments from B have **sequence** numbers related to **B-to-A flow**, and **acknowledge** numbers related to the **B-to-A flow**.

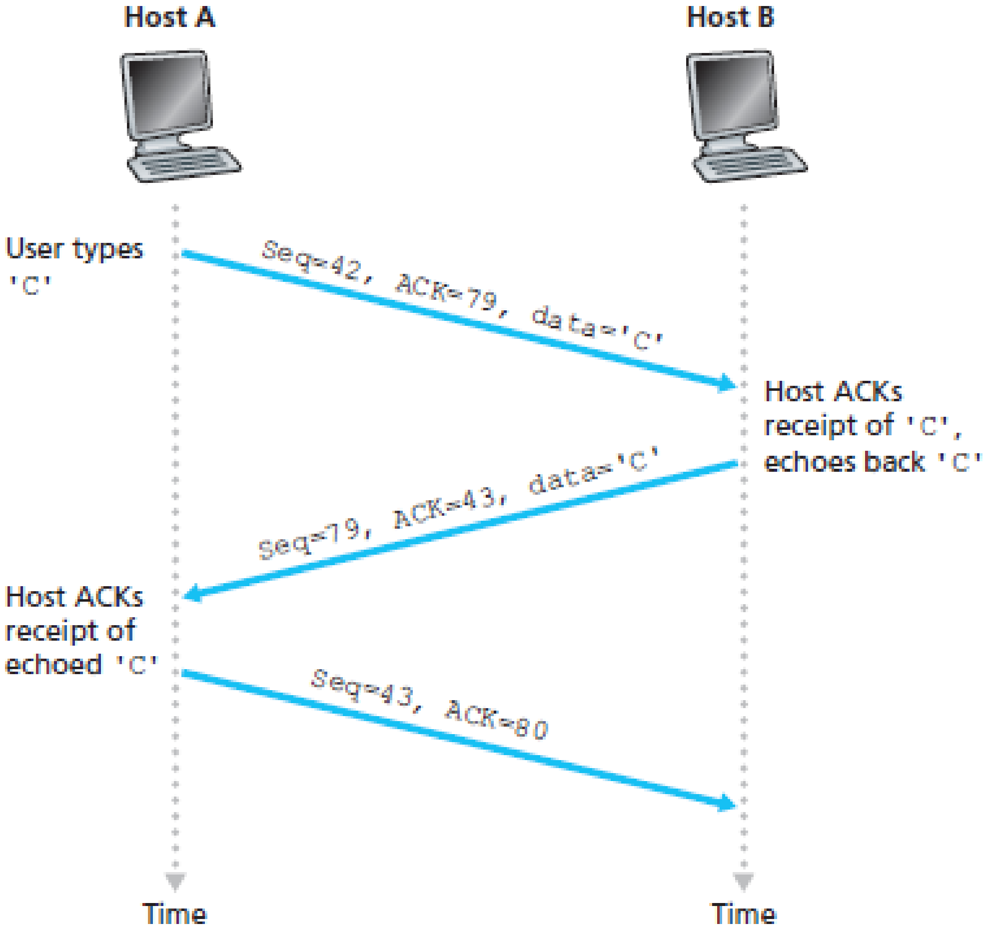


Transport Layer

TCP Sequence and Acknowledgment Numbers

- This is a simplified example of two hosts sending data each other.
  - We are considering 2 messages of **just 1 byte** (1 char) form A to B and vice versa.
  - Specifically, host A transmits a ‘c’ **that is echoed back** by B.

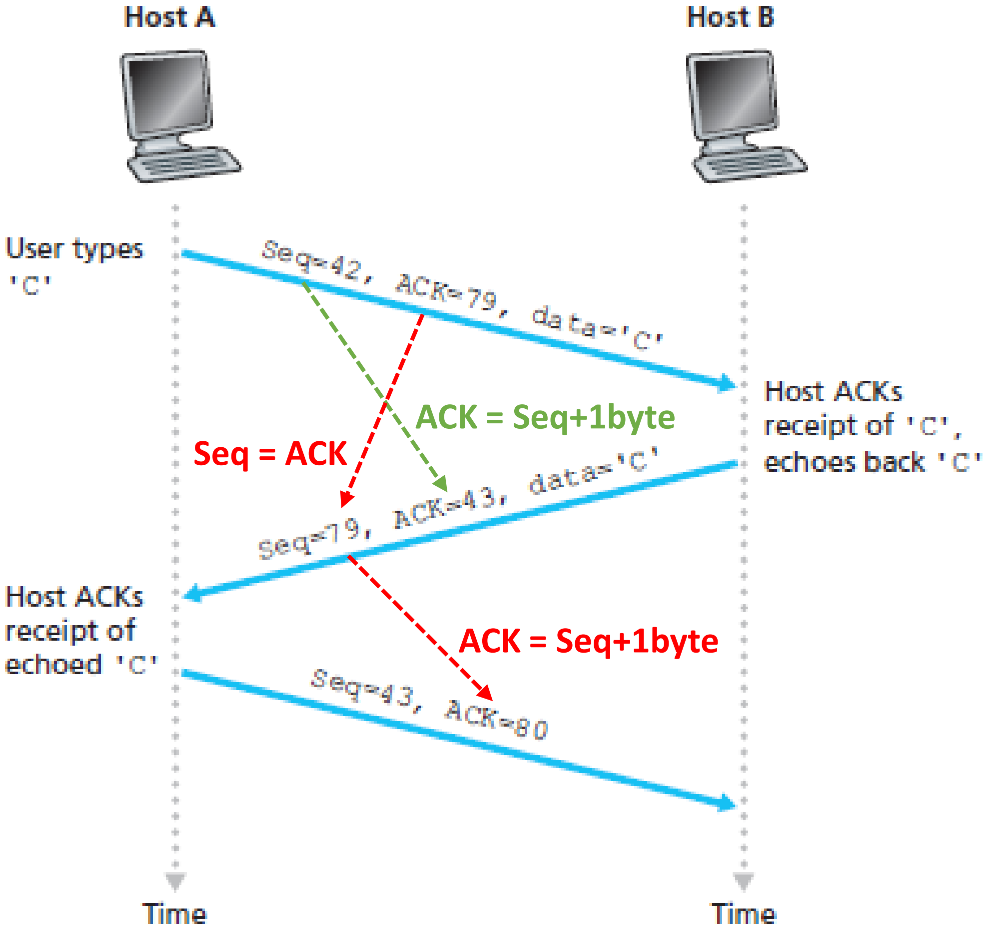
- Here that segments provide **ACK and DATA at the same time** (so the ACK flag is 1).



Transport Layer

TCP Sequence and Acknowledgment Numbers

1. From A to B:
   - The **ack number as the seq number of the next expected packet** in the B-to-A flow (byte 79).
   - The **seq number** as the position of **the single byte** we are transmitting (byte 42).
2. From B to A:
   - The **ack number as the seq number of the next expected packet** in the A-to-B flow (i.e., seq + 1 byte of the 'c' char).
   - The **seq number** as the position of **the byte we are transmitting** (byte 79).
3. From A to B:
   - Pure **ACK segment** is sent, having seq number 43 (as expected by B) and ack number 80 (next expected).



A-to-B info are in green.
B-to-A info are in red.

Transport Layer

Retransmission Timeout

- Previously we have emphasized that **TCP reliable data transfer mechanism makes extensive use of timeouts**.

- Timeouts can be used **in combination with ACKs** to understand (or at least to guess) if segments have been lost and should be retransmitted.
  - Notice that **wrong ACK can be received for different reasons** (not only when lost).
  - For example, if **segments arrives in the wrong order** at the destination a different ACK is sent.

- ACK management can be tricky. The basic approach followed by pipelining methods (sender-side) is to **ignore wrongly received ACKs**:
  - Retransmission is performed **only when timeout is elapsed**.
  - Timeout estimation is **critical**.

Transport Layer

RTT Estimation

- To suitably se timeouts **we need a round-trip time (RTT) estimation**.
- The timeout, i.e., the time to wait for a segment's ACK, should be **larger than RTT** otherwise unnecessary retransmissions would be sent.

- TCP estimates this value by **sampling the RTT of successfully acknowledged segments** that have not been retransmitted (one-shot).
- Since the time of a sampled RTT ($samRTT$) **may fluctuate** (due to traffic congestion, load of the receiver, etc.) the RTT estimation ($estRTT$) is given by the **exponential weighted moving average (EWMA)**:

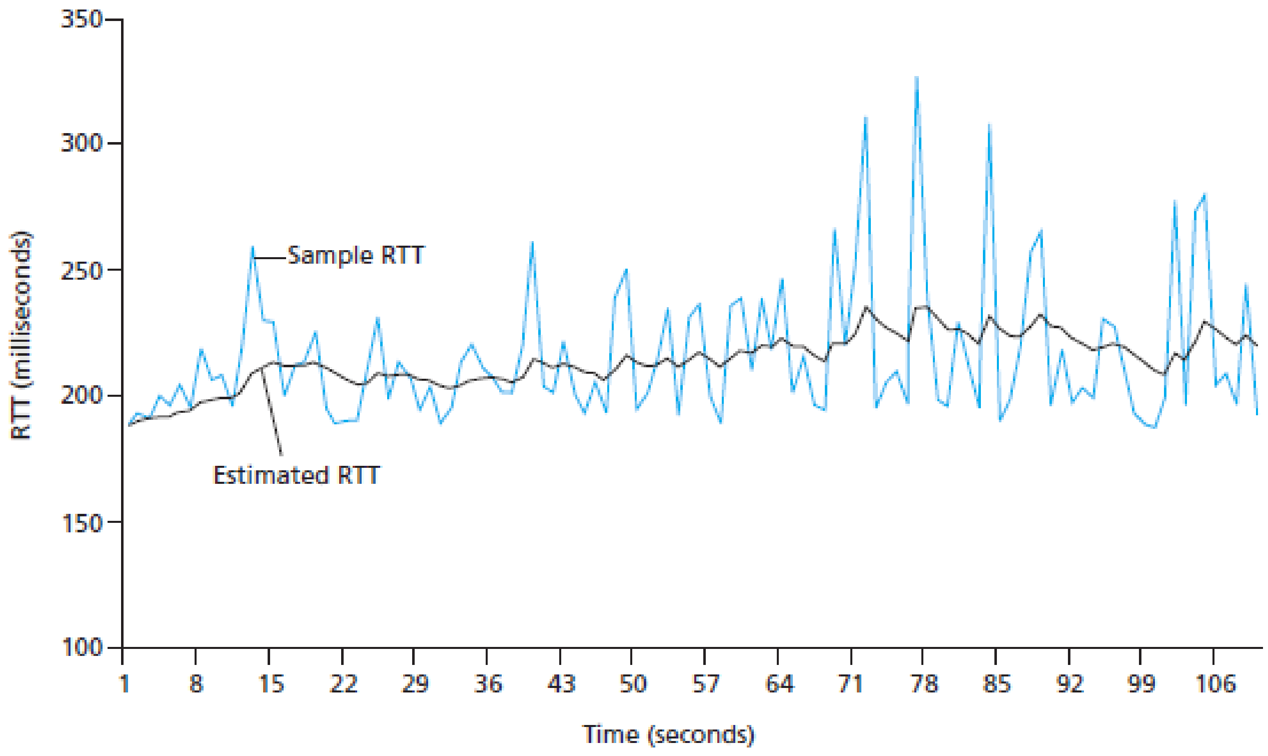
$$estRTT_t = (1 - \alpha)estRTT_{t-1} + \alpha\ samRTT_t$$

Where α is a parameter which is usually *0.125* (i.e., *1/8*).

Transport Layer

RTT Estimation

- This is an example of how RRT (sampled and estimated) behaves in a **real scenario** (between USA and France).
- It is also useful to measure the **variability of the RTT** ($devRTT$) from the difference between the sample ($samRTT$) and the estimation ($estRTT$):



$$devRTT_t = (1-\beta)devRTT_{t-1} + \beta|samRTT_t - estRTT_t|$$

Where β is a parameter which is usually *0.25* (i.e., *1/4*).

Transport Layer

Retransmission Timeout

- Having this estimations, **it is possible to define a retransmission timeout** for TCP that is not lower nor too higher than the estimated RTT:

$$timeout_t = estRTT_t + 4devRTT_t$$

- Here, the deviation of the RTT is used to set a **reasonable and adaptive margin** from the estimated RTT.
  - it increases when RTT oscillates, so the timeout window is larger **when we are uncertain** about our current RTT.

- By default, the **initial value** of the RTT (at t=0) is 1 second.

Transport Layer

Fast Retransmit

- One of the problems with timeout-triggered retransmissions is that the **timeout period can be relatively long**.
- To limit this issue TCP applies a mechanism called **fast retransmit** that uses **wrong ACK**:
  - When a TCP receiver receives a segment with a **sequence number that is larger than the expected one** it means that a segment has been reasonably missed.
  - The receiver **resends the old ACK** (duplicate ACK) having ack number of the expected segment.
  - If the **sender receives N duplicates** (typically 3 duplicates), it assumes that the previous segment has been lost so it is (fast) retransmitted **well before the deadline**.