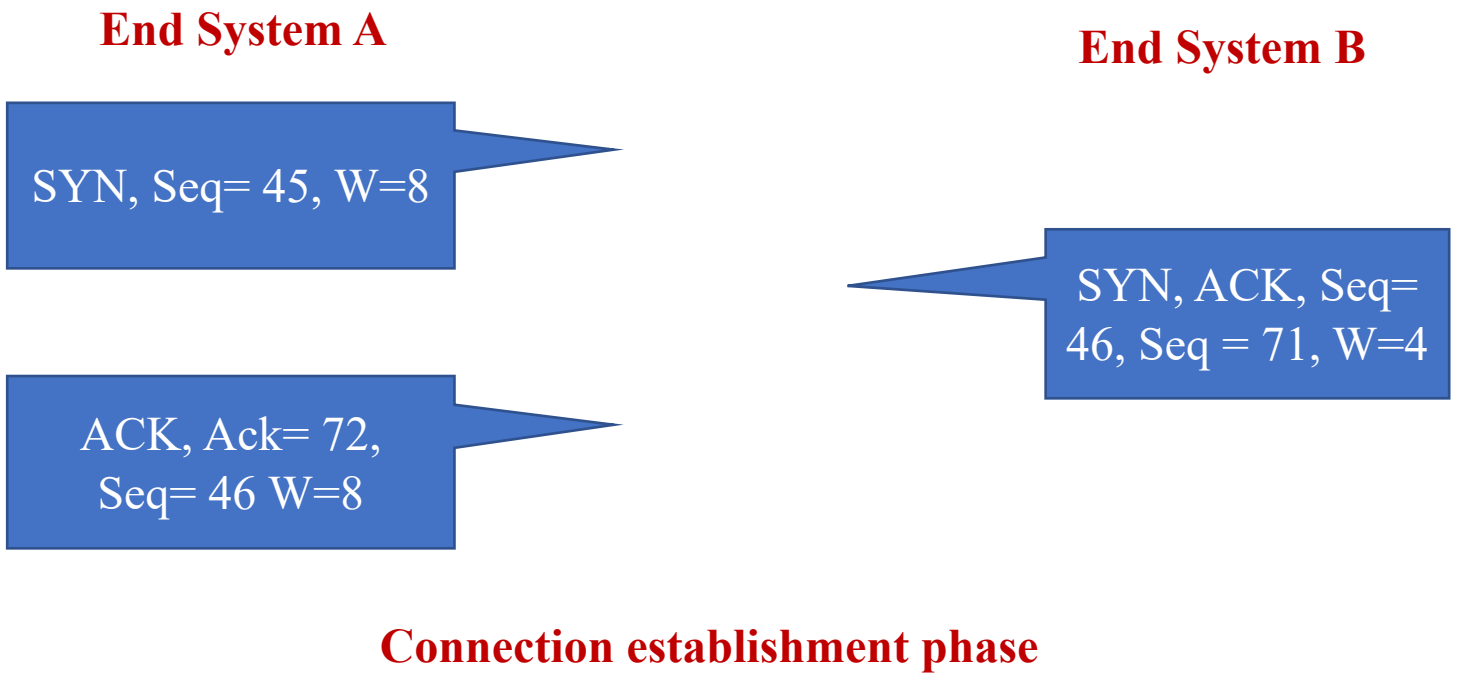
Few issues: TCP

Window size

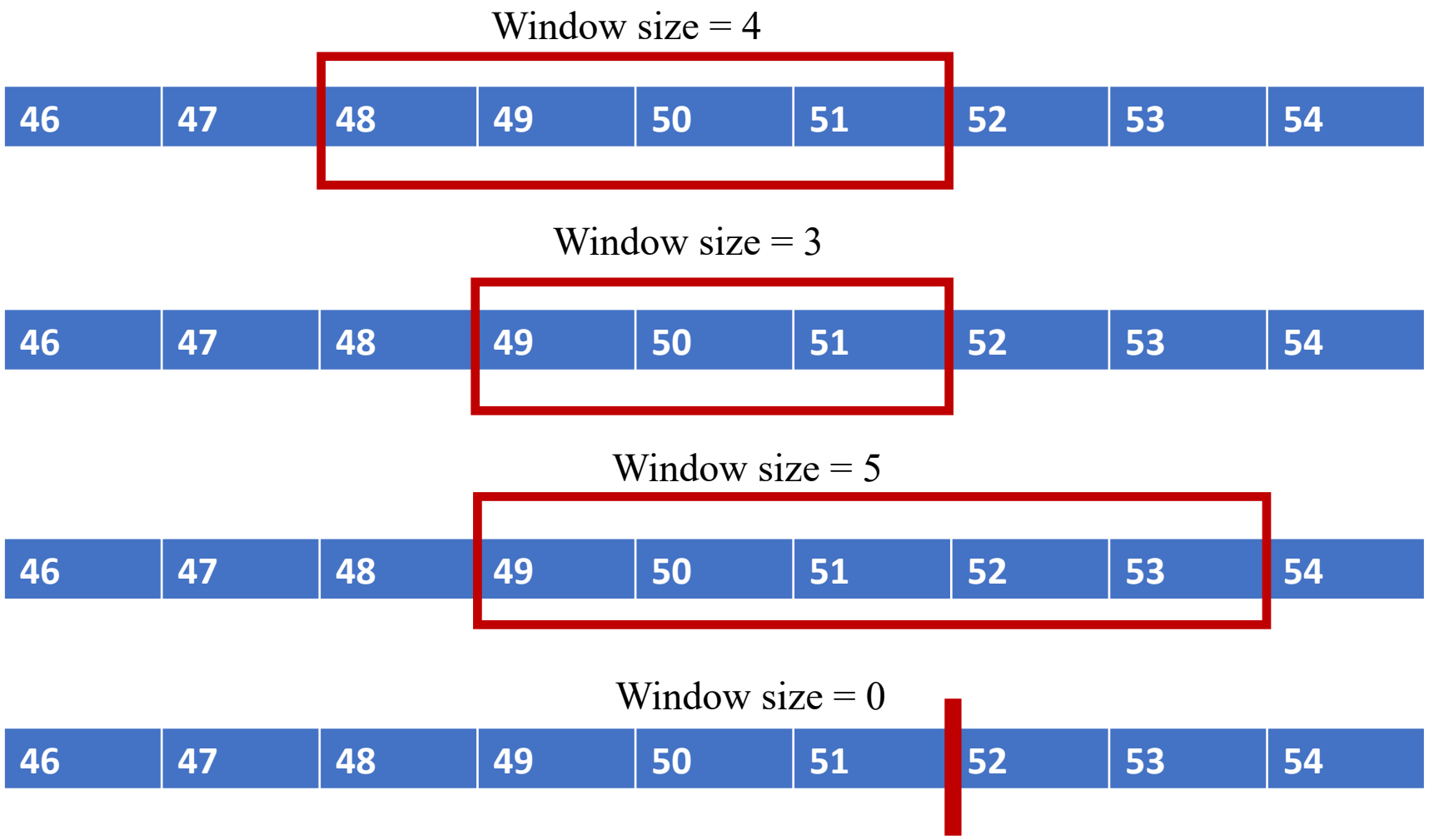
- The window size can be from 0 t0 65535 (as it is 16 bits long)
- This window size is variable and controlled be the receiving end
- If a receiver has full buffers, it shrinks the window at the sending end to stop further transmission
- The starting size of the window is negotiated during connection establishment phase

Window size: Example



Connection establishment phase

Sliding Window mechanism



Silly window syndrome

- **Serious performance issue:**
  - State of TCP connection, when the window size is shrunk to such a large extent that very small amount of user data is transported in each TCP segment
- **Cause:**
  - When the sending application process generates data at faster rate than the receiving application
  - Eventually the window size will be shrunk to zero and the receivers buffers will also full
  - If the application process at the receiving end reads one octet at a time from the receiver buffer,
    - right edge of the sender's window can slide one octet by sending an acknowledge
    - The sender can send the next TCP segment containing one octet of user data
      - Results in very inefficient use of network resource (1 octet payload but 40 by header)

Silly window syndrome

- Remedy:
  - Delayed Acknowledgement:
    - TCP standard specifies that transmission of the acknowledgement is delayed by the receiver till that time when it has either 50% vacant buffer or sufficient vacant buffer for 1 MSS
    - Maximum delay can be done is 500ms
  - Nagle's Algorithm:
    - This is a self-clocking algorithm for pacing generation of TCP segments
    - The first byte/octet handed over by the application process is sent immediately by the TCP layer.
    - Thereafter, transmission of TCP segments is clocked by the acknowledgements
    - The TCP layer accumulates additional octets until an acknowledgement arrives or the accumulated user data becomes equal to MSS.
    - This ensures reasonable amount of data is accumulated before sending

Estimation of Retransmission Timeout (RTO)

- Dur unreliable IP connection packets may lost and hence retransmission of packet is obvious
- If acknowledgment for a TCP segment is not received within a defined Retransmission Time (RTO), the TCP segment is retransmitted
- RTO time is generally kept greater than Round Trip Time (RTT) from the source to destination.
- If RTO is less than RTT then it makes congestion due to unnecessary retransmission, causing higher RTT and more loss of IP packets
- If RTO is kept much higher than RTT
  - Long idle time if a TCP packet is lost and there are no data in window to transmit

Estimation of RTT

- Good estimation of RTT is important
- RTT depends on
  - The distance between the source and destination and the transmission speed
    - RTT is to be determined for each connection
  - The congestion in the network layer
    - Therefore some adaptive algorithm is required for estimation

Estimation of RTT

- Common approaches to estimate RTT:
  - **Simple average**
  - **Exponential average**
  - Jacobson's algorithm
  - Karn's Algorithm

Estimation of RTT

- Simple Average:
  - Simple average of observed RTTs of last (N+1) TCP segment is taken
  - RTT is the duration between the time of transmission of a TCP segment and time of reception of its acknowledgement
  - Average RTT (ARTT) is calculated as

$$\boldsymbol{ARTT(N+1) = \frac{1}{N+1}\sum_{i}^{N+1} RTT(i)}$$

Here RTT(i) = the round trip time observed for the i-th TCP segment

- This can also be written as

$$\boldsymbol{ARTT(N+1) = \frac{N}{N+1}ARTT(N) + \frac{1}{N+1}RTT(N+1)}$$

- TCP does not use simple average for estimating RTT.
- It uses exponential average

Estimation of RTT

- We have $\boldsymbol{ARTT(N+1) = \frac{N}{N+1} ARTT(N) + \frac{1}{N+1} RTT(N+1)}$

- The last observation given a weight 1/(N+1) and the past average is given a weight of N/(N+1).
- RFC 793 specifies these weights based on nature of RTT observations
- The estimated RTT is referred to as smoothed RTT (SRTT) estimate
  - **SRTT(N+1) = $\boldsymbol{\alpha}$ SRTT(N) + (1 - $\boldsymbol{\alpha}$) RTT(N+1)**
    - $\boldsymbol{\alpha}$ is smoothing constant and vary from ***0 to 1, typically considered as 7/8***
- RTO can now be calculated as **RTO(N+1) = $\boldsymbol{\beta}$ SRTT(N+1)**, $\boldsymbol{\beta}$ varies from ***1.3 to 2***

Example:

- If the TCP round-trip time, RTT, is currently 30 msec and the following acknowledgements come in after 26, 32, and 24 msec, respectively, what is the new RTT estimate using the Jacobson algorithm? Use $\alpha = 0.9$.
- Solution:

$$SRTT = \alpha\, SRTT + (1- \alpha)\, R$$

$$SRTT1 = 0.9 \times 30 + (1 - 0.9) \times 26 = 29.6$$

$$SRTT2 = 0.9 \times 29.6 + (1 - 0.9) \times 32 = 29.84$$

$$SRTT3 = 0.9 \times 29.84 + (1 - 0.9) \times 24 = 29.256$$