TCP III - Error Control

TCP Error Control

ARQ Error Control

- Two types of errors:
  - **Lost packets**
  - **Damaged packets**

- Most Error Control techniques are based on:
  1. Error Detection Scheme (Parity checks, CRC).
  2. Retransmission Scheme.

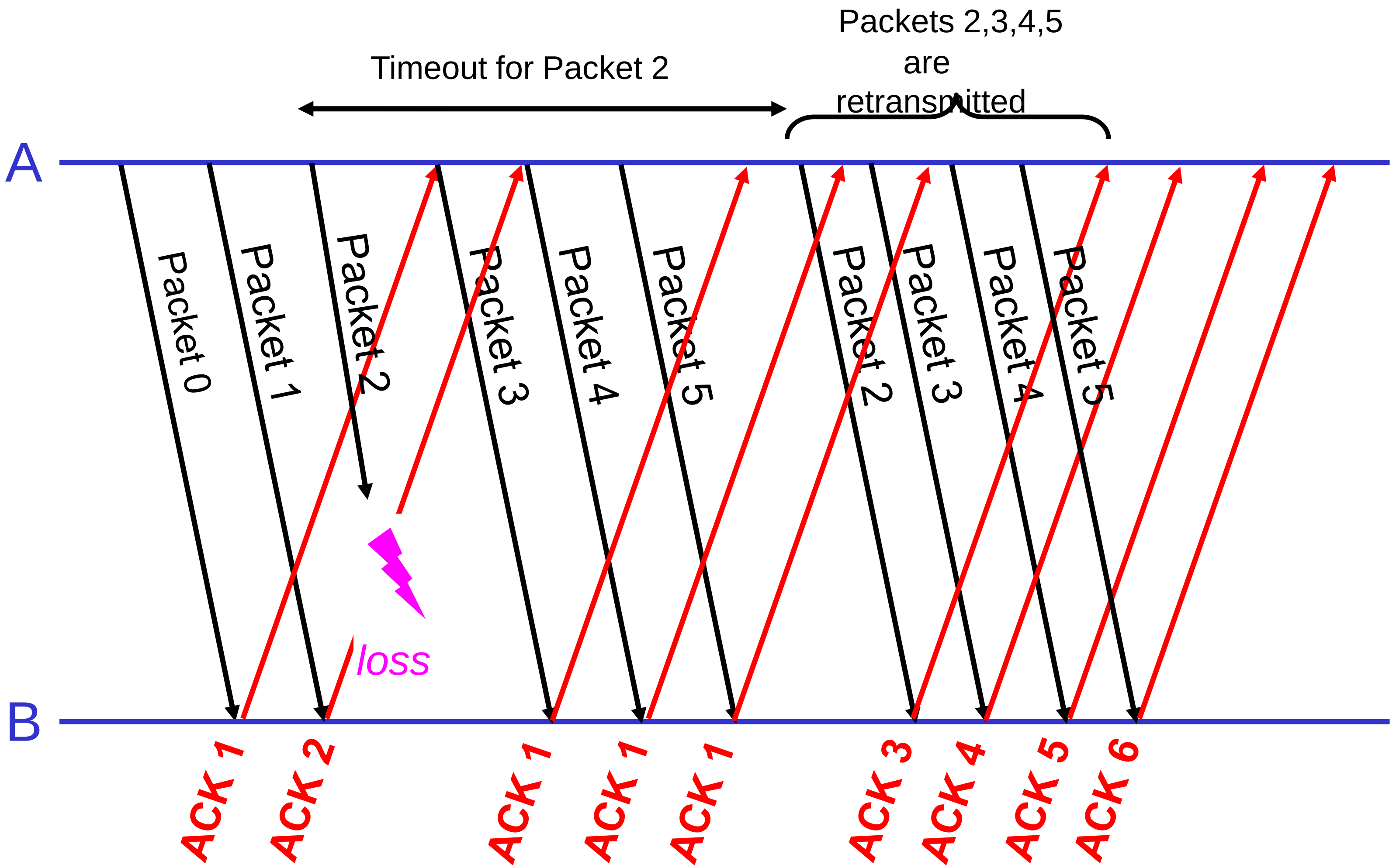
- Error control schemes that involve error detection and retransmission of lost or corrupted packets are referred to as **Automatic Repeat Request (ARQ) error control**.

Background: ARQ Error Control

- The most common ARQ retransmission schemes:
  - Stop-and-Wait ARQ
  - Go-Back-N ARQ
  - Selective Repeat ARQ
- The protocol for sending ACKs in all ARQ protocols are based on the sliding window flow control scheme
- TCP uses a version of the Go-Back-N Protocol

Background: Go-Back-N ARQ

- Go-Back-N sends cumulative acknowledgments



Retransmissions in TCP

- A TCP sender retransmits a segment when it assumes that the segment has been lost:

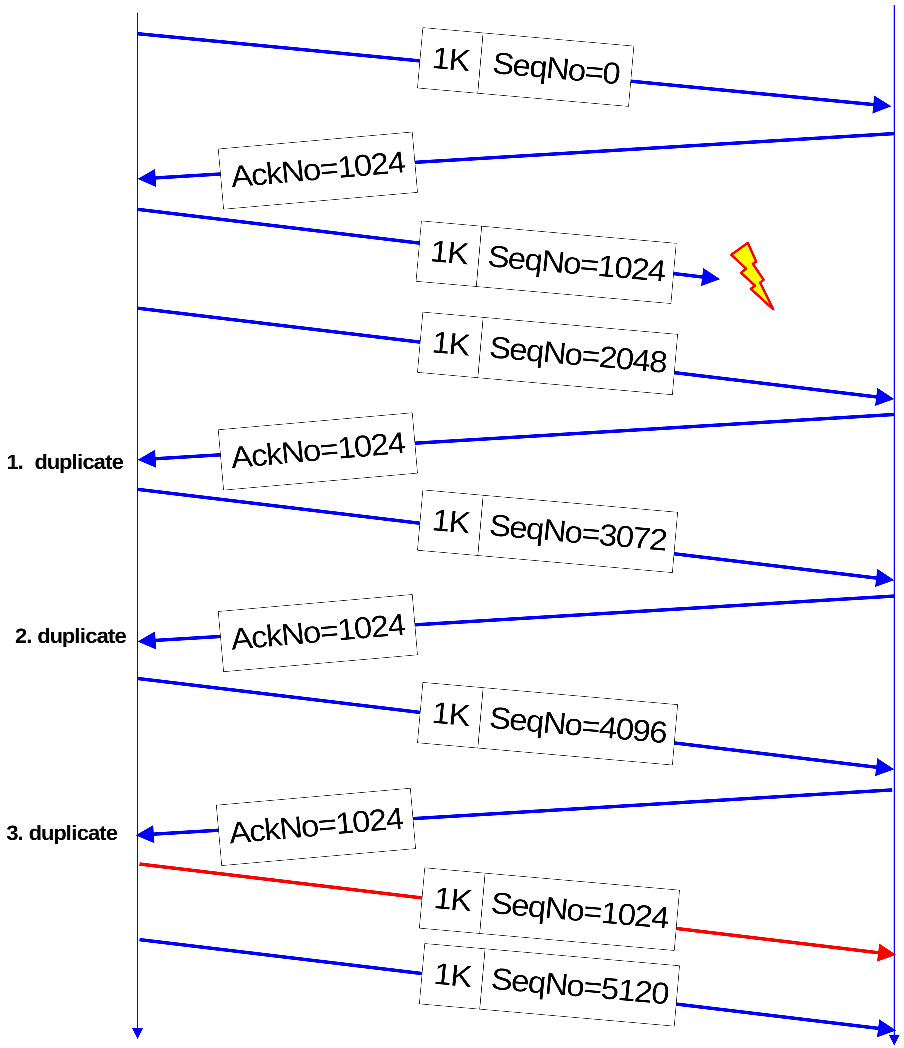
  1. No ACK has been received and a timeout occurs
  2. Multiple ACKs have been received for the same segment

Receiving duplicate ACKs

- If three or more duplicate ACKs are received in a row, the TCP sender believes that a segment has been lost.

- Then TCP performs a retransmission of what seems to be the missing segment, without waiting for a timeout to happen.

- This can fix losses of single segments



Retransmission Timer

- TCP sender maintains one retransmission timer for each connection
- When the timer reaches the retransmission timeout (RTO) value, the sender retransmits the first segment that has not been acknowledged

- The timer is started when
  1. When a packet with payload is transmitted and timer is not running
  2. When an ACK arrives that acknowledges new data,
  3. When a segment is retransmitted

- The timer is stopped when
  – All segments are acknowledged

How to set the timer

- **Retransmission Timer:**
  - The setting of the retransmission timer is <u>crucial</u> for good performance of TCP
  - Timeout value too small → results in unnecessary retransmissions
  - Timeout value too large → long waiting time before a retransmission can be issued
  - A problem is that the delays in the network are not fixed
  - Therefore, the retransmission timers must be adaptive

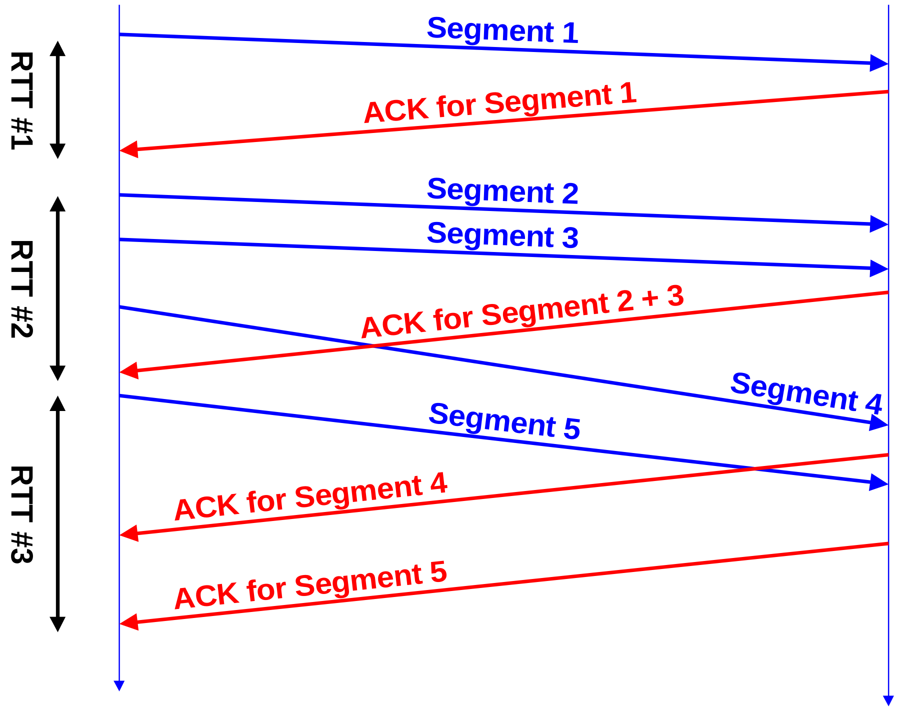
Setting the value of RTO:

- The RTO value is set based on round-trip time (RTT) measurements that each TCP performs

• Each TCP connection measures the time difference between the transmission of a segment and the receipt of the corresponding ACK

• There is only one measurement ongoing at any time (i.e., measurements do not overlap)

• Figure on the right shows three RTT measurements



Setting the RTO value

- RTO is calculated based on the *RTT* measurements
  - Uses an exponential moving average to calculate estimators for delay (*srtt*) and variance of delay (*rttvar*) from

- The RTT measurements are smoothed by the following estimators *srtt* and *rttvar*:

$$srtt_{n+1} = \alpha \; RTT + (1- \alpha) \; srtt_n$$
$$rttvar_{n+1} = \beta \, (\, | \, RTT - srtt_n \, | \,) + (1- \beta) \; rttvar_n$$
$$RTO_{n+1} = srtt_{n+1} + 4 \; rttvar_{n+1}$$

  - The gains are set to $\alpha = 1/4$ and $\beta = 1/8$

Setting the RTO value (cont'd)

- Initial value for RTO:
  - Sender should set the initial value of RTO to

$$RTO_0 = 3 \text{ seconds}$$

- RTO calculation after first RTT measurements arrived

$$srtt_1 = RTT$$
$$rttvar_1 = RTT / 2$$
$$RTO_1 = srtt_1 + 4\ rttvar_{n+1}$$

- When a timeout occurs, the RTO value is doubled

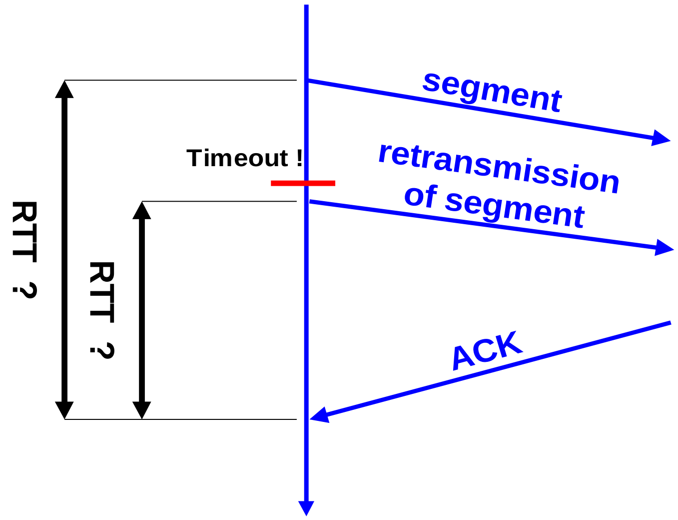
$$RTO_{n+1} = max\ (\ 2\ RTO_n,\ 64)\ seconds$$

This is called *an exponential backoff*

Karn’s Algorithm

If an ACK for a retransmitted segment is received, the sender cannot tell if the ACK belongs to the original or the retransmission.

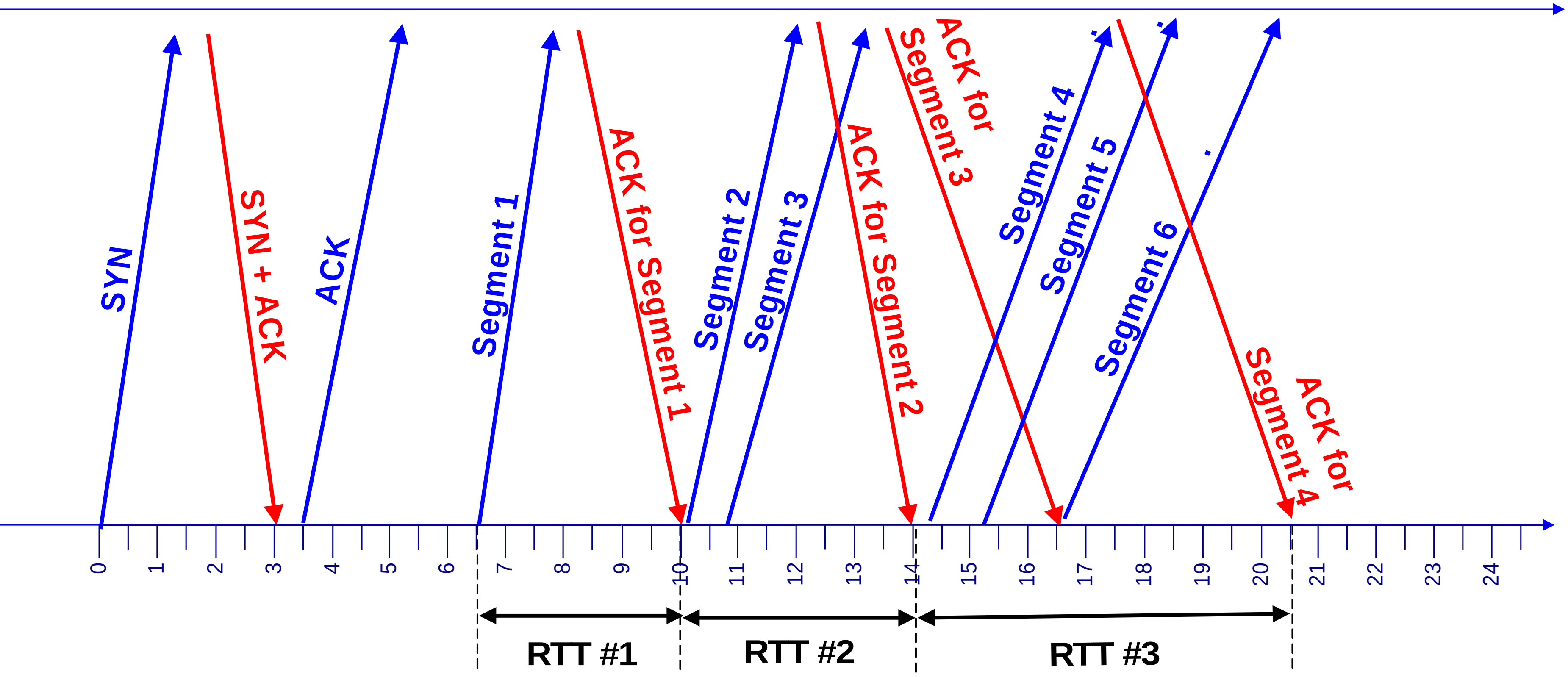
→ RTT measurements is ambiguous in this case



Karn’s Algorithm:

- Don’t update *RTT* on any segments that have been retransmitted
- Restart RTT measurements only after an ACK is received for a segment that is not retransmitted

RTO Calculation: Example



TCP Retransmission Timer

- **Retransmission Timer:**
  - The setting of the retransmission timer is crucial for efficiency
  - Timeout value too small → results in unnecessary retransmissions
  - Timeout value too large → long waiting time before a retransmission can be issued

  - A problem is that the delays in the network are not fixed
  - Therefore, the retransmission timers must be adaptive

Round-Trip Time Measurements

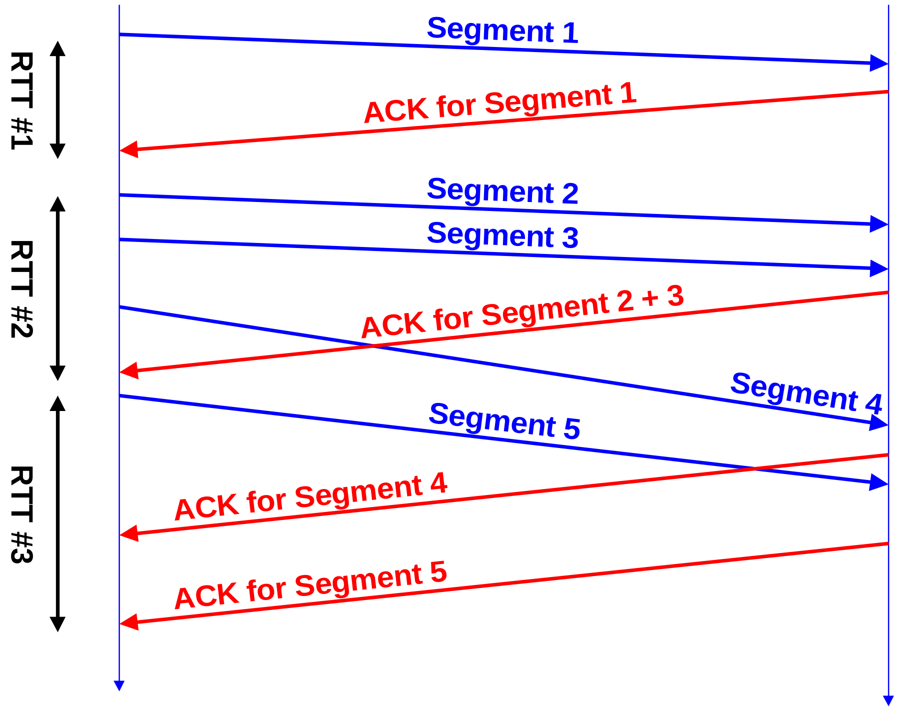
- The retransmission mechanism of TCP is adaptive
- The retransmission timers are set based on round-trip time (RTT) measurements that TCP performs

The RTT is based on time difference between segment transmission and ACK

But:

TCP does not ACK each segment

Each connection has only one timer



Round-Trip Time Measurements

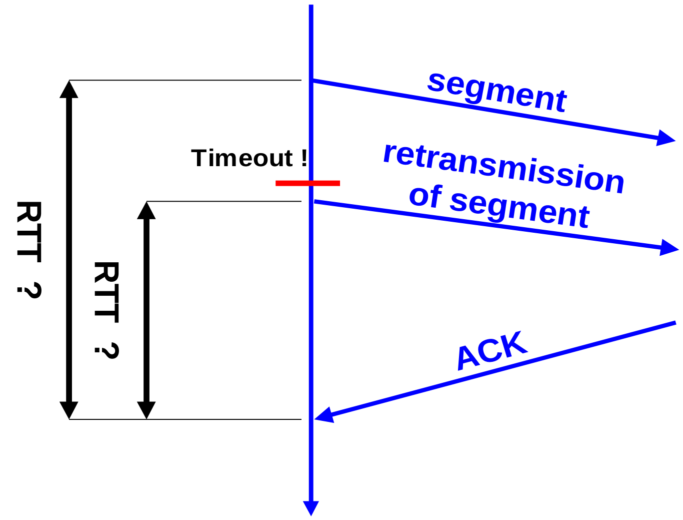
- Retransmission timer is set to a **Retransmission Timeout (RTO) value**.
- RTO is calculated based on the *RTT* measurements.
- The RTT measurements are smoothed by the following estimators *srtt* and *rttvar*:

$$srtt_{n+1} = \alpha \ RTT \ + (1- \alpha) \ srtt_n$$
$$rttvar_{n+1} = \beta (| RTT - srtt_{n+1} |) + (1- \beta) \ rttvar_n$$
$$RTO_{n+1} = srtt_{n+1} + 4 \ rttvar_{n+1}$$

- The gains are set to $\alpha = 1/4$ and $\beta = 1/8$
- $srtt_0 = 0$ *sec*, $rttvar_0 = 3$ *sec, Also:* $RTO_1 = srtt_1 + 2 \ rttvar_1$

Karn's Algorithm

- If an ACK for a retransmitted segment is received, the sender cannot tell if the ACK belongs to the original or the retransmission.



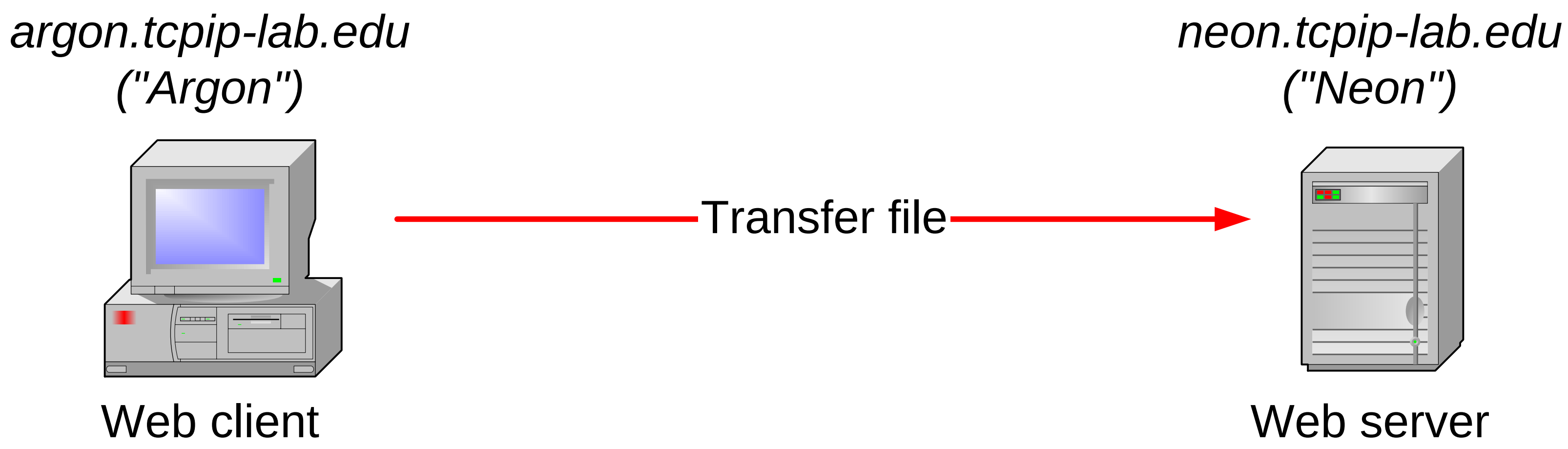
Karn's Algorithm:

Don't update *srtt* on any segments that have been retransmitted.

Each time when TCP retransmits, it sets:

$RTO_{n+1} = max\ (\ 2\ RTO_n, 64)$ (exponential backoff)

Measuring TCP Retransmission Timers



- Transfer file from Argon to neonn
- Unplug Ethernet of Argon cable in the middle of file transfer

Interpreting the Measurements

- The interval between retransmission attempts in seconds is:

  1.03, 3, 6, 12, 24, 48, 64, 64, 64, 64, 64, 64, 64.
- Time between retrans-missions is doubled each time (Exponential Backoff Algorithm)
- Timer is not increased beyond 64 seconds
- TCP gives up after 13th attempt and 9 minutes.