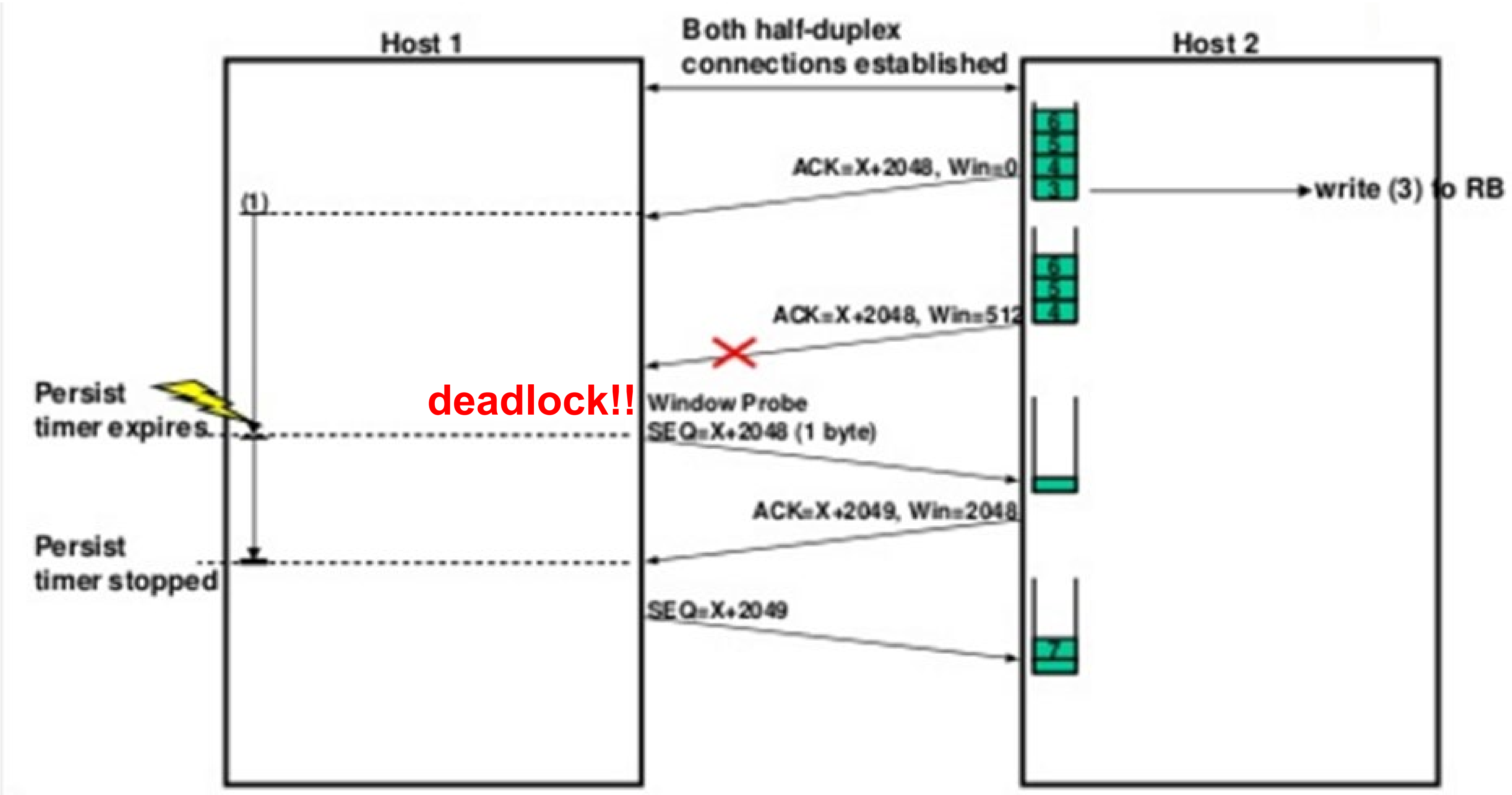
TCP Error Control (cont.)

Persistence Timer – timer used to correct 0-size receive window deadlock (see slide 11, SWS)

- **assume**:
  - (a) sender receives and ACK with RcvWindow = 0
  - (b) sender stops transmission
  - (c) receiver eventually frees buffer and sends and ACK with non-zero RcvWindow, but **this ACK gets lost**!
- **consequence**: **deadlock** – sender may wait forever!
- **SOLUTION**: each time a sender gets a 0-window segment it starts **persistence timer**
  - (a) if timer goes off send a probe segment with only 1-byte of data
  - (b) probe segments alert receiving TCP that acknowledgment may have got lost and should be resent
- initially **persistence timer = 2*RTT = retransmission timer**; after that it is doubled until reaching 60 sec; finally one probe every 60 sec …

TCP Error Control (cont.)

Example **[Persistence Timer]**



TCP Error Control (cont.)

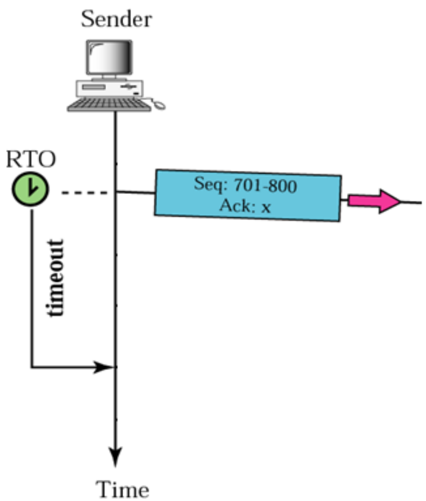
Retransmission Timer / Timeout (RTO) – **determines how long to wait for ACK of a previously sent segment before retransmission**

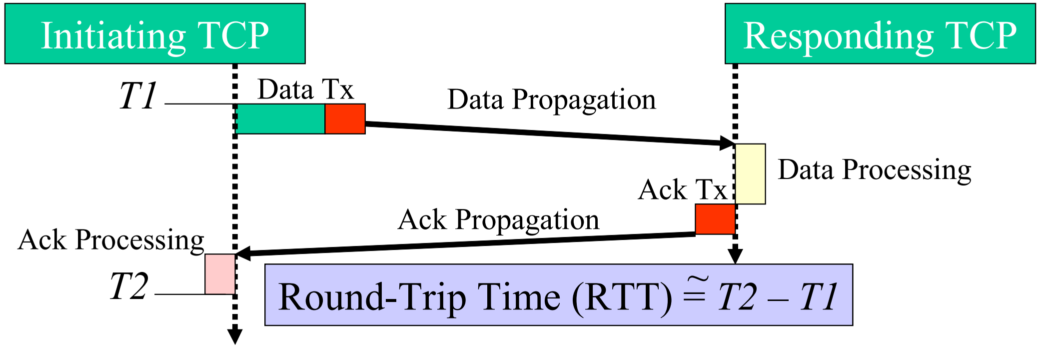
- **depends on distance and network congestion ⇒ different for each connection, may change during connection-lifetime**

estimated   accounts for irregularities in the network

$$\mathbf{RTO} = \mathbf{RTT} + \mathbf{\Delta(RTT)}, \quad \text{or simply } \mathbf{RTO} \approx \mathbf{2 * RTT}$$

$$\mathbf{RTT} \approx \mathbf{t(ACK\ received) - t(data\ sent)}$$





Calculation of RTT – **TCP determines approximate RTT between devices and adjusts it over time to compensate for increases and decreases**

- **does not use a single number – instead, uses dynamic / adaptive formulas that employ a sequence of earlier observations of RTT**

TCP Error Control (cont.)

Average / Smoothed RTT Update Formulas

1) **Simple Average**

$$\underbrace{\mathrm{ARTT}(K+1)}_{\text{estimated}} = \frac{1}{K+1}\sum_{i=1}^{K+1} \underbrace{\mathrm{RTT}(i)}_{\text{observed}}$$

easier to calculate – not necessary to calculate entire summation

$$\boxed{\mathrm{ARTT}(K+1) = \frac{K}{K+1}\mathrm{ARTT}(K) + \frac{1}{K+1}\mathrm{RTT}(K+1)}$$

- **drawback**: **gives equal weight to each observation**
- **solution**: **give greater weight to more recent instances since they are more likely to reflect future behavior**

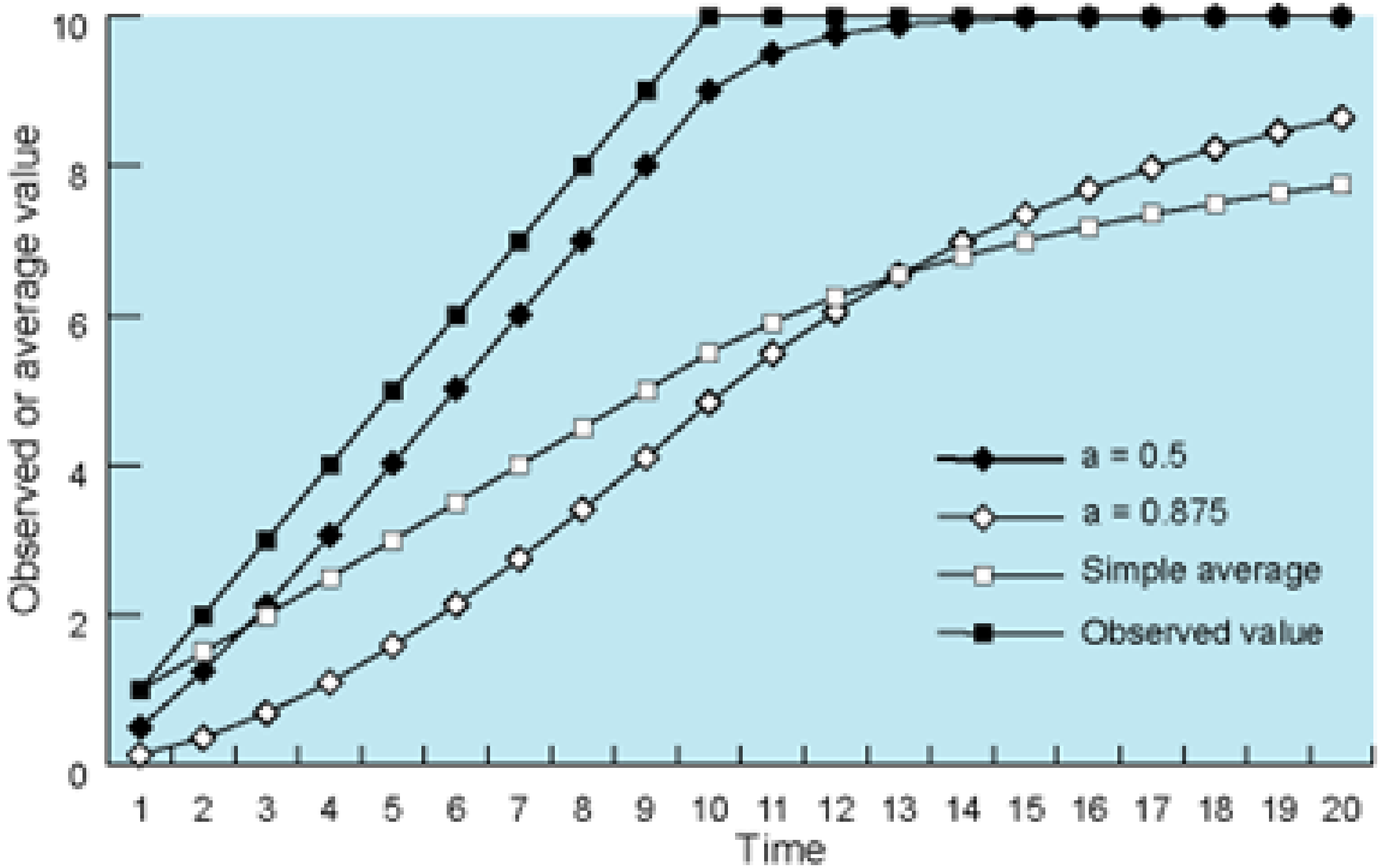
2) **Exponential (Smoothed) Average**

$$\boxed{\mathrm{SRTT}(K+1) = \alpha \cdot \mathrm{SRTT}(K) + (1-\alpha)\cdot \mathrm{RTT}(K+1)}$$

- **$0<\alpha<1$ $\Rightarrow$ smaller α gives greater weight to more recent observations**
- **$\alpha=0.85 \Rightarrow$ last 10 observations matter**
- **$\alpha=0.5 \Rightarrow$ only last 4-5 observations matter**

TCP Error Control (cont.)

Example **[simple vs. exponential average – significance of choosing right α]**

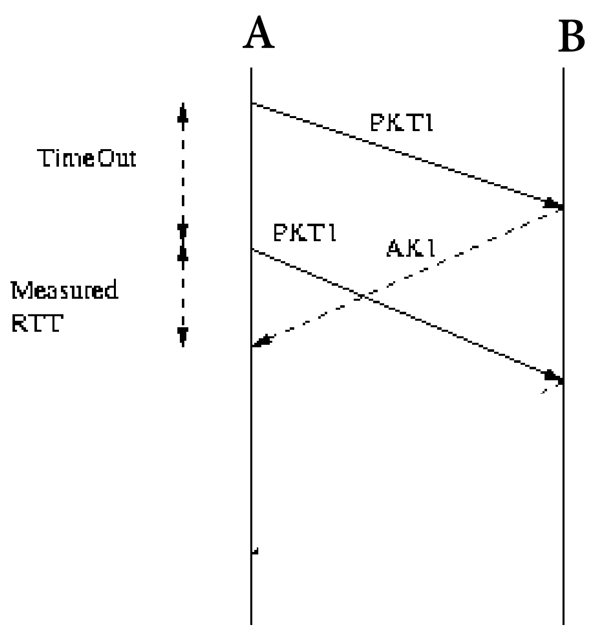


(a) Increasing function

TCP Error Control (cont.)

Karn's Algorithm – **assume an ACK has arrived after <u>retransmission</u> of a TCP segment – does this ACK correspond to the original or retransmitted segment?!**

- **the source cannot distinguish between two cases**
- **misinterpreatation can cause RTT to be set much too high or much too low**
- **solution – Karn's Algorithm: <u>do not update the value of ARTT with RTTs measured for retransmitted segments</u>**



Exercise

1. In _________________ data are sent or processed at a very inefficient rate, such as 1 byte at a time.

   (a) Nagle's syndrome
   (b) silly window syndrome
   (c) sliding window syndrome
   (d) delayed acknowledgment.

2. Suppose Host A sends two TCP segments back to back to Host B over a TCP connection. The first segment has a sequence number 90, the second segment has sequence number 110.

   (a) How much data is in the first segment?
   (b) Suppose that the first segment is lost but the second segment arrives at B. In the ACK that Host B sends to Host A, what will be the ACK number?