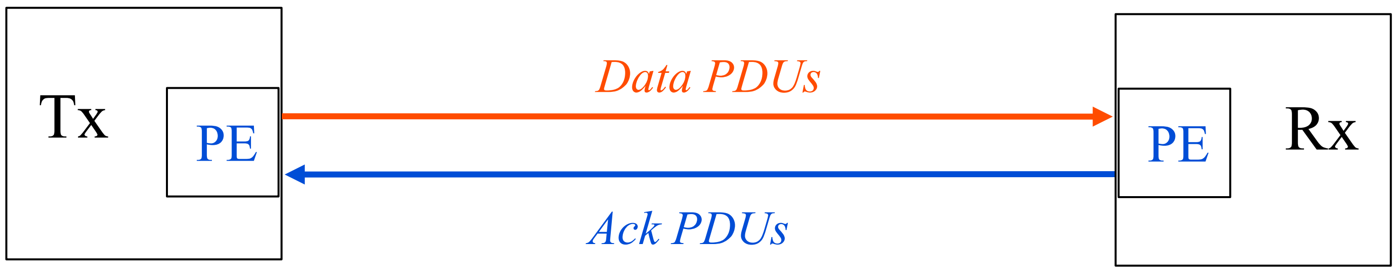
COMP2011 -- **Networks**, Databases and Graphics

Dealing Errors at Packet Level

Dr. Shi Zhou
Department of Computer Science
University College London

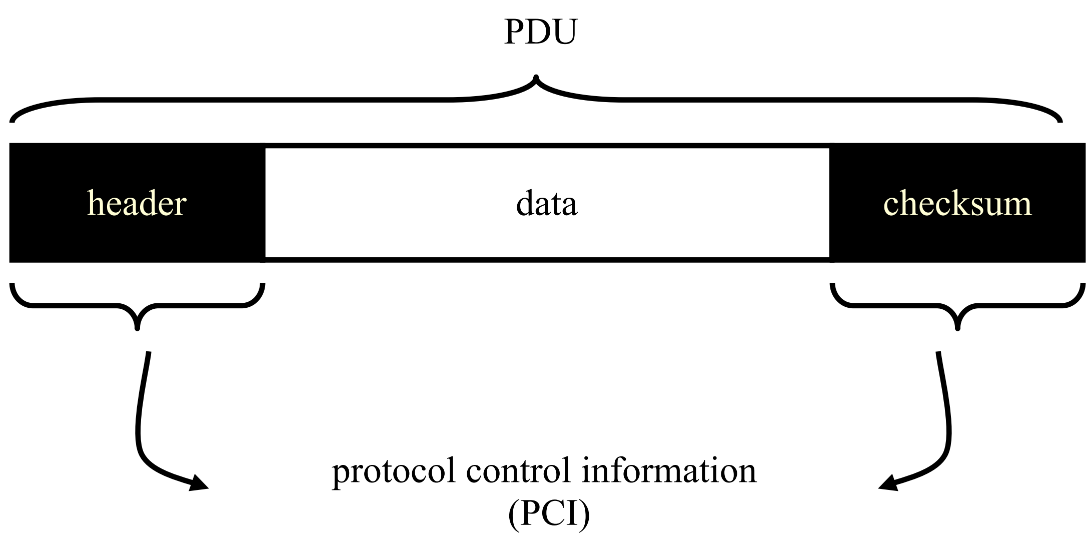
Scenario



- One host (the **transmitter**, TX) sends data to another host (the **receiver**, Rx)
- Data is sent in packets.
- Hosts use a ***protocol*** to co-ordinate the transmission of the data so as to ensure reliable delivery: repeat lost data, control data flow.
- Hosts have ***protocol entities*** *(PEs),* programs, to implement the protocol.
- PEs send ***protocol data units*** *(PDUs)* to each other: these are packets carrying the data from Tx to Rx and replies from Rx to Tx

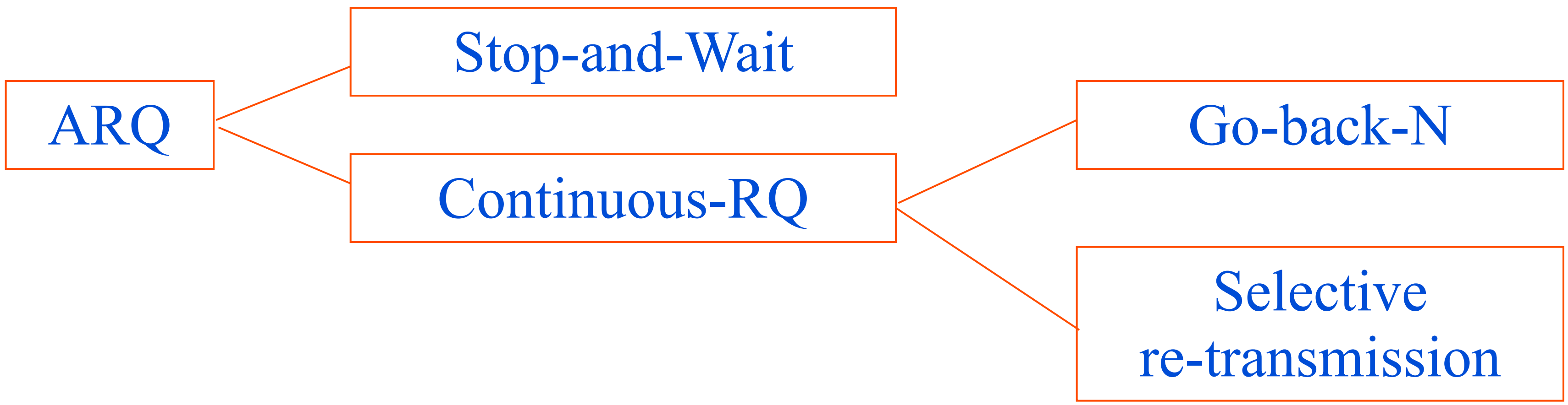
ARQ Protocol

- Automatic Repeat Request (ARQ) protocol
- Protocol data units (PDU)
  - Data - payload
  - Protocol control information (PCI) - protocol overhead
    - header
    - checksum, CRC (Cyclic redundancy check)



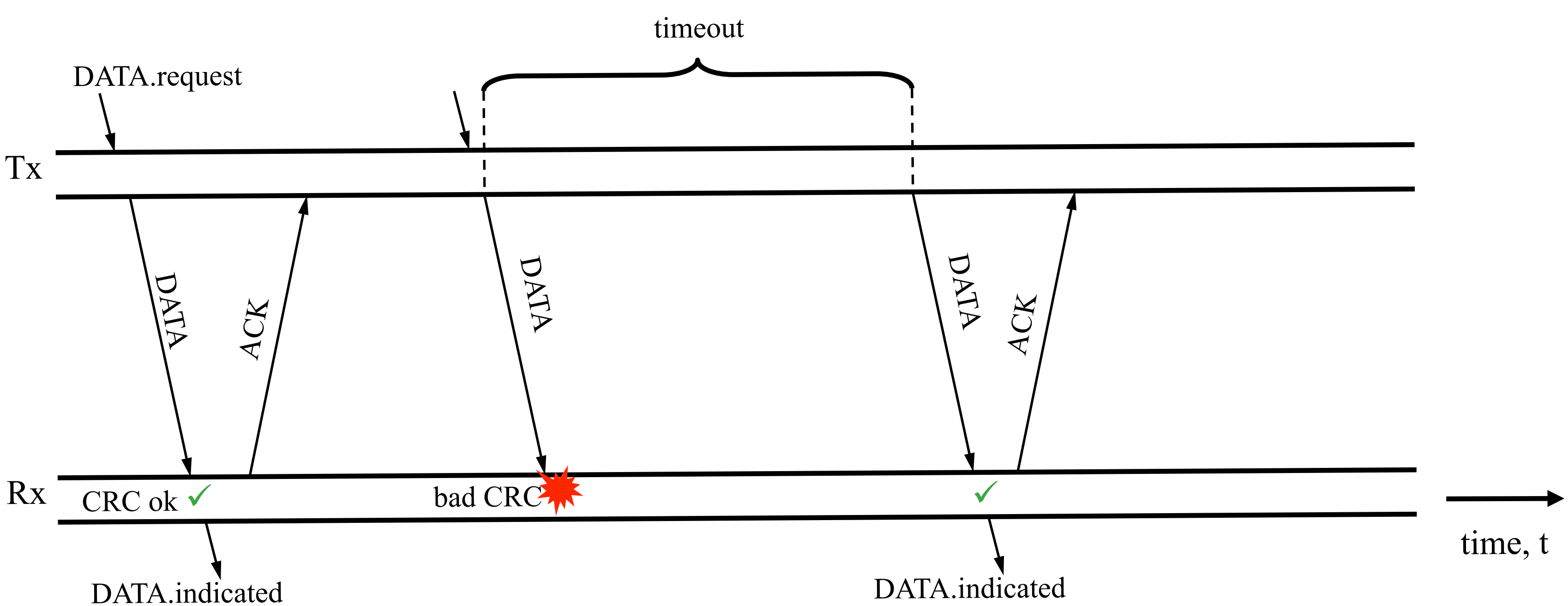
ARQ Protocol

- Error control – repeat damaged PDUs
- Flow control -- limited resources at end-systems



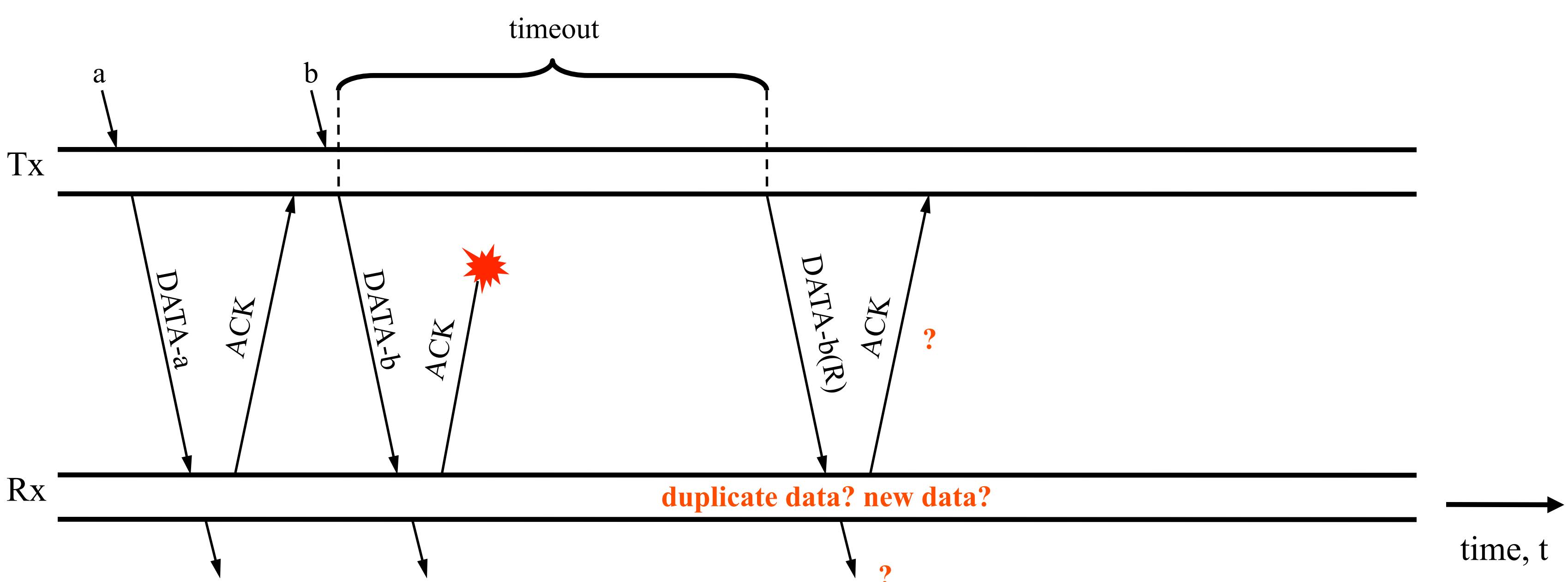
ARQ (basic):Stop-and-Wait

- Tx sends DATA PDU
- Rx checks CRC in the DATA PDU; if OK, Rx sends ACK PDU
- Tx uses timeout
  - Re-transmit DATA if no ACK received
  - Must allow for round-trip time (RTT)



A problem with Stop-and-Wait

- If no ACK received at Tx, it must resend Data
  - either lost/bad DATA or lost ACK?
  - if lost ACK, Rx receives duplicate DATA!
- How to detect duplicates?
  - Sequence number (SN)
  - Last packet or new pack? only 1 bit SN required

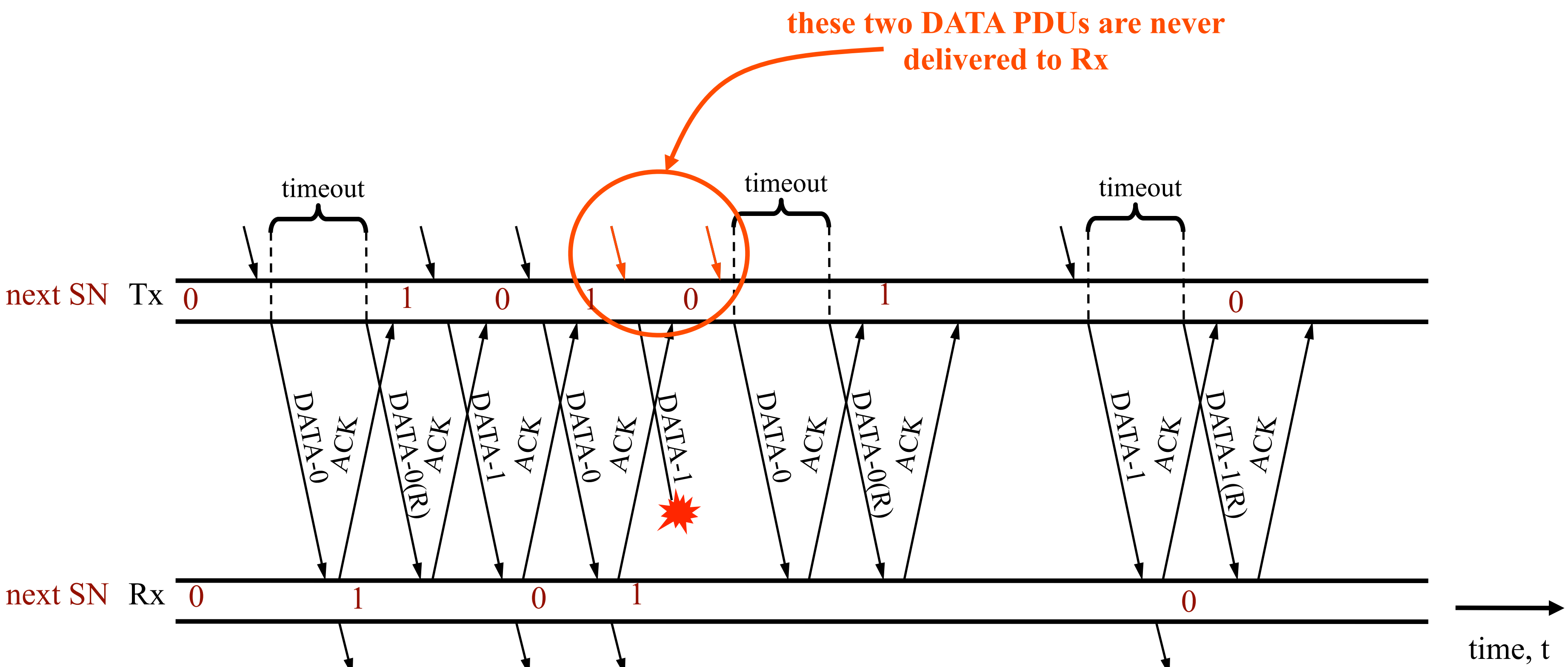


Improved Stop-and-Wait

- Suppose 1 bit SNs are only in DATA PDUs:

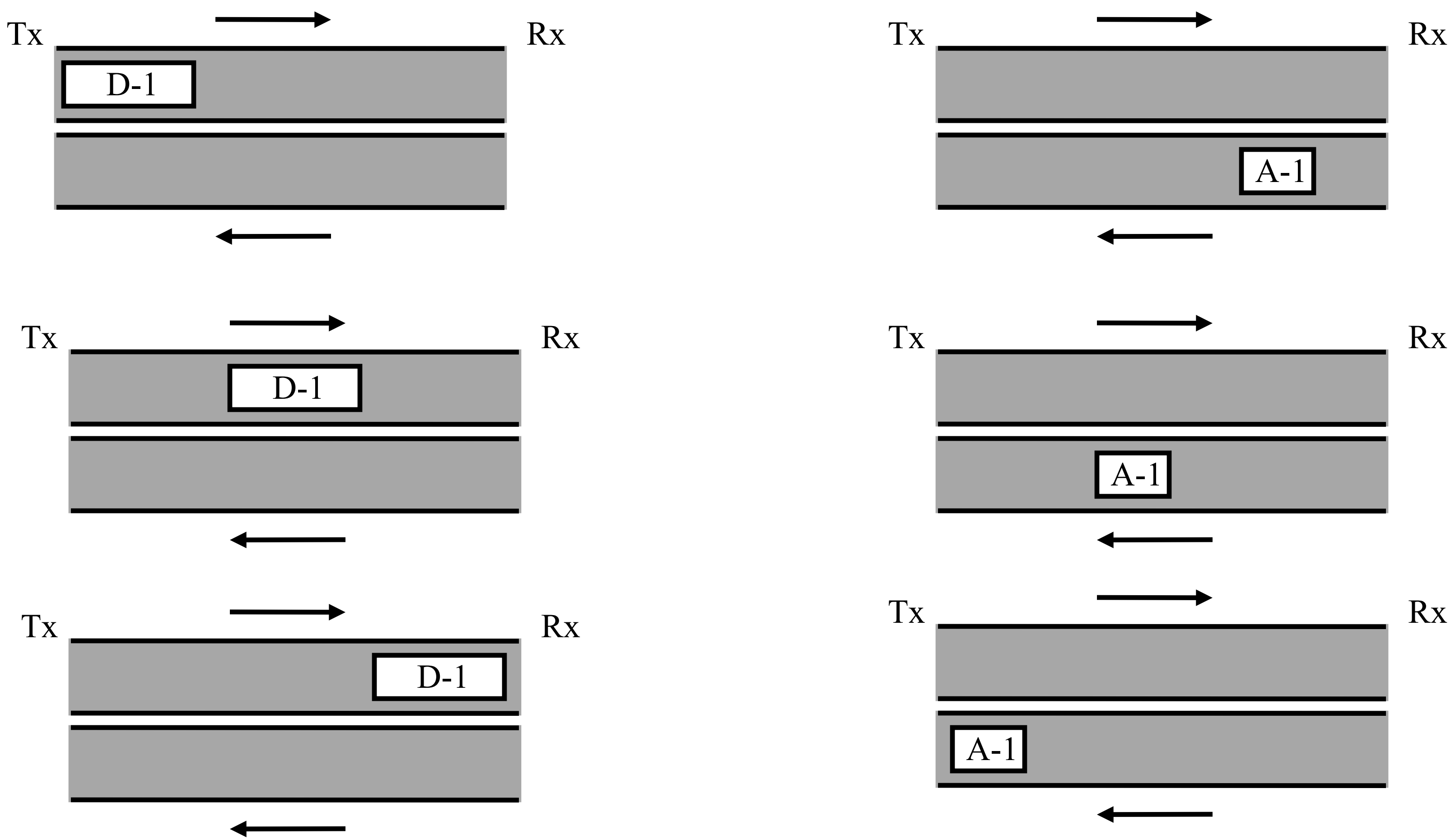
– short timeout causes problem as delayed ACK can cause failure

– need SN in ACK PDUs as well



Stop-and-Wait:

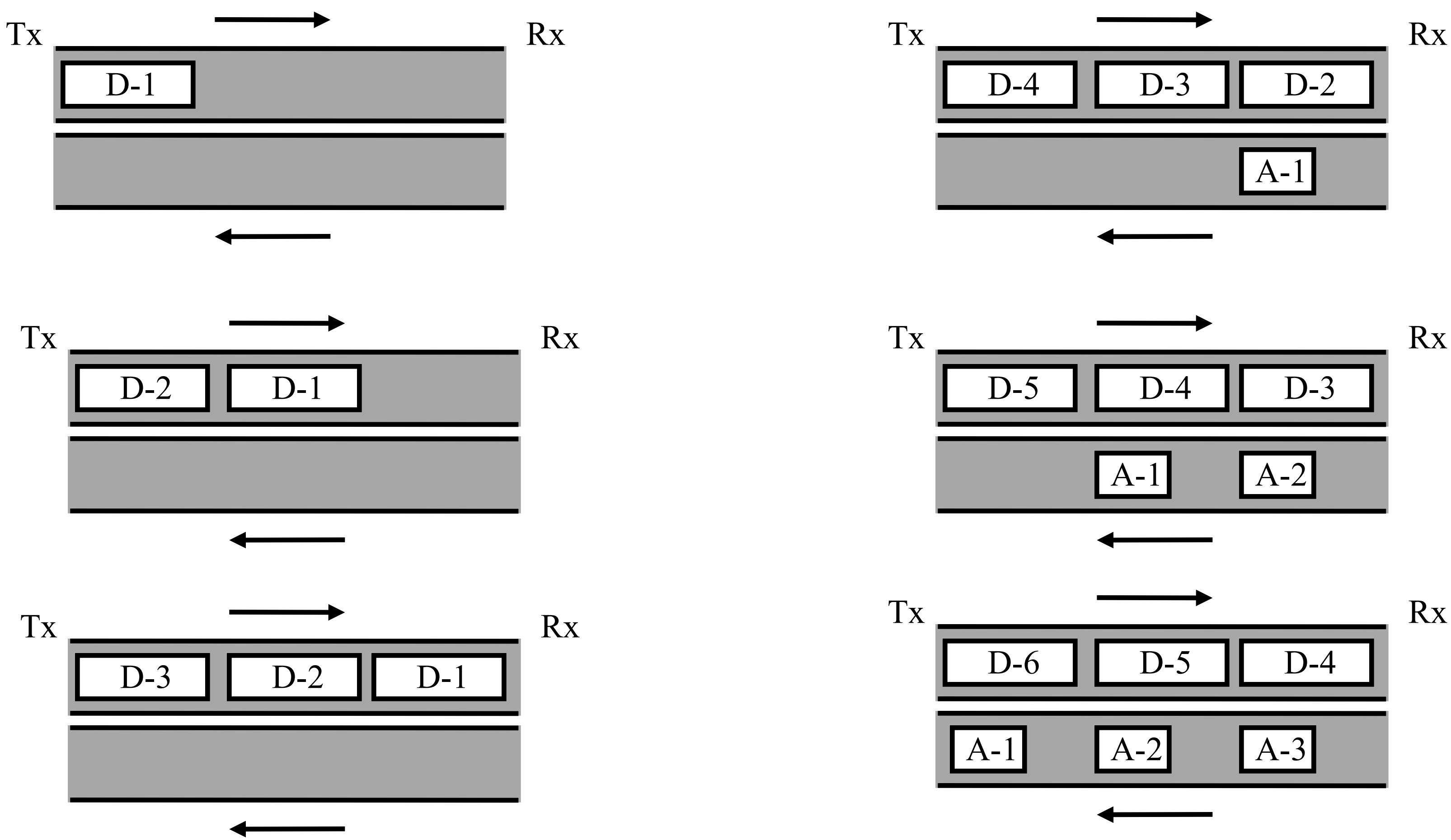
inefficient where propagation time >> packet transmit time



Continuous-RQ

Further improvement on Stop-and-Wait

- Tx continues sending data before first ACK:
  - **Pipelining** – many packets in transit

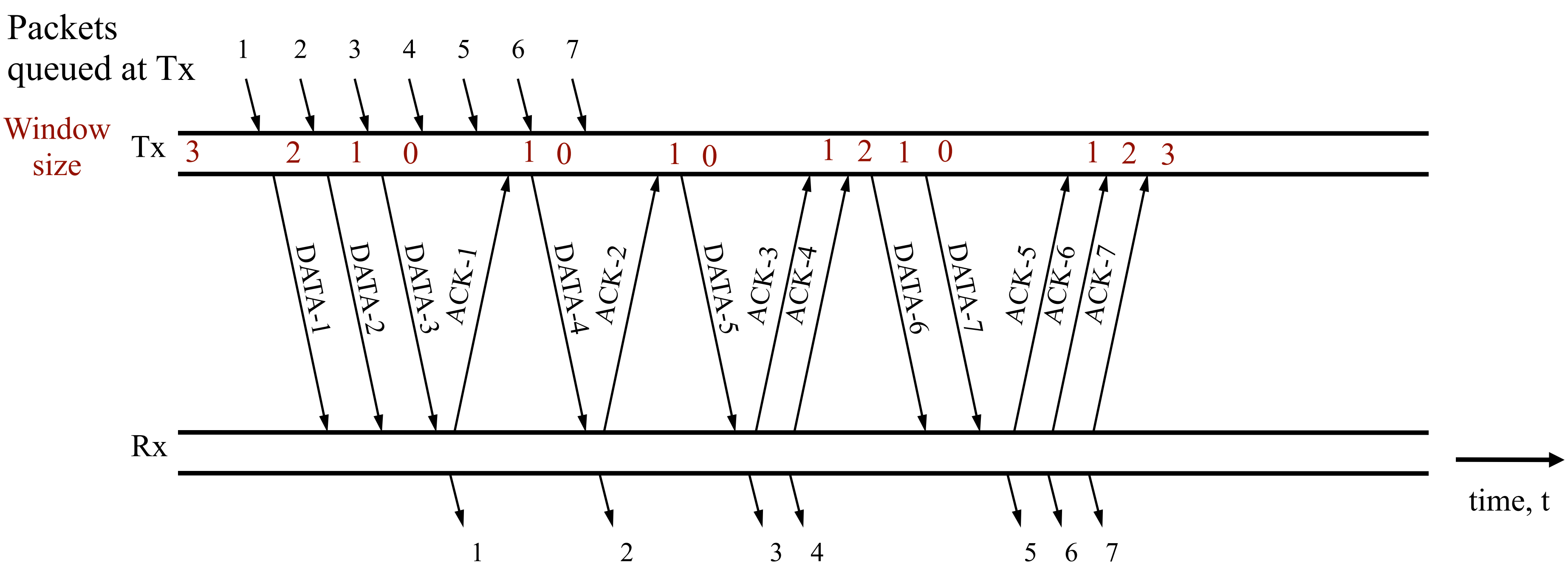


Continuous-RQ

- Capacity of "pipe":
  - **bandwidth delay product: data-rate × RTT**
  - For a satellite link at 64 Kbps with RTT=540ms, 64Kbps × 540ms = 34.6Kb, i.e. up to 36 PDU (with 1000bit) "in flight" at any one time.
  - On a LAN, transmission time is greater than propagation delay, so there will be less than one PDU in flight on average.
  - Rx must deal with all received data
- Must queue data at Tx in case need to retransmit
- Need flow control (from Rx to Tx) to control data flow.

Continuous-RQ: flow control

- **Window** size:
  – Tx PDU "credits"
  – can only send if window (credits) available
  – should be sufficient to keep the pipeline full
- Rx may also have notion of window:
  – must know which SNs to expect



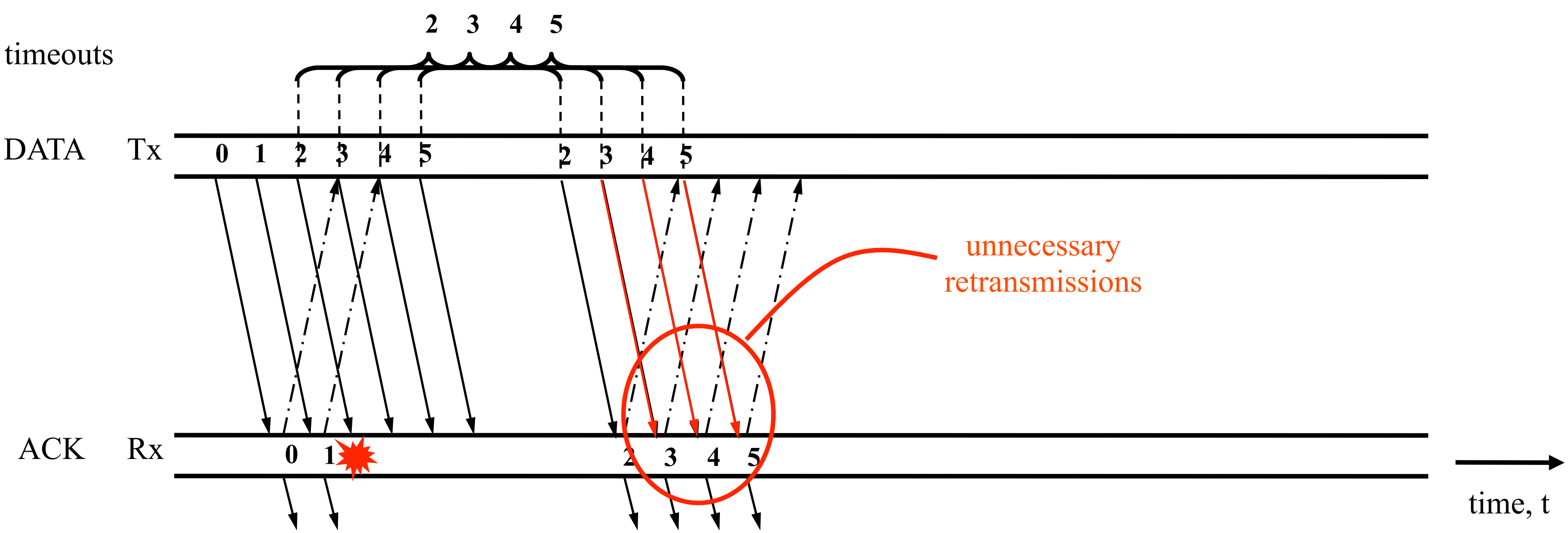
Continuous-RQ: dealing with lost PDUs

- Continuous RQ:
  - works well for reliable underlying service
- What if some PDUs:
  - get lost/corrupted?
  - arrive out of order?
- Re-transmission strategies(if packet lost or corrupted)
  - Go-back-N (re-send missing packet and all following)
    - Rx window size is one
  - Selective re-transmission (re-send just missing packet)
  - trade-off: network traffic vs. buffer space + complexity

Continuous RQ: Go-back-N

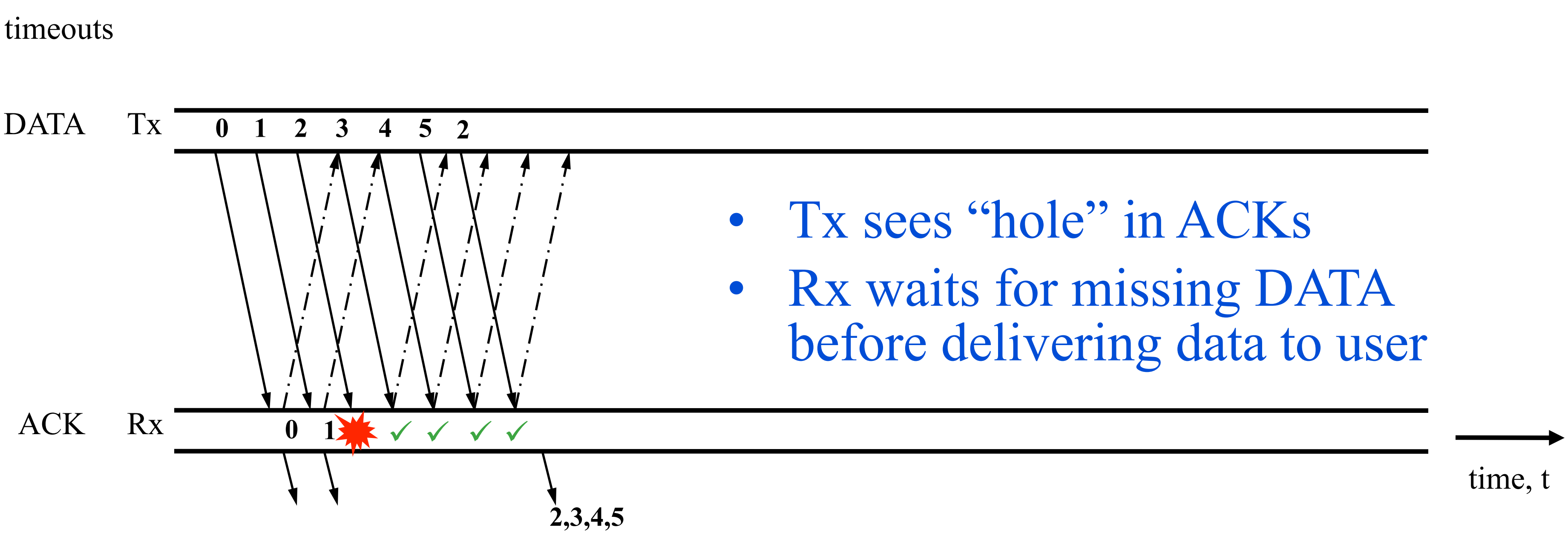
- If DATA PDU lost at Rx:
  - wait for re-transmission
  - ignore all following DATA

- Window size (w)
- SN modulus (m)
- $m > w$



Continuous RQ: Selective re-transmission

- Rx sends ACK for each DATA
- If DATA lost/corrupted,
  - send ACK for additional DATA
  - wait for re-transmission



- Tx sees “hole” in ACKs
- Rx waits for missing DATA before delivering data to user

Selective re-transmission: sequence numbers

Relationship between sequence number modulus, *m*, and maximum window size, *w*, must be **m >= 2w -1**

Scenario: selective retransmission with m=6, w=5 (valid for Go-back N)

Assume: 5 packets sent with sequence numbers – 0, 1, 2, 3, 4

window is full with 5 outstanding packets awaiting 5 Acks

5 Acks are sent: Rx is happy and is awaiting further packets with sequence numbers 5, 0, 1, 2, 3, ….

All 5 Acks are lost on way back to Tx.

Result: Tx times out on each ACK eventually resending each packet 0,1,2,3,4.

Rx receives resent packets 0,1,2,3,4.

Problem: Rx can see this 5 new packets with packet 5 lost and not as re-send of 5 lost packets – ***Need m >= 2w –1 to prevent this!!!!***

Full duplex service

- Duplex case:
  - protocol operations in both directions, simultaneously
- ACKs can be **piggybacked** on DATA:
  - separate SN fields for data and ACK in header
- Relies on two-way data flow:
  - many “pure” ACKs still possible (e.g. WWW access)
- Implementations:
  - large data frames delay ACKs, then give rise to timeouts

Sliding window

Assume: sequence number modulus m = 5
and window size w=3

1. Packet sequence: 0 1 2 3 4 0 1 2 3 4 0 1 2 3 4 0

Packets delivered

Packets in window:
sent but not ACKed

Packets queued at Tx

2. ACK PDU for 1 is received, Tx send the next packet (SN=4)

3. Packet sequence: 0 1 2 3 4 0 1 2 3 4 0 1 2 3 4 0

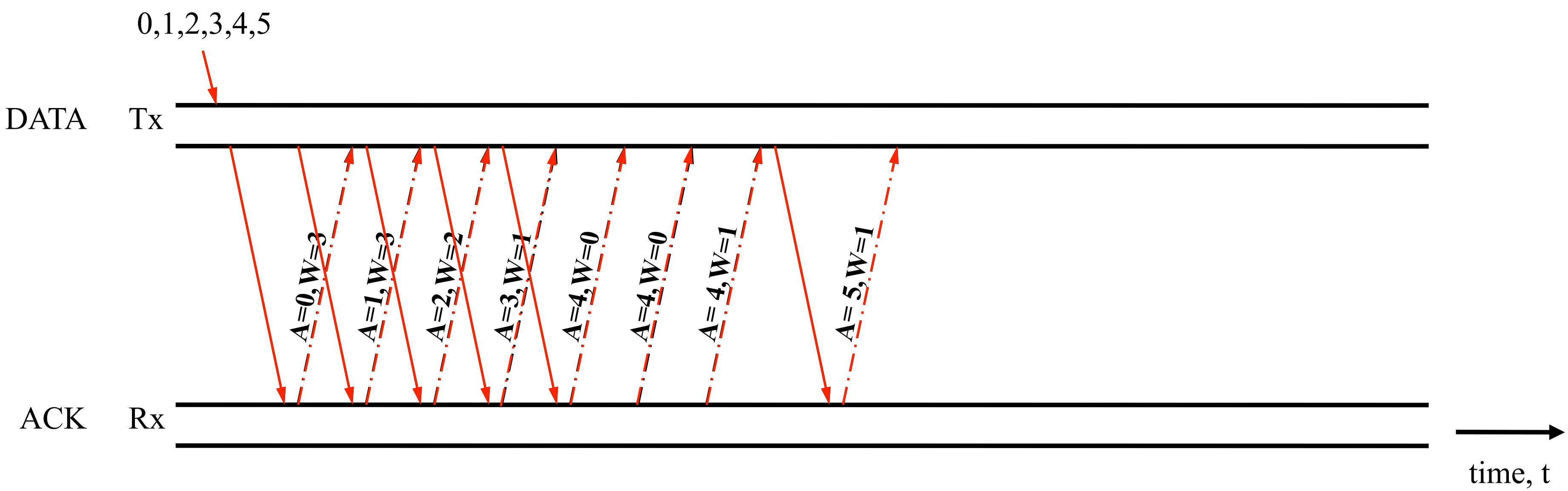
Sliding window – window moves along sequence of packets

Flow control

- Rx tells Tx how much data to send – **window**
- Rx can:
  - vary window size (increase or decrease)
  - reduce window to 1: Stop-and-Wait
  - other window sizes allow selective re-transmission
- ACK from Rx contains:
  - SN for data to be acknowledged (A)
  - window size for Tx to use (W)
- Tx can send (W - u) packets:
  - u is the number of packets outstanding

Flow control

- Rx controls TX rate with ACKs
- Often used in Transport layer



The End