Introduction to Computer Networks

Reliable Transmission

Outline

- **Introduction**
- **Stop-and-Wait protocol**
- **Sliding Window Protocol**
- **Issues with Sliding Window Protocol**

Reliable Transmission

- **Data transmissions over a communication link may have errors caused by signal interference.**

- **Usually, CRC is used to detect errors.**

- **Some error codes are strong enough to correct errors but the overhead is typically too high.**

- **Corrupt frames must be discarded.**

- **For reliable transmission, we must recover from these discarded frames.**

LAN
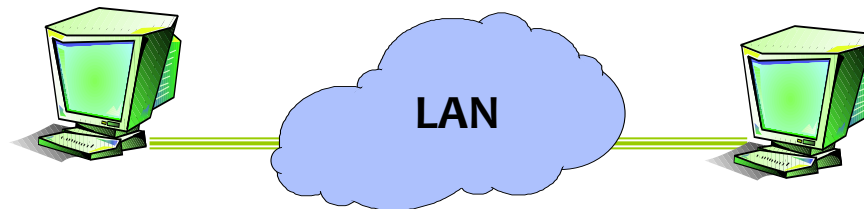
Reliable Transmission

- **We can do this by using a combination of two fundamental mechanisms**
  - **Acknowledgements**
  - **Timeouts**

- **An *acknowledgement* (ACK for short) is a small control frame that a protocol sends back to its peer saying that it has received the earlier frame.**
  - **A control frame is a frame with header only (no data).**

- **The receipt of an *acknowledgement* indicates to the sender that its frame was successfully delivered.**

Reliable Transmission

- If the sender does not receive an *acknowledgment* after a reasonable amount of time, then it retransmits the original frame.

- The action of waiting a reasonable amount of time is called a *timeout*.

- The general strategy of using *acknowledgements and timeouts* to implement reliable delivery is sometimes called **Automatic Repeat reQuest (ARQ)**.

Outline

- **Introduction**
- **Stop-and-Wait protocol**
- **Sliding Window Protocol**
- **Issues with Sliding Window Protocol**

Stop and Wait Protocol

■ **Idea of stop-and-wait protocol is straightforward**

- **After transmitting one frame, the sender waits for an acknowledgement before transmitting the next frame.**

- **If the acknowledgement does not arrive after a certain period of time, the sender times out and retransmits the original frame**
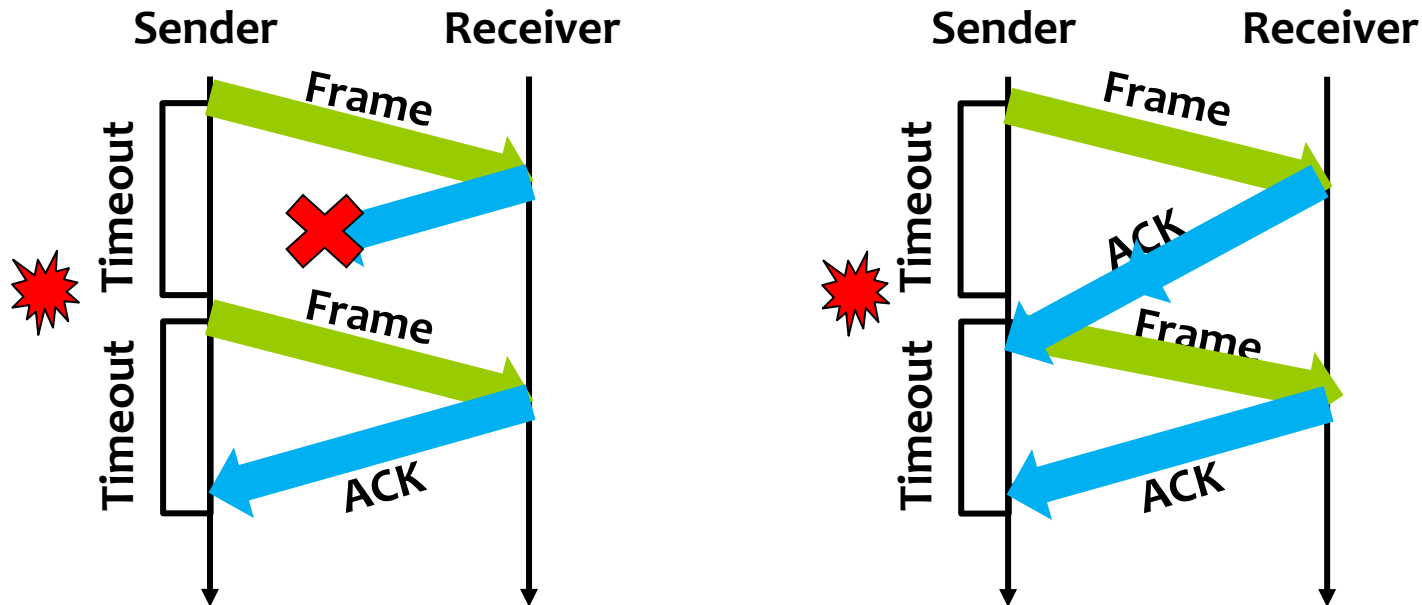
Stop and Wait Protocol
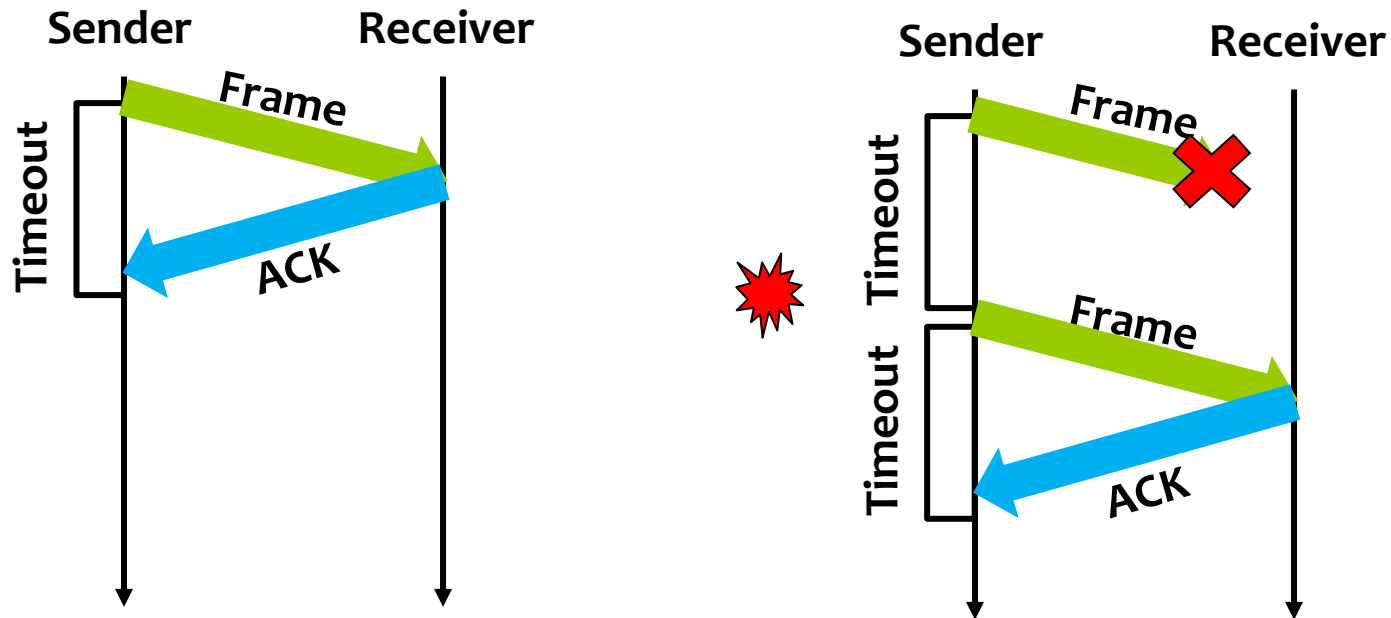
Four different scenarios for the stop-and-wait algorithm.
(a) The ACK is received before the timer expires;
(b) The original frame is lost;

Stop and Wait Protocol

Four different scenarios for the stop-and-wait algorithm.
 (c) The ACK is lost;
 (d) The timeout fires too soon (or the ACK is delayed)

Stop and Wait Protocol

- **If the acknowledgment is lost or delayed in arriving**
  - The sender times out and retransmits the original frame.
  - As a result, duplicate copies of frames will be delivered
- **How to solve this ?**
  - Use 1 bit sequence number (0 or 1)
  - When the sender retransmits frame 0, the receiver can determine that this is a second copy of frame 0 rather than the first copy of frame 1 and therefore can ignore it (the receiver still acknowledges it, in case the first acknowledgement was lost)

Stop and Wait Protocol

Sender   **Receiver**

Frame *0*

ACK *0*

Frame *1*

ACK *1*

Frame *0*

ACK *0*
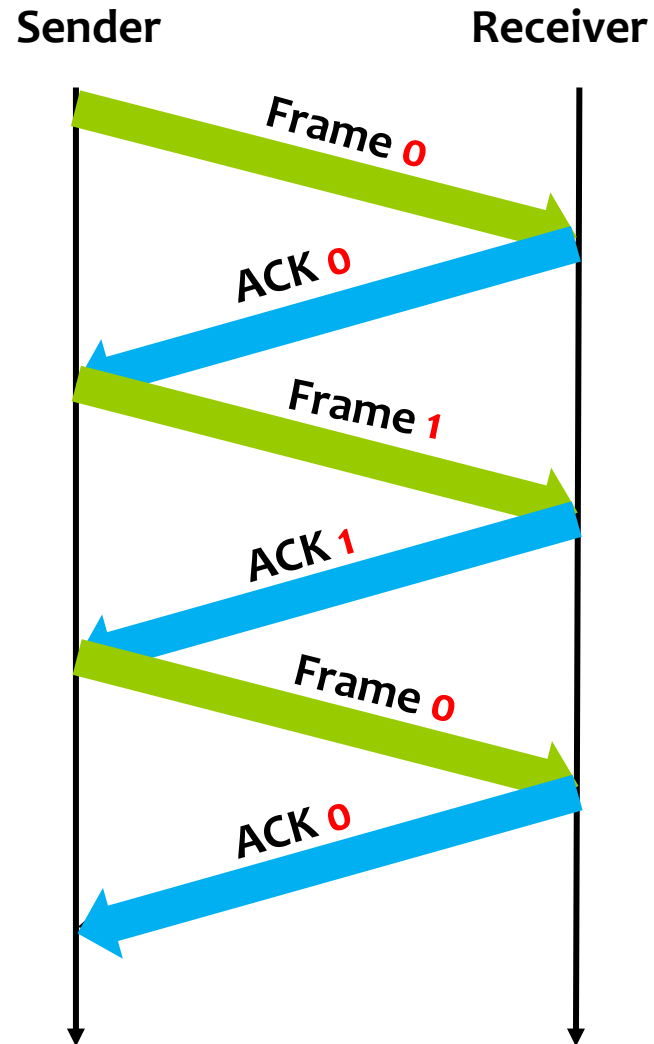
**Timeline for stop-and-wait with 1-bit sequence number**

Stop and Wait Protocol

- **The sender has only one outstanding frame on the link at a time -- Far below the link's capacity**

- **For example, consider a 2 Mbps link with a 40 ms RTT**
  - **The link has a delay bandwidth product of 80 Kb or 10 KB**
  - **Since the sender can send only one frame per RTT and assuming a frame size of 1 KB**
  - **Maximum Sending rate**
    - **Bits per frame/Time per frame = 8kb ÷ 40ms = 200 Kbps**

      Or about 1/10 of the link's capacity (2Mbps)
  - **To fully use the link, the sender should transmit up to ten frames before having to wait for an ACK.**

Outline

- **Introduction**

- **Stop-and-Wait protocol**

- **Sliding Window Protocol**

- **Issues with Sliding Window Protocol**

Sliding Window Protocol

Sender   Receiver
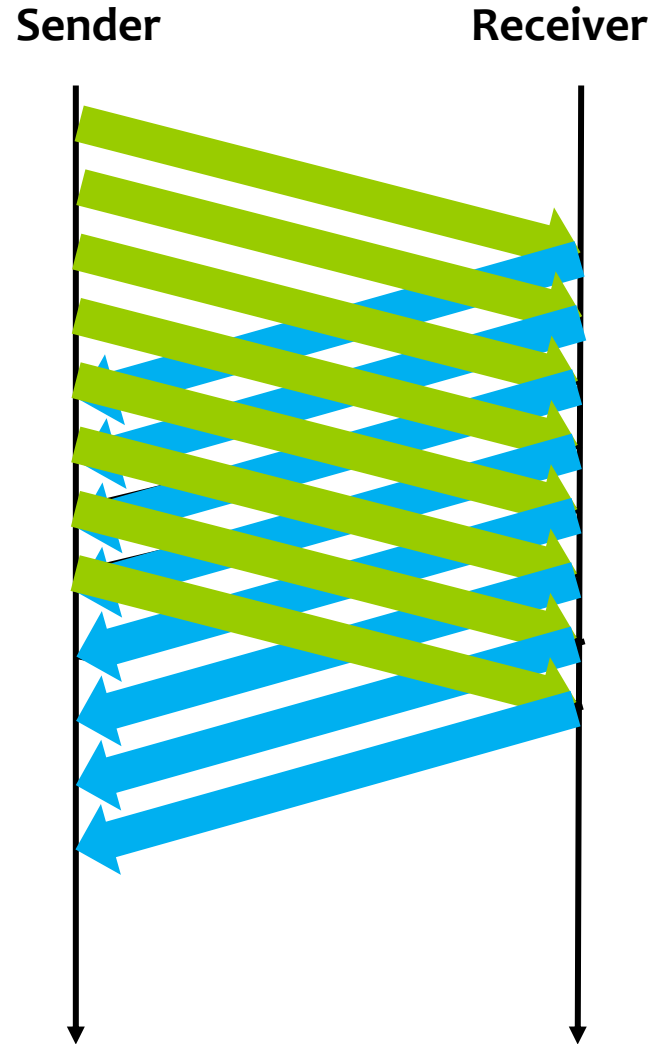
Timeline for Sliding Window Protocol
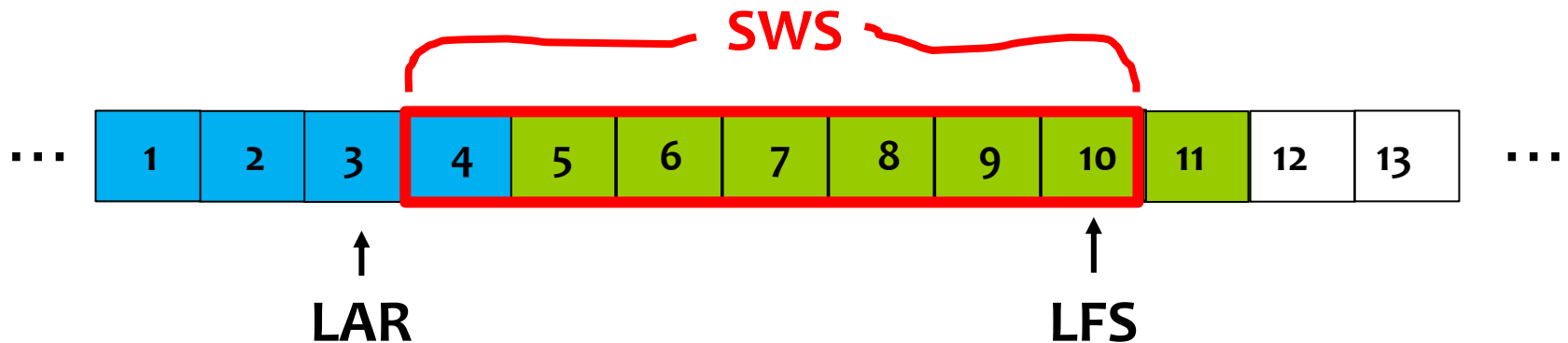
Sliding Window Protocol

- **Sender assigns a sequence number denoted as SeqNum to each frame.**

  - **Assume it can grow infinitely large**

- **Sender maintains three variables**

  - **Sending Window Size (SWS)**

    - Upper bound on the number of outstanding (unacknowledged) frames that the sender can transmit

  - **Last Acknowledgement Received (LAR)**

    - Sequence number of the last acknowledgement received

  - **Last Frame Sent (LFS)**

    - Sequence number of the last frame sent

Sliding Window Protocol

■ **Sender also maintains the following invariant**

LFS – LAR ≤ SWS

Sliding Window on Sender

Sliding Window Protocol

- **When an acknowledgement arrives**
  - the sender **moves LAR to right**, thereby allowing the sender to transmit another frame

- **Also the sender associates a timer with each frame it transmits**
  - It retransmits the frame if the timer expires before the ACK is received

- **Note that the sender needs to buffer up to SWS frames for retransmissions, if necessary**

Sliding Window Protocol

■ **When the sender is allowed to slide its Window ?**

- ● **When the acknowledgement of LAR+1 is received**
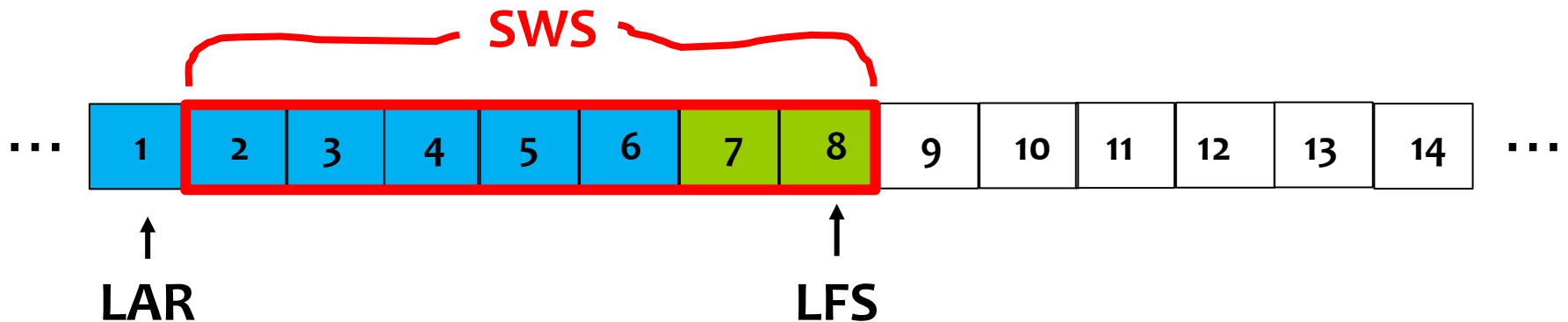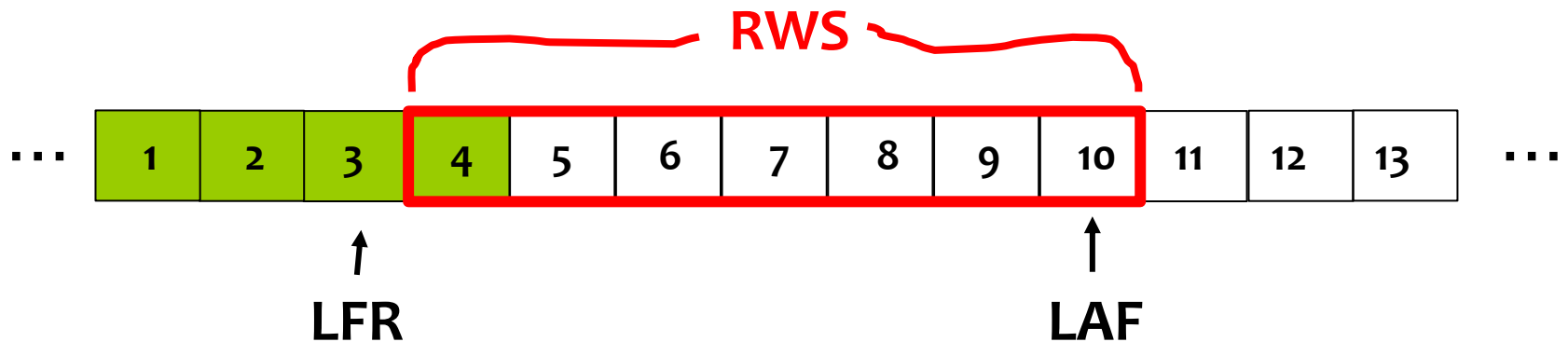

Sliding Window on Sender

Sliding Window Protocol

■ **Receiver maintains three variables**

- **Receiving Window Size (RWS)**
  - ‣ Upper bound on the number of out-of-order frames that the receiver is willing to accept

- **Largest Acceptable Frame (LAF)**
  - ‣ Sequence number of the largest acceptable frame

- **Last Frame Received (LFR)**
  - ‣ Sequence number of the last frame received

Sliding Window Protocol

■ **Receiver also maintains the following invariant**
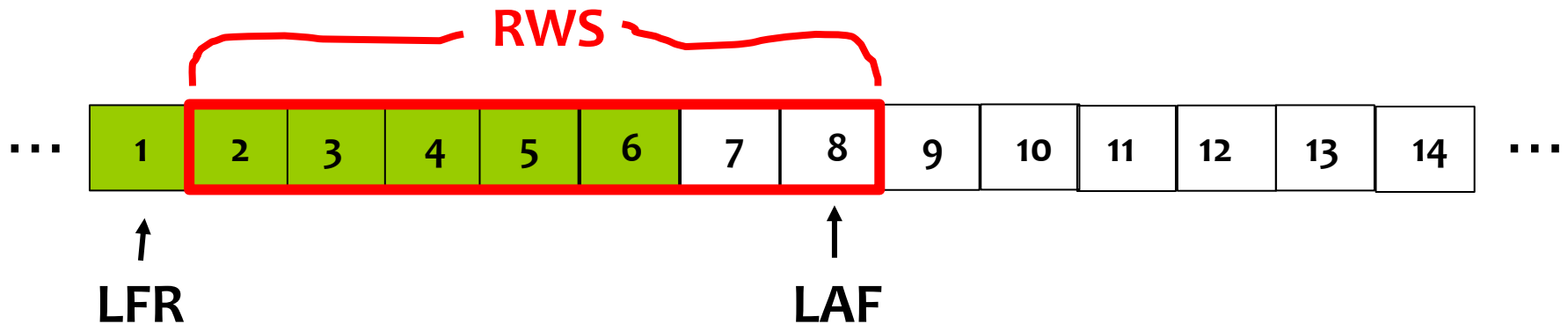
$$\text{LAF} - \text{LFR} \leq \text{RWS}$$

Sliding Window on Receiver

Sliding Window Protocol

■ **When a frame with sequence number** SeqNum **arrives, what does the receiver do?**

- **If** SeqNum ≤ LFR **or** SeqNum > LAF
  - ▸ Discard it (the frame is outside the receiver window)

- **If** LFR < SeqNum ≤ LAF
  - ▸ Accept it

Sliding Window Protocol

■ **When the receiver is allowed to slide its Window ?**

● **When the frame of LFR+1 is received**

**Sliding Window on Receiver**

Sliding Window Protocol

- **For example, suppose LFR = 1 and RWS = 7**

  → LAF = 8

- **If frames 4,6,3 and 5 arrive sequentially, they are out of order, but will still be buffered because they are within the receiver window**

- **But no ACK will be sent since frame 2 is yet to arrive**

- **Finally, frame 2 arrives (retransmitted or delayed)**

- **Now Receiver Acknowledges Frame 6**

  and bumps LFR to 6

  and LAF to 13 (Window sliding)

Sliding Window Protocol

- **Cumulative Acknowledgement**

  - Let **SeqNumToAck** denote the largest sequence number not yet acknowledged, such that all frames with sequence number **less than SeqNumToAck have been received**

  - The receiver acknowledges the receipt of **SeqNumToAck** even if high-numbered packets have been received

    - This acknowledgement is said to be **cumulative.**

  - The receiver then sets

    - **LFR = SeqNumToAck -1 and**

    - **LAF = LFR + RWS (Window sliding)**

Sliding Window Protocol
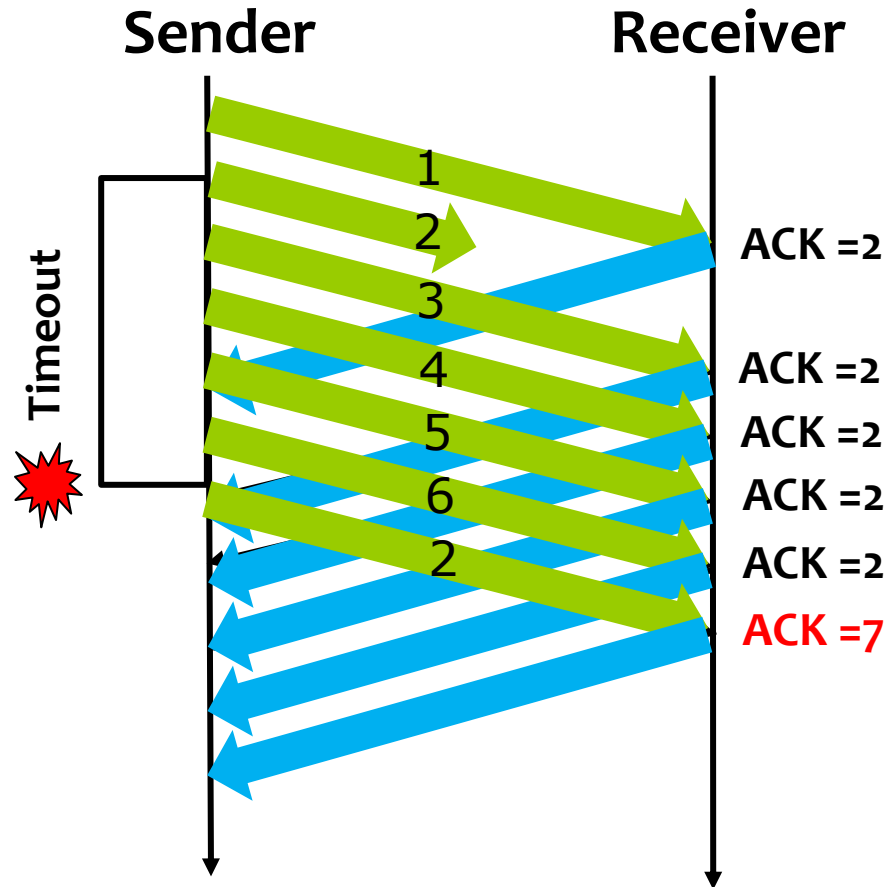
Cumulative Acknowledgement example (SeqNumToAck = 2)

Outline

- **Introduction**

- **Stop-and-Wait protocol**

- **Sliding Window Protocol**

- **Issues with Sliding Window Protocol**

Issues with Sliding Window Protocol

- **When timeout occurs, the amount of data in transit decreases**
  - Since the sender is unable to advance its window

- **When the packet loss occurs, this scheme is no longer keeping the pipe full**
  - The longer it takes to notice that a packet loss has occurred, the more severe the problem becomes

- **How to improve this**
  - **Negative Acknowledgement (NAK)**
  - **Additional Acknowledgement**
  - **Selective Acknowledgement**

Issues with Sliding Window Protocol

- **Negative Acknowledgement (NAK)**
  - Receiver sends NAK for frame 2 when frame 3 arrive (in the cumulative acknowledgement example)
    - However this is unnecessary since sender's timeout mechanism will be sufficient to catch the situation

- **Additional Acknowledgement**
  - Receiver sends additional ACK for frame 2 when frame 3 arrives
    - Sender uses duplicate ACK as a clue for frame loss

- **Selective Acknowledgement**
  - Receiver will acknowledge exactly those frames it has received, rather than the highest number frames
    - Receiver will acknowledge frames 3,4,5, and 6
    - Sender knows frame 2 is lost
    - Sender can keep the pipe full (additional complexity)

Issues with Sliding Window Protocol

How to select the window size ?

- **SWS is easy to calculate**

  - **Delay × Bandwidth**

- **RWS is more flexible**

  - **Two common settings**

    - » **RWS = 1**

      No buffer at the receiver for frames that arrive out of order

    - » **RWS = SWS**

      The receiver can buffer frames that the sender transmits

- **Does it make any sense to keep RWS > SWS ?**

Issues with Sliding Window Protocol

■ **Finite Sequence Number**

- ● **Frame sequence number is specified in the header field**
  - ▸ **Finite size**
    - » **3 bits: eight possible sequence number: 0, 1, 2, 3, 4, 5, 6, 7**
  - ▸ **It is necessary to wrap around**

Issues with Sliding Window Protocol

- **How to distinguish between different frames of the same sequence number?**
  - **Number of possible sequence number must be larger than the number of outstanding frames allowed**
    - Stop and Wait: One outstanding frame
      - » 2 distinct sequence number (0 and 1)
    - Let **MaxSeqNum** be the number of available sequence numbers
    - SWS + 1 ≤ MaxSeqNum
      - Is this sufficient?
      - Depends on RWS
      - If RWS = 1, then sufficient
      - If RWS = SWS, then not good enough

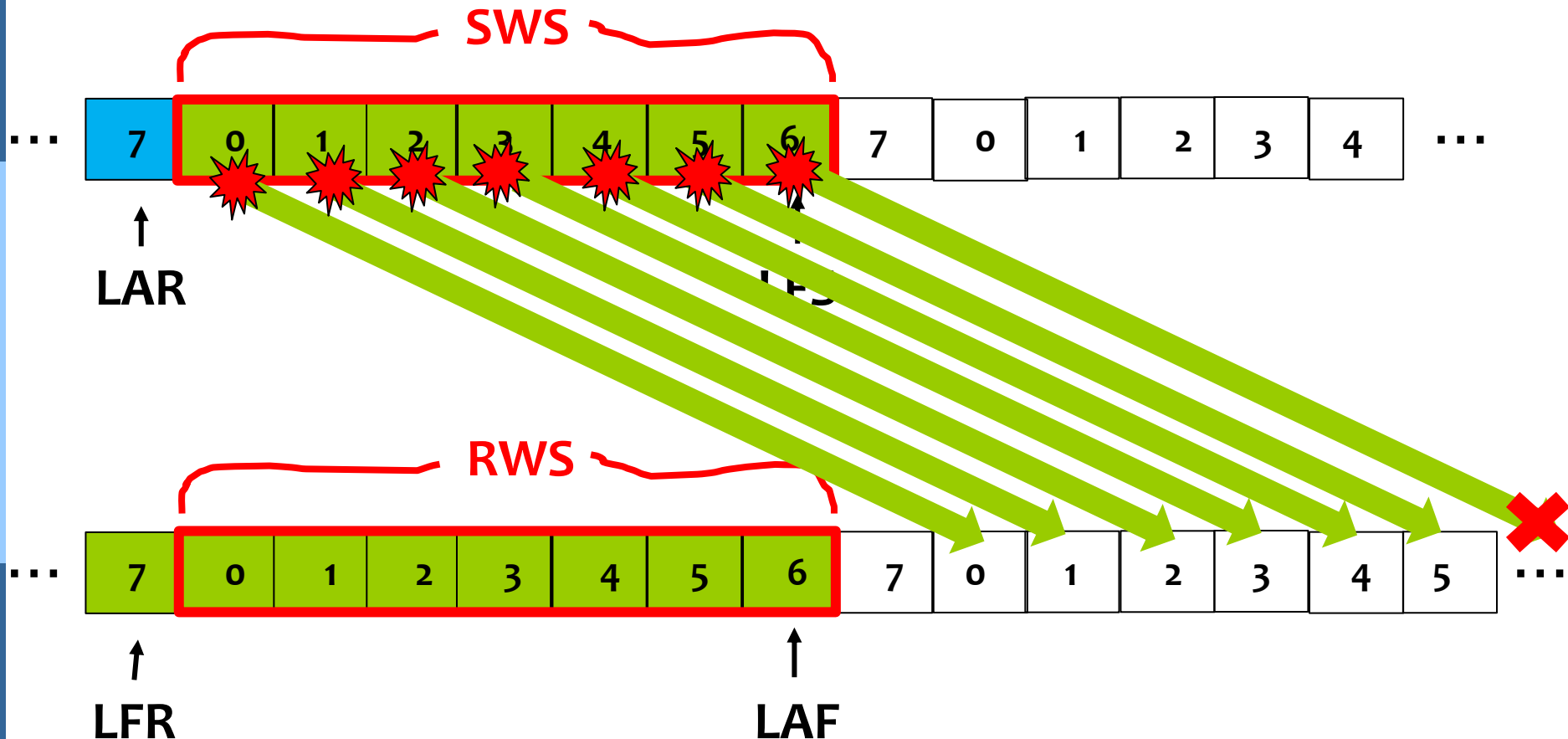# Issues with Sliding Window Protocol

For example, we have eight sequence numbers

0, 1, 2, 3, 4, 5, 6, 7

RWS = SWS = 7 = 8-1

1. Sender sends 0, 1, …, 6

2. Receiver receives 0, 1, … ,6

3. Receiver acknowledges 0, 1, …, 6

   but **ACK (0, 1, …, 6) are lost**

4. Sender timeouts and retransmits **0, 1, …,5**, 6

5. Receiver is expecting 7, **0, …., 5**

➡ frames 0-5 will be accepted (duplicated !!)

   frame 6 will be discarded (correct)

Issues with Sliding Window Protocol

Problem when SWS + 1 ≤ MaxSeqNum and SWS = RWS

MaxSeqNum = 8, SWS = RWS = 7

Issues with Sliding Window Protocol

To avoid this,

If RWS = SWS

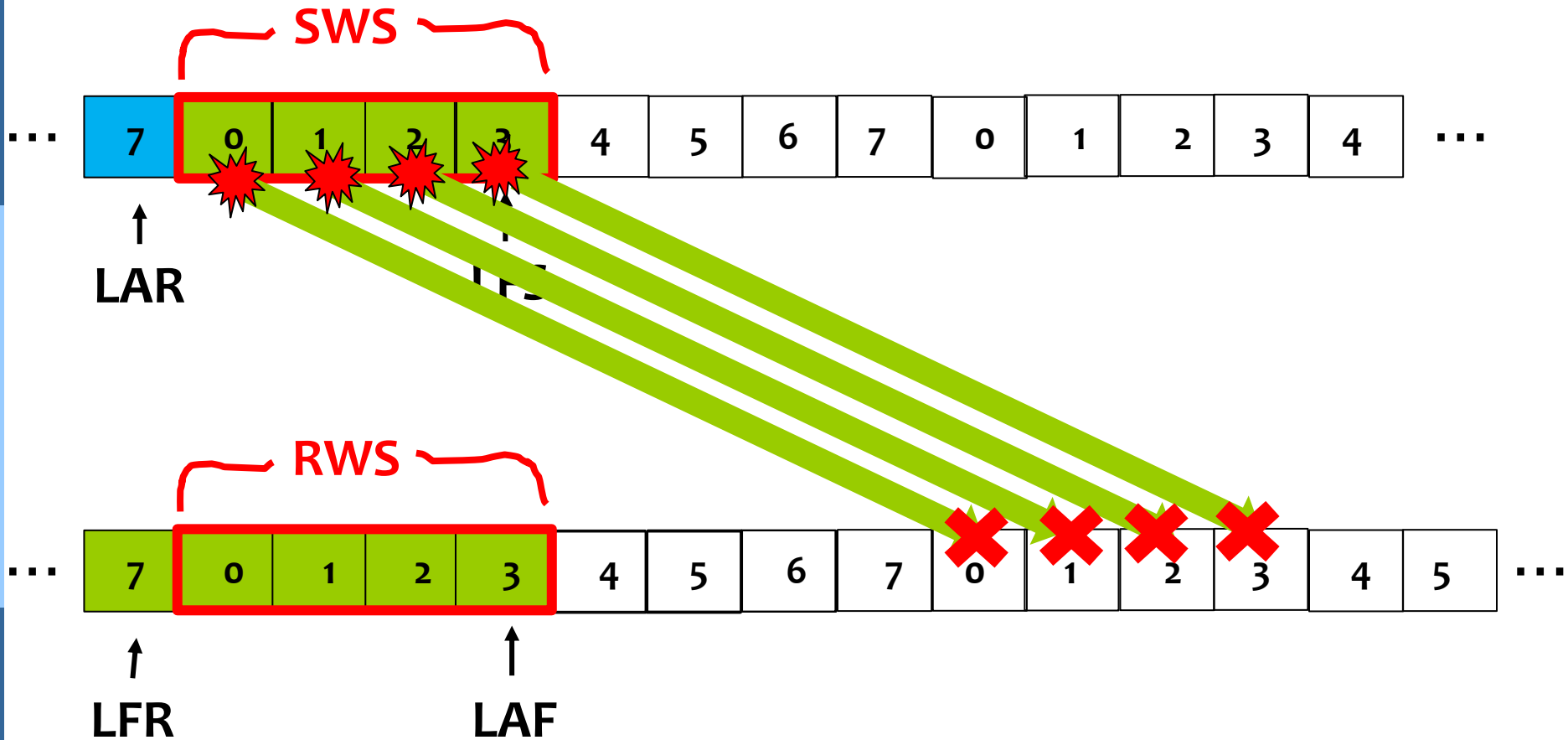**SWS < (MaxSeqNum + 1)/2**

Issues with Sliding Window Protocol

For example, we have eight sequence numbers

0, 1, 2, 3, 4, 5, 6, 7

RWS = SWS = 4 < (8+1)/2 = 4.5

1. Sender sends 0, 1, 2, 3

2. Receiver receives 0, 1, 2, 3

3. Receiver acknowledges 0, 1, 2, 3

   but ACK (0, 1, 2, 3) are lost

4. Sender timeouts and retransmits 0, 1, 2, 3

5. Receiver is expecting 4,5,6,7

➔ all frames 0-3 will be discarded (correct now !!)

Issues with Sliding Window Protocol

Example when **SWS < (MaxSeqNum+1)/2**

MaxSeqNum = 8, SWS = RWS = 4 < (8+1)/2

Issues with Sliding Window Protocol

- **Sliding Window Protocol provides three features**
  - **Reliable Transmission**
  - **Preserve the order**
    - ▸ Each frame has a sequence number
    - ▸ Out of order frames will be buffered
  - **Flow control**
    - ▸ Receiver is able to throttle the sender by setting the value of RWS
   - ▸ Keeps the sender from overrunning the receiver

Summary

- **Two fundamental mechanisms are used to provide reliable transmission over a communication link**
  - **Acknowledgements**
  - **Timeouts**
- **Stop-and-Wait Protocol is a reliable protocol, but the performance is not good enough**
  - **Only one outstanding frame**
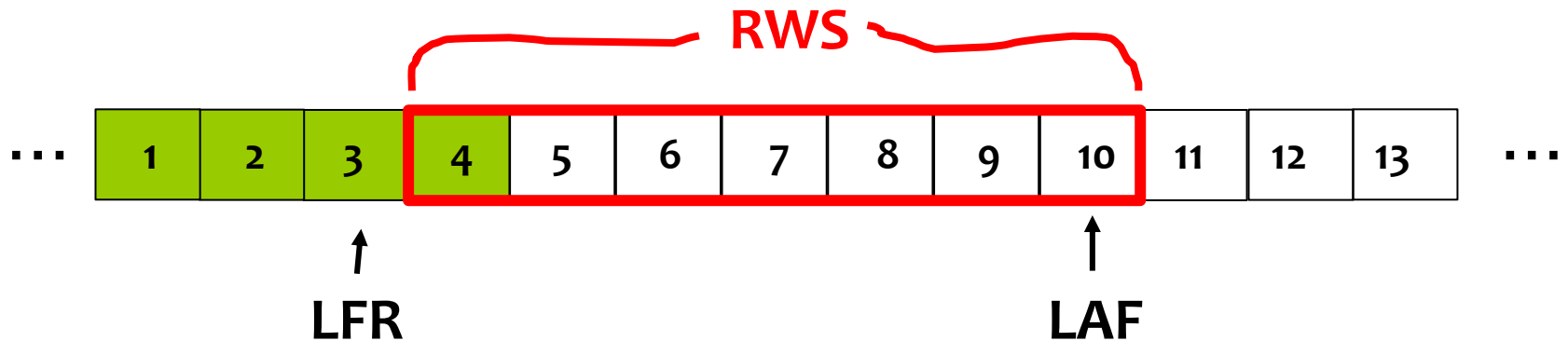 - **Receiver may receive duplicated frames**

Summary

- **Sliding Window Protocol is a reliable and good performance protocol**
  - **Sequence number is added for each frame**
  - **Multiple outstanding frames (keeping pipe full)**
  - **Sender**
    - **Sending Window Size (SWS) = delay x bandwidth**
    - **Last Acknowledgement Received** (**LAR**)
    - **Last Frame Sent** (**LFS**)
  - **Receiver**
    - **Receiving Window Size (RWS) = 1 or SWS**
    - **Largest Acceptable Frame (LAF)**
    - **Last Frame Received (LFR)**

Summary

- **Sender maintains the invariant:** $LFS - LAR \leq SWS$
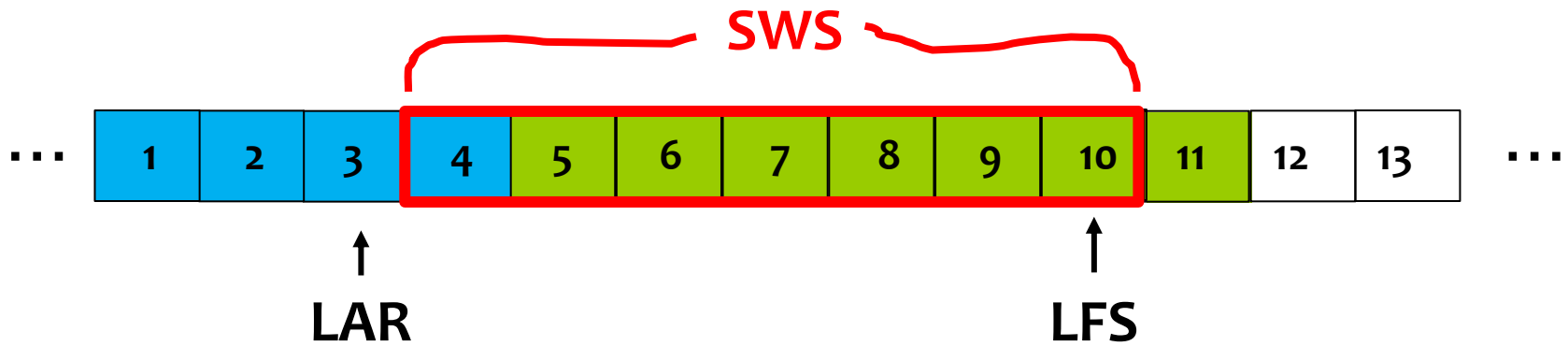
- **Receiver maintains the invariant:** $LAF - LFR \leq RWS$

Summary

- **The values of SWS and RWS may be different.**

- **If RWS = SWS, it is better SWS < (MaxSeqNum + 1)/2 otherwise, the receiver may still receive duplicated frames.**

- **The Acknowledgement mechanism is an option**
  - NAK
  - Cumulative acknowledgement

- **Sliding Window Protocol provides three features**
  - Reliable Transmission
  - Preserve the order
  - Flow control (receiver determines the RWS)