Data Link Control

Flow and Error Control

- Flow Control
  - Flow control refers to a set of procedures used to **restrict the amount of data that the sender can send** before waiting for acknowledgment from the receiver
  - If the channel is **error-free**:
    - Stop-and-Wait flow control
    - Sliding-Window flow control
- Error Control
  - Refers to procedures to **detect and correct** errors
  - Includes the following actions:
    - Error detection
    - Positive Acknowledgement (**ACK**): if the frame arrived with no errors
    - Negative Acknowledgement (**NAK**): if the frame arrived with errors
    - Retransmissions after **timeou**t: Frame is retransmitted after certain amount of time if no acknowledgement was received
  - These actions are called A**utomatic Repeat Request (ARQ)**

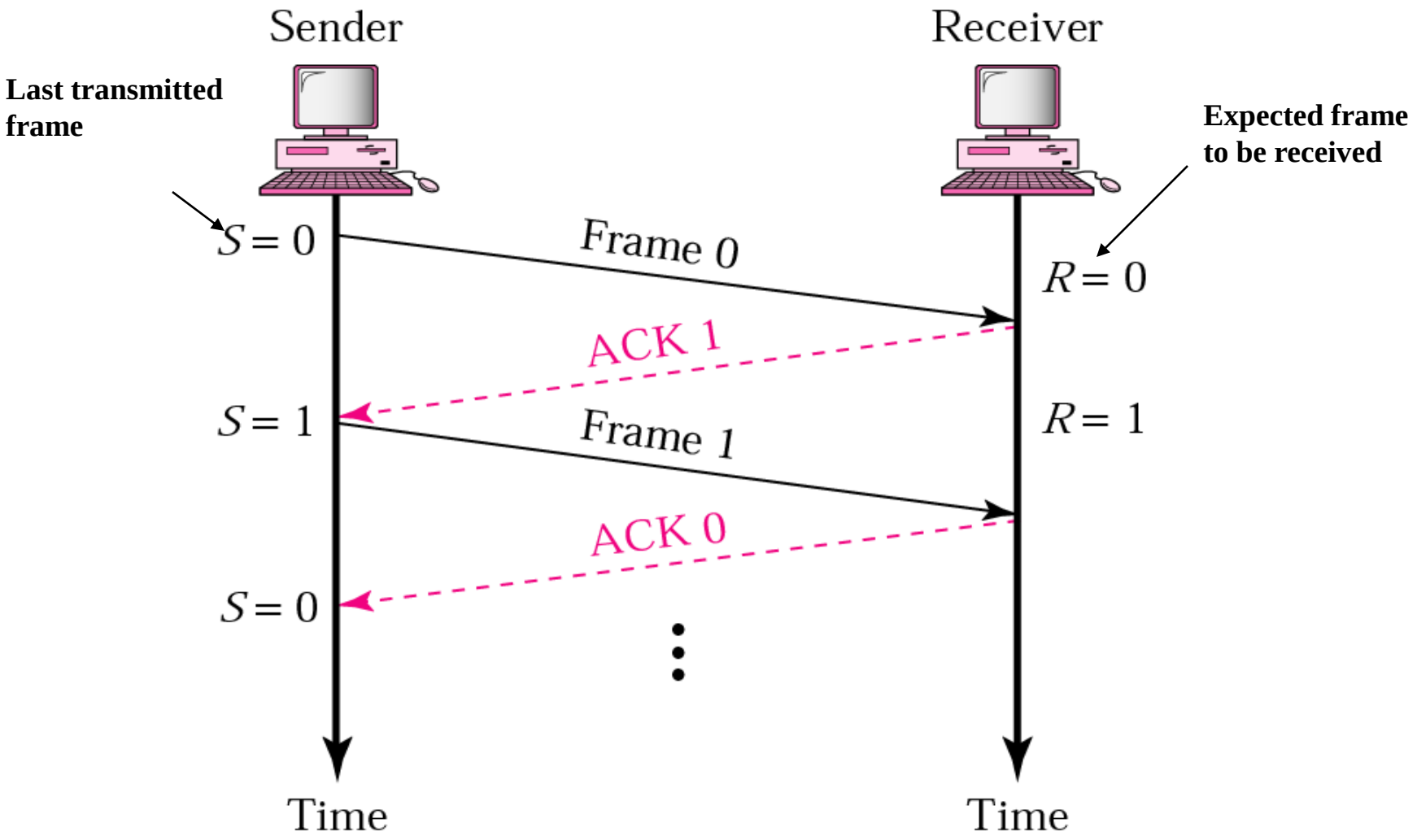
11.2 Flow and Error Control

- Usually Error and flow control protocols are combined together to provide reliable data transfer service called data link control
    - Stop-and-Wait ARQ
    - Go-Back-N ARQ
    - Selective repeat ARQ
- ARQ provide **reliable data transfer** service over **unreliable networks**
- ARQ ensure that transmitted **data** is delivered accurately to the destination despite errors that occur during transmission and satisfies the following:
    - **Error free**
    - **Without duplicates**
    - **Same order** in which they are transmitted
    - **No loss**

Stop-and-Wait ARQ

- **Simplest** flow and error control protocol
- Data frames and **ACK** frames are numbered alternately (0,1)
- **When receiver sends ACK1: it** acknowledges data frame 0 and is expecting data frame 1, **ACK0** acknowledges data frame 1 and is expecting data frame 0
- Sequence Numbers are incremented **modulo 2**. 1+1 = 0 , and 0+1=1
- Receiver has a counter **(R)** which hold **the number of the expected frame to be received**. **R** is incremented by 1 modulo 2 when the **expected frame is received**.
- The sender keeps a counter **(S)** which **holds the number of the last transmitted frame. S** is incremented by 1 modulo 2 when the **acknowledgement of the last transmitted frame is received** and the **next frame is transmitted**

Normal operation



11.5 Stop-and-Wait ARQ – Cont'd

- If the **receiver** detects an error in the frame it **discards it**
- If the **receiver** receives an out-of-order frame (frame 0 instead of frame 1 or vice versa), it knows that the expected frame is lost or damaged and **discards** the out-of-order frame and **resend** the previous **ACK**
- If the **sender** receives an **ACK** with a **different** number than the current value of **S+1**, it **discards it**
- The **sending device** keeps a **copy** of the **last frame** transmitted until it receives the right acknowledgment (**ACK**) for the frame
- The **sender starts** a **timer** when it sends a frame. If an **ACK** is not received within the allocated time, the sender assumes that the frame was lost or damaged and **resends** it

Stop-and-Wait ARQ, lost frame

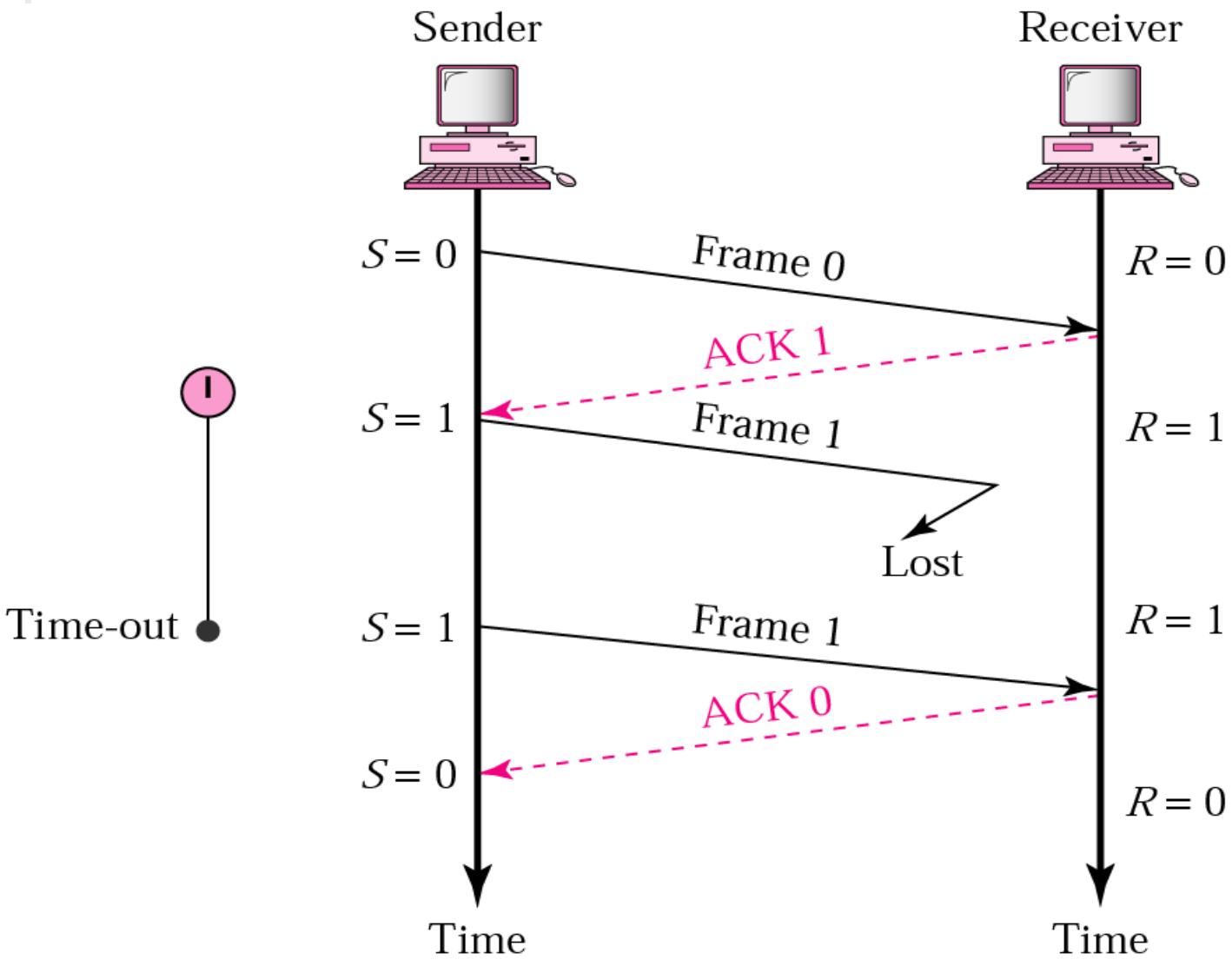
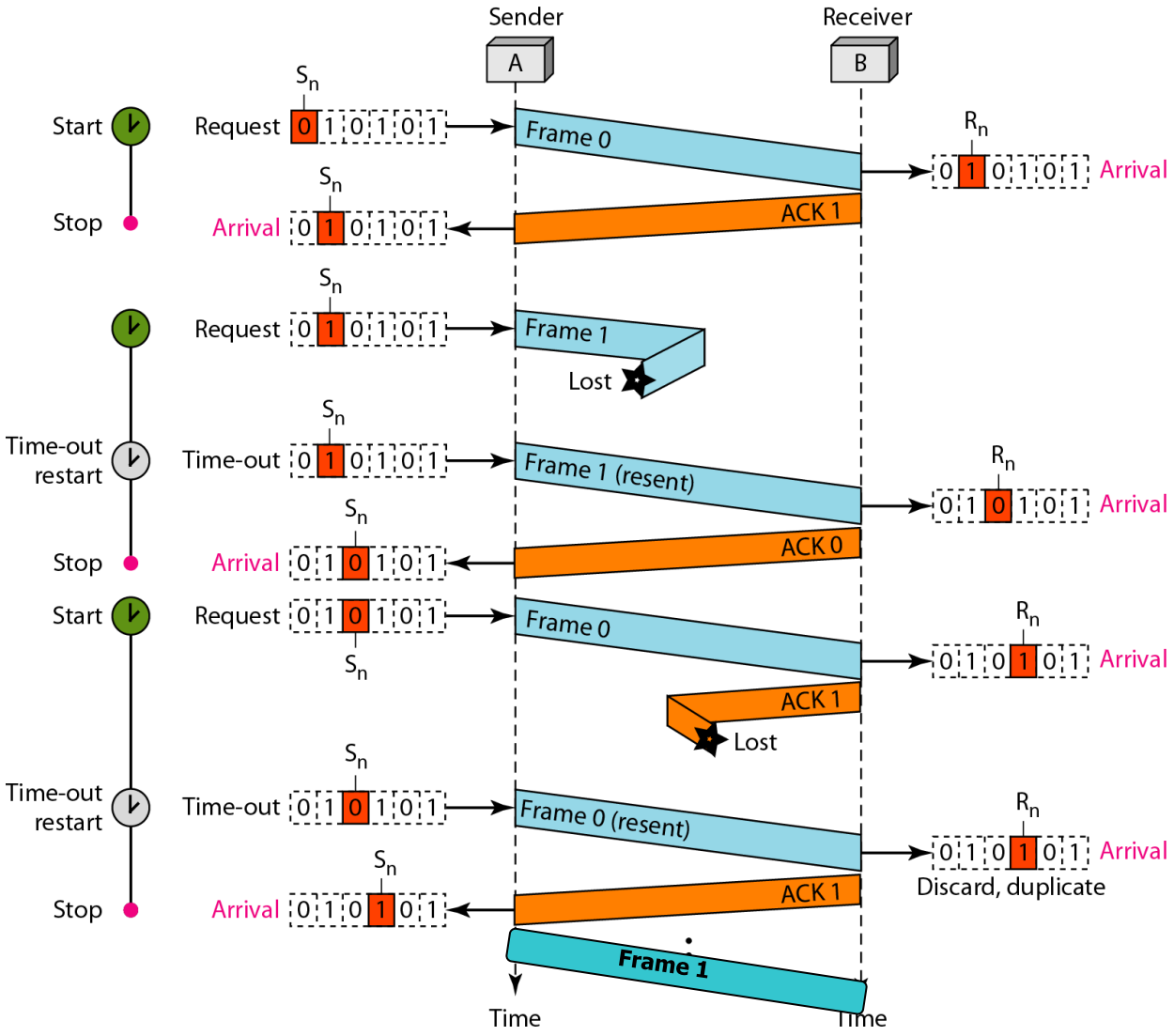


Figure 11.11 *Flow diagram for Example 11.3*



Note:

In Stop-and-Wait ARQ, numbering frames prevents the retaining of duplicate frames.

Note:

Numbered acknowledgments are needed if an acknowledgment for frame is delayed and the next frame is lost.

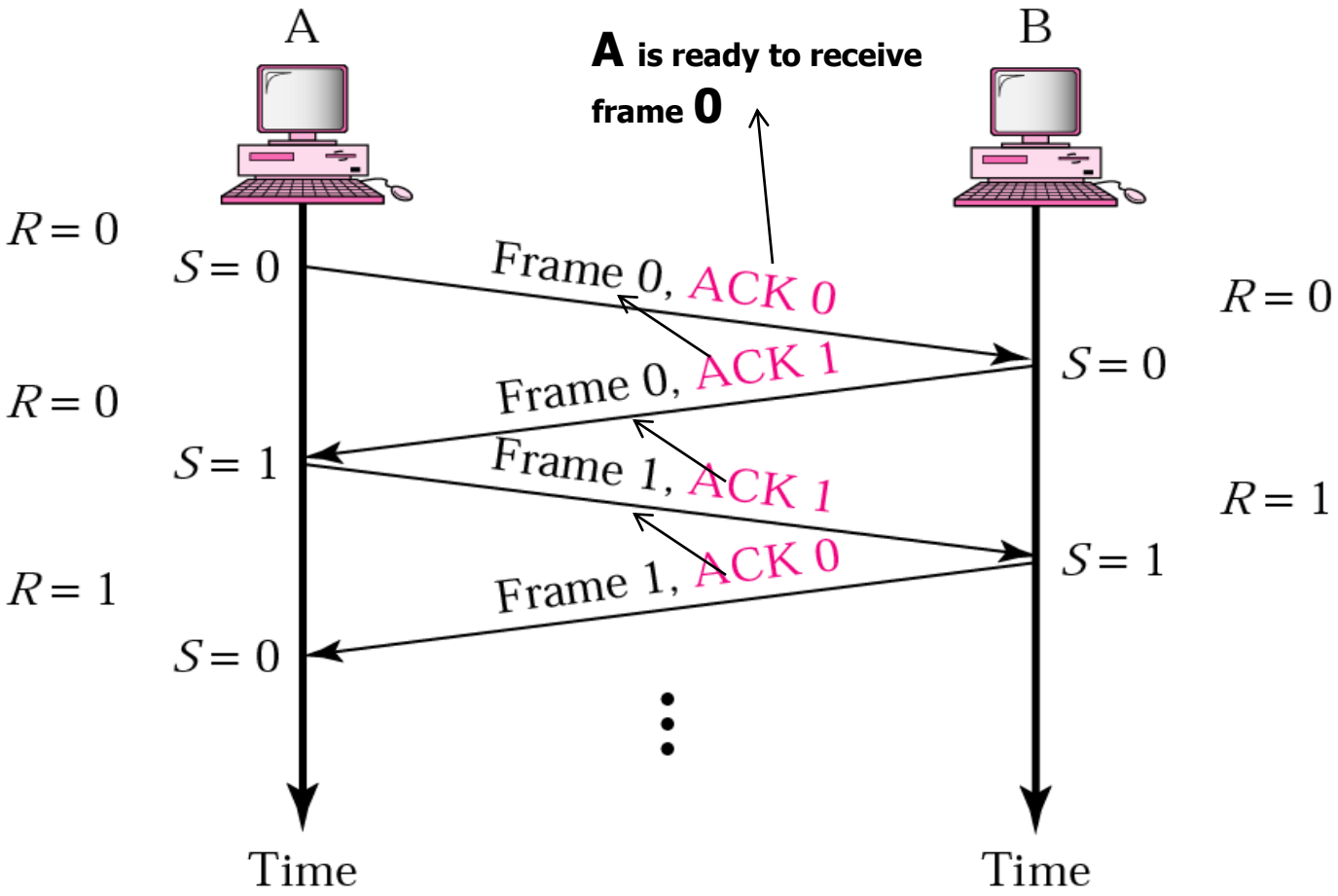
Performance

- At any time, there is **one** frame, the one sent and waiting to be acknowledged.
- Works well when **propagation time** is **much less** than **transmission time**
  - Once the sender finishes from transmitting the frame it will arrive in very short time to the receiver because propagation time is very small. The acknowledgement frame has very small transmission time so it will arrive quickly to the sender. As a result, the sender will not **stay idle for long period of time.**
- To improve efficiency, multiple frames should be in transition while sender is waiting for ACK. This is implemented in sliding window algorithms

Bidirectional Transmission

- The previous protocol is unidirectional (data is transmitted in one direction)
- For bidirectional both nodes can send data
- To save **bandwidth** data frame *to be sent* and control information (ACK) of the *received frame* are **combined into one frame** (**Piggybacking**)
- If a node receive a frame and does not have data to be sent then an ACK frame is **sent alone**

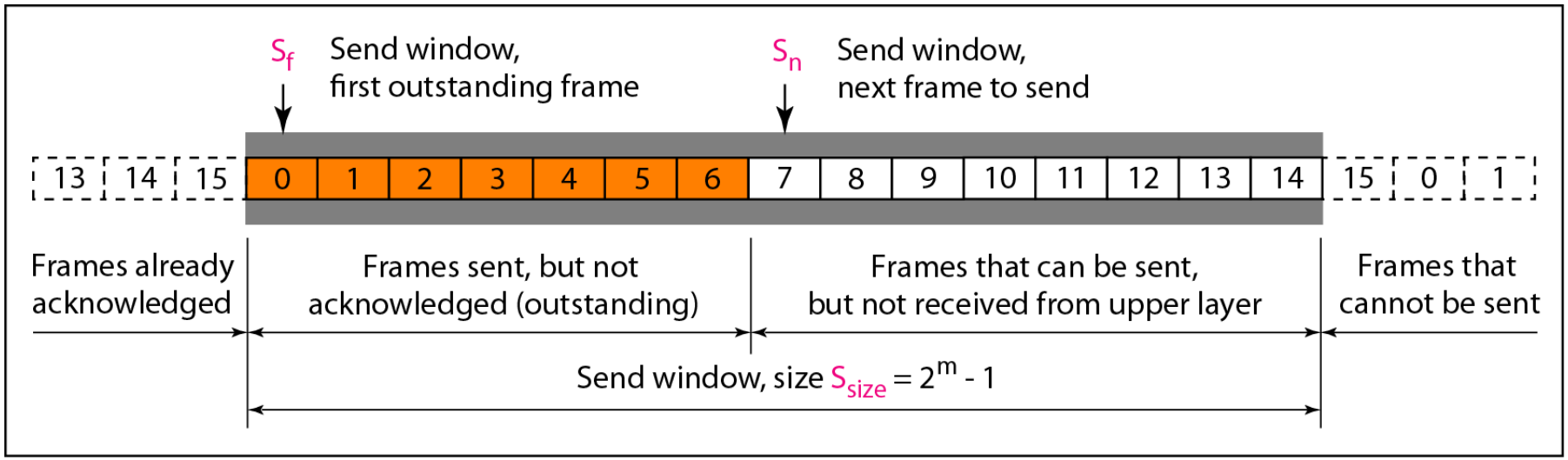
Piggybacking



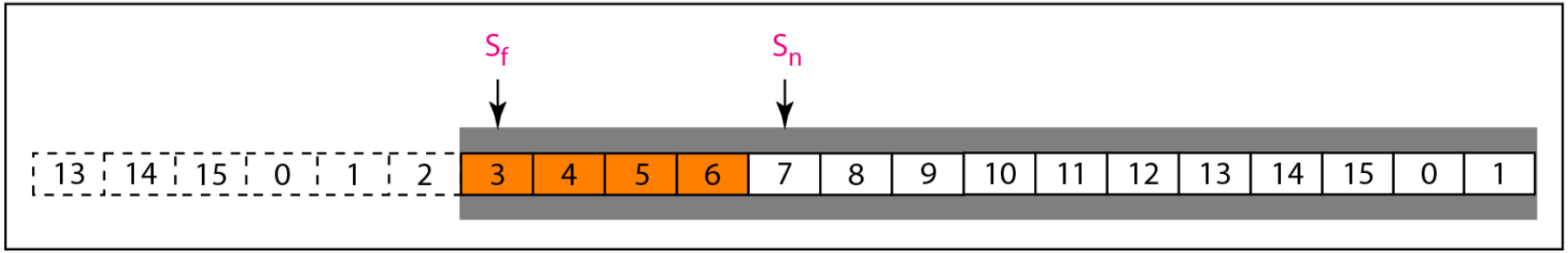
11.5 Go-Back-N ARQ

- **Setup – Sender sliding window**
- Frames are **numbered sequentially** using **m-bit** in the **frame heade**r
  - For m bits frame sequence numbers $0 – (2^m – 1)$, repeated
  - m=3 ➔ (0,1,2,3,4,5,6,7,0, 1,2,3,4,5,6,7,…)
- Sender can send multiple frames while waiting for ACK, but **total number of unacknowledged frames** should **not** exceed **$2^m – 1$** which is called **window**
- **Window** is an **imaginary placeholder** that covers the frames sequence numbers which can be in **transit**
- The frames to **the left** of the window are already **acknowledged** and to **the right** can not be sent until window slides over them
- Frames **inside** the window are **outstanding** (have been sent but not yet acknowledged OR **will be sent soon once** received from upper-layer)
- **Sender window will slide to frame number (i) when (ACK i) is received.**

Figure 11.12 *Send window for Go-Back-N ARQ*

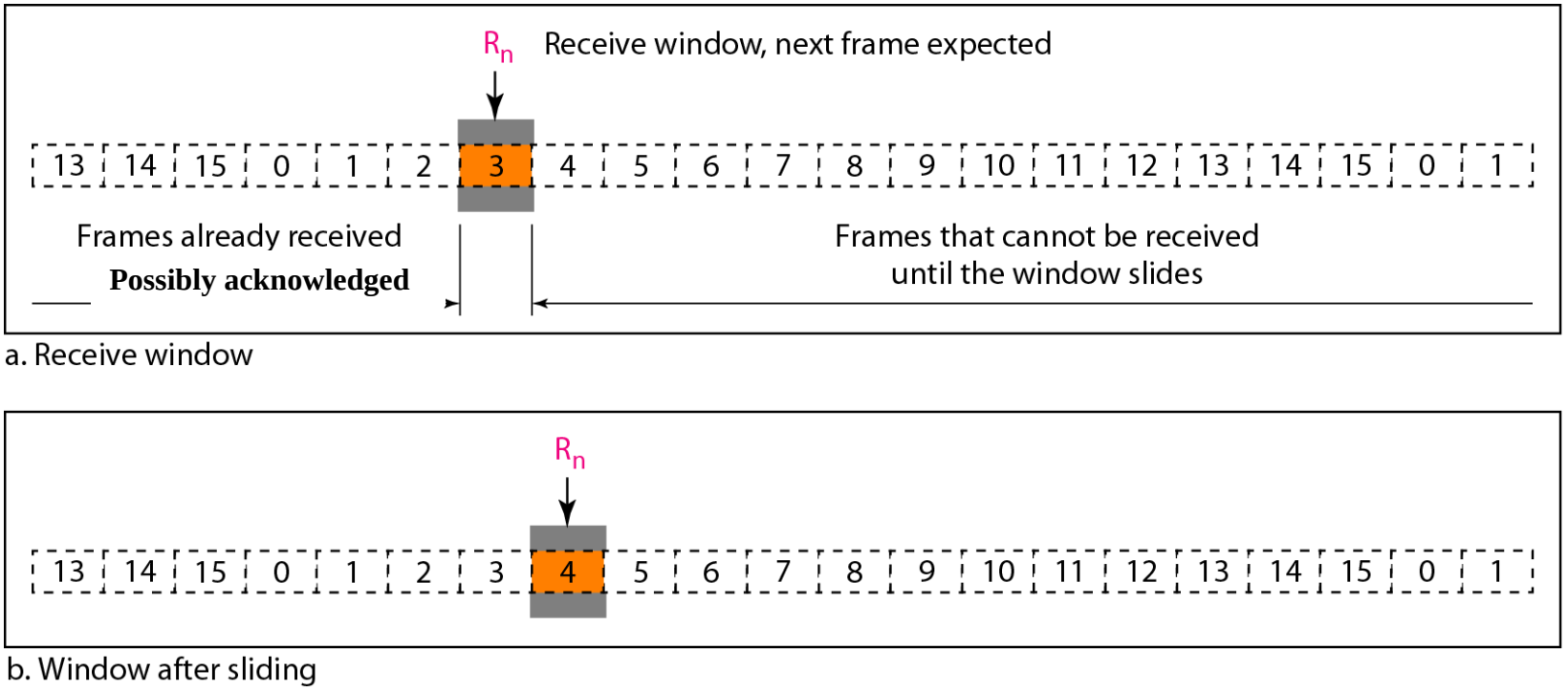


a. Send window before sliding



b. Send window after sliding

Figure 11.13 *Receive window for Go-Back-N ARQ*



Once a frame is received without error and in order, the window will slides over to the next placeholder

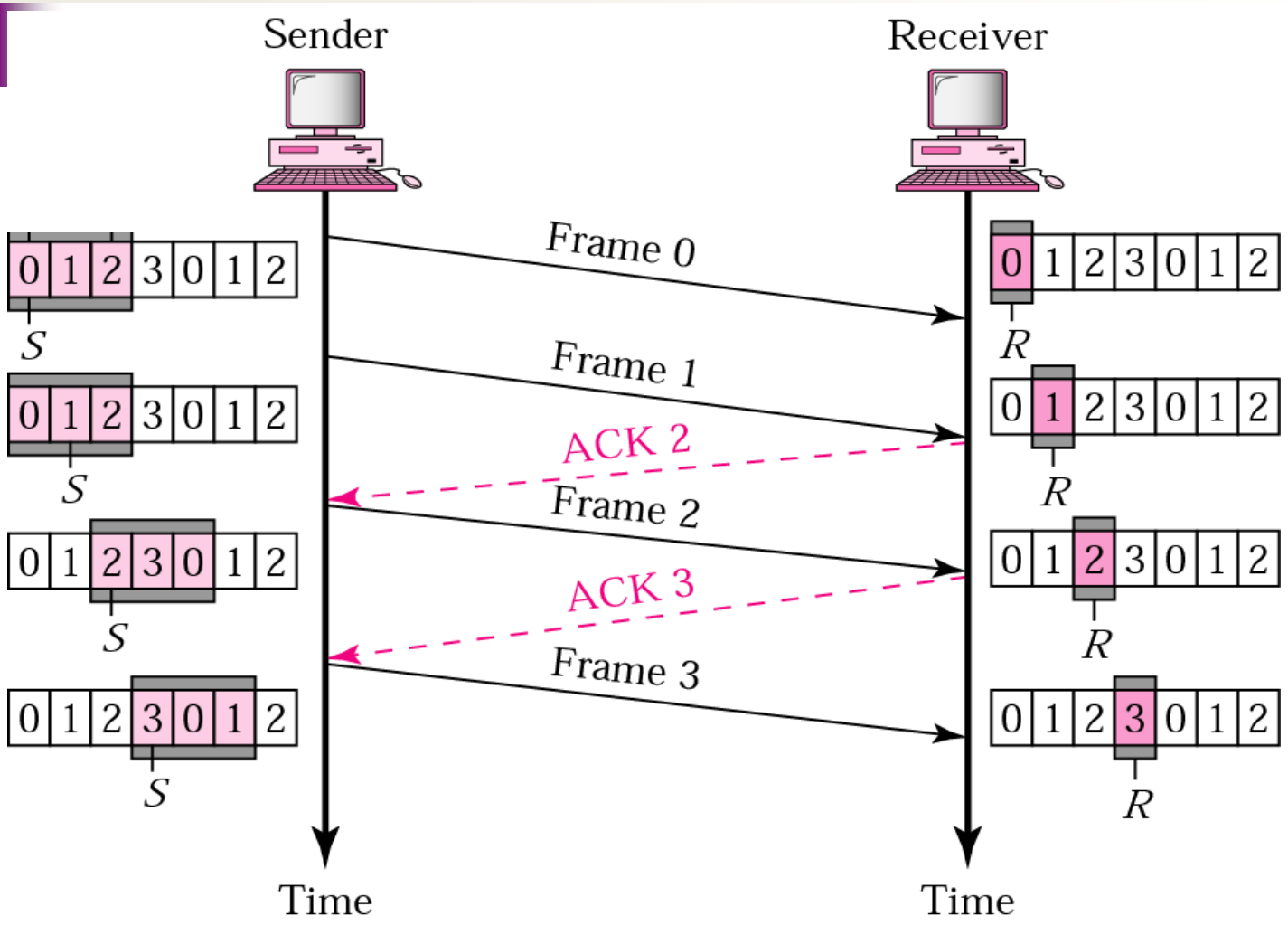
Note

The receive window is an abstract concept defining an imaginary box of size 1 with one single variable R_n. The window slides when a correct frame has arrived; sliding occurs one slot at a time.

11.5 Go-Back-N ARQ

- **Setup –Receiver sliding window**
- The size of the receiver window is always **1** and points to the **next expected frame** number to arrive
- This means that frames should arrive **in order**
- If the expected frame is received without errors, the receiver window slides over the **next sequence number**.
- **Operation**
- The receiver sends a positive ACK if a frame has arrived **without error** and **in order** (with the expected sequence number)
- Receiver does not have to acknowledge each individual frame received correctly and in order.
- Receiver can send **cumulative ACK** for several frames (**ACK 5 acknowledges frames (0,1,2,3,4) and expecting frame 5)**
- If the frame is **damaged or out-of-order**, the receiver **discards it** (and **stay *silent***) and also **discards all subsequent frames** until it receives the one expected.
  - In this case, **no** ACK will be transmitted
- If the sender timer expires before receiving an ACK, it will **resend *ALL* frames beginning with the one expired until the last one sent** (Go-Back-N).

Go-Back-N ARQ, normal operation



Sender window will slide to frame number (i) when (ACK i) is received.

Go-Back-N ARQ, lost frame

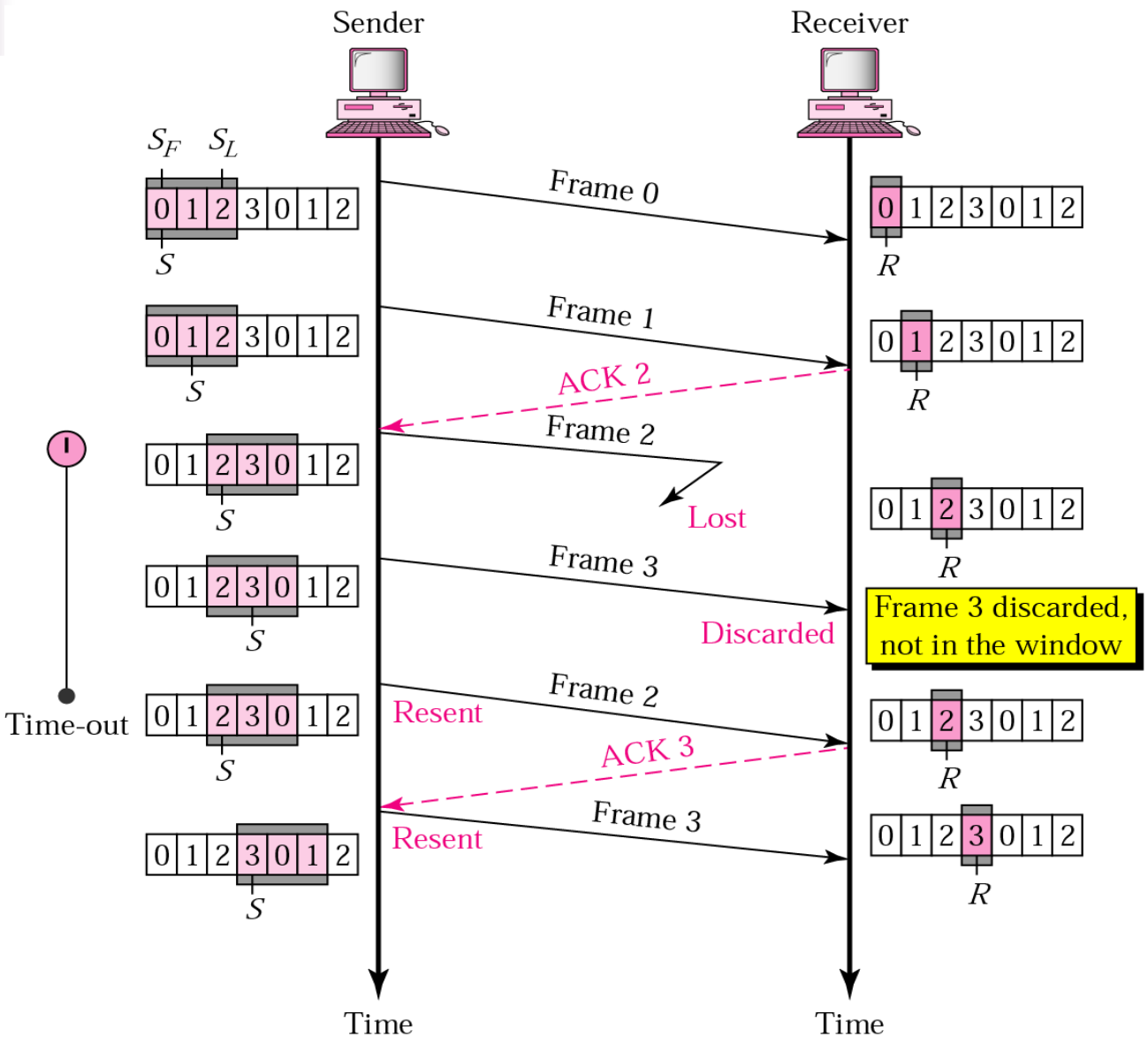
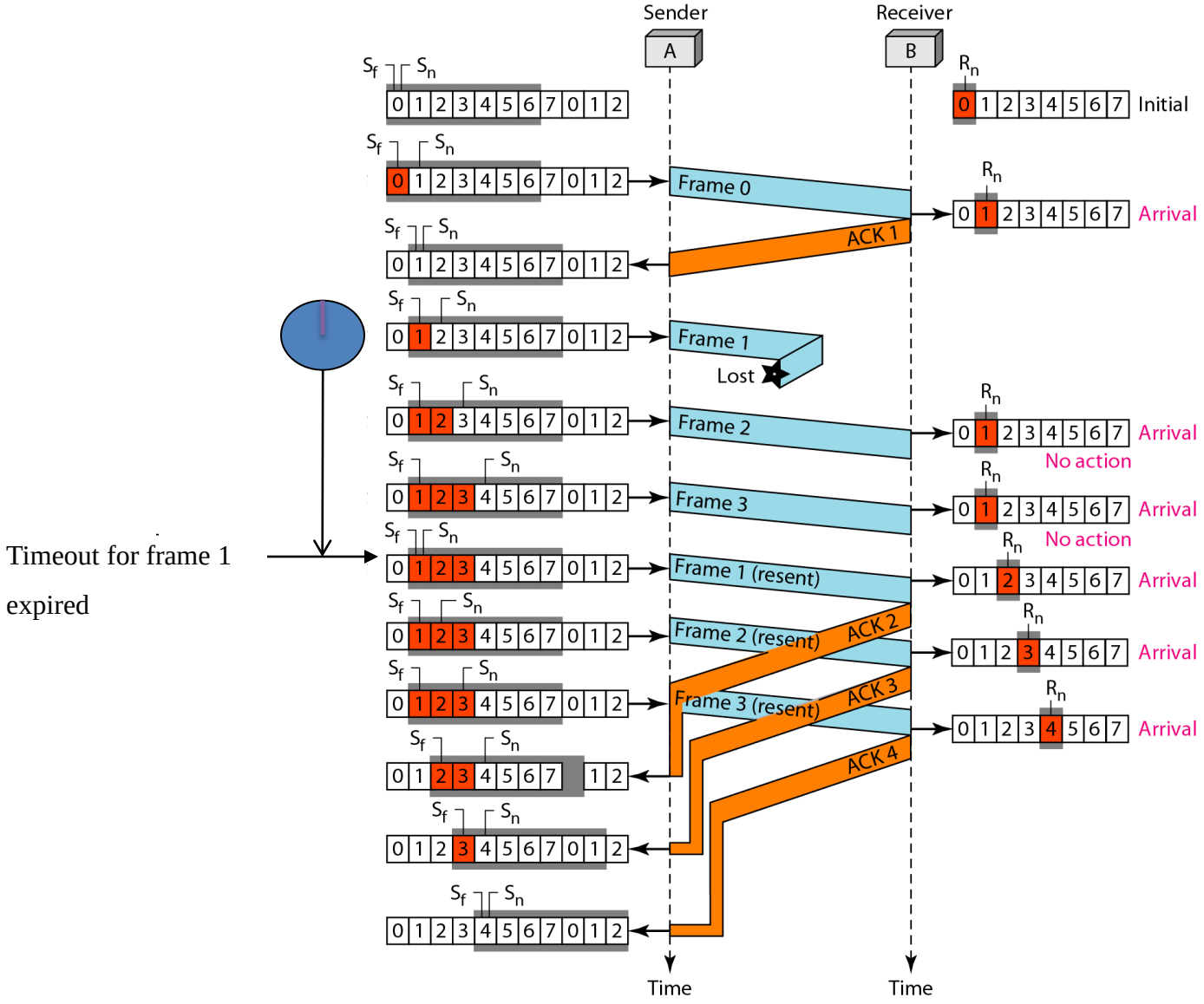


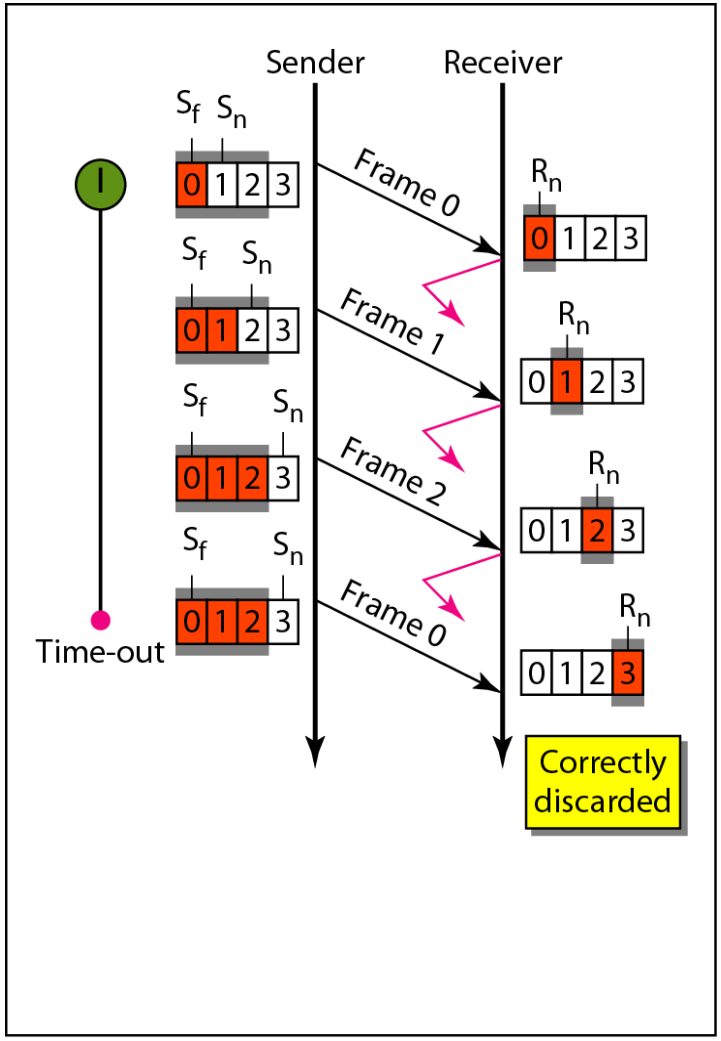
Figure 11.17 *Flow diagram for Example 11.7*



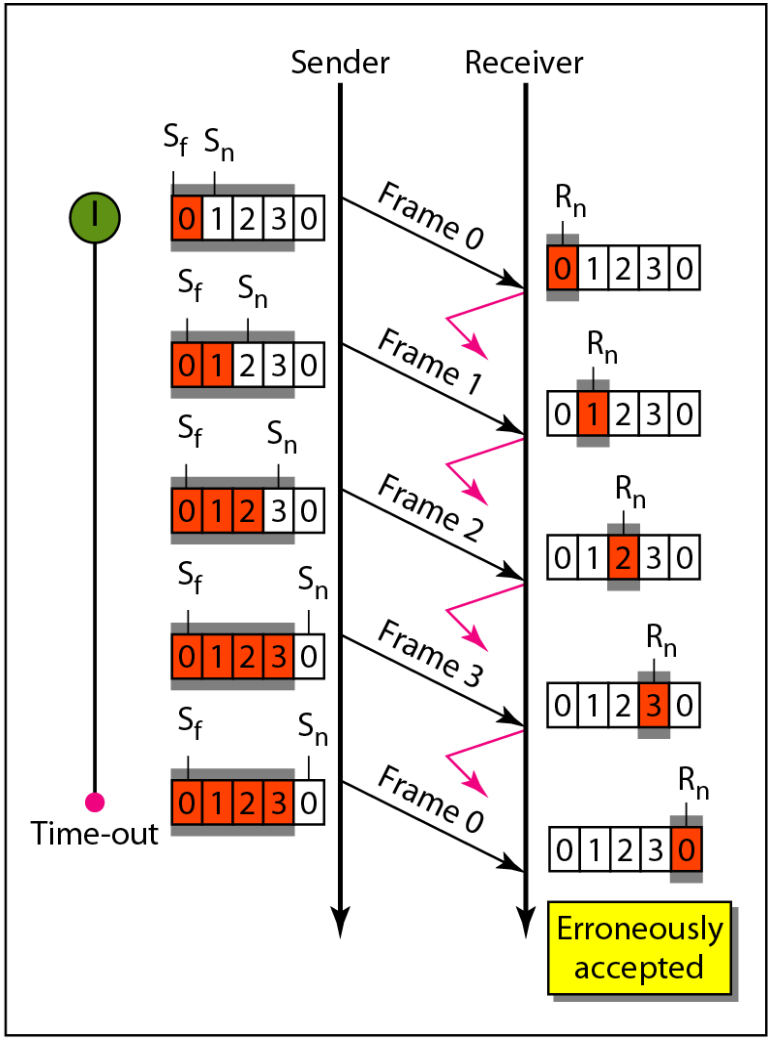
Note

In Go-Back-N ARQ, the size of the send window must be less than 2^m; the size of the receiver window is always 1.

Figure 11.15 *Window size for Go-Back-N ARQ*



a. Window size < 2^m



b. Window size = 2^m

Performance

- Better than Stop and wait because it keeps the sender busy while waiting for acknowledgement
- Does not allow error free but out of order frames to be accepted by the receiver
- This protocol **can waste a lot of bandwidth** if the error rate is high because it requests the sender to retransmit the frame in error and all the subsequently transmitted frames

11.5 Selective-Repeat ARQ

- **Setup-Window size (for both sender and receiver)**
  - For m bit sequence number **the maximum** window size 2^{m-1}
- **Operation**
  - Sender in case of **damaged or lost frame** retransmits
    - Those which are **negatively acknowledged**
    - Those for which timer expires
  - Receiver can **accept out of order frames** and buffer (store) them **until the lost, damaged, or delayed frame arrives.**
  - Receiver **does not acknowledge out of order frames** but **buffers** them only
  - If the receiver receives an **out-of-order, error free frame**, it will send a frame called **Negative Acknowledge (NAK)** with the number of the frame to be retransmitted only.
  - NAK improves the performance because it requests retransmission of the lost frame **before** the corresponding sender timer expires

Figure 11.18 *Send window for Selective Repeat ARQ*

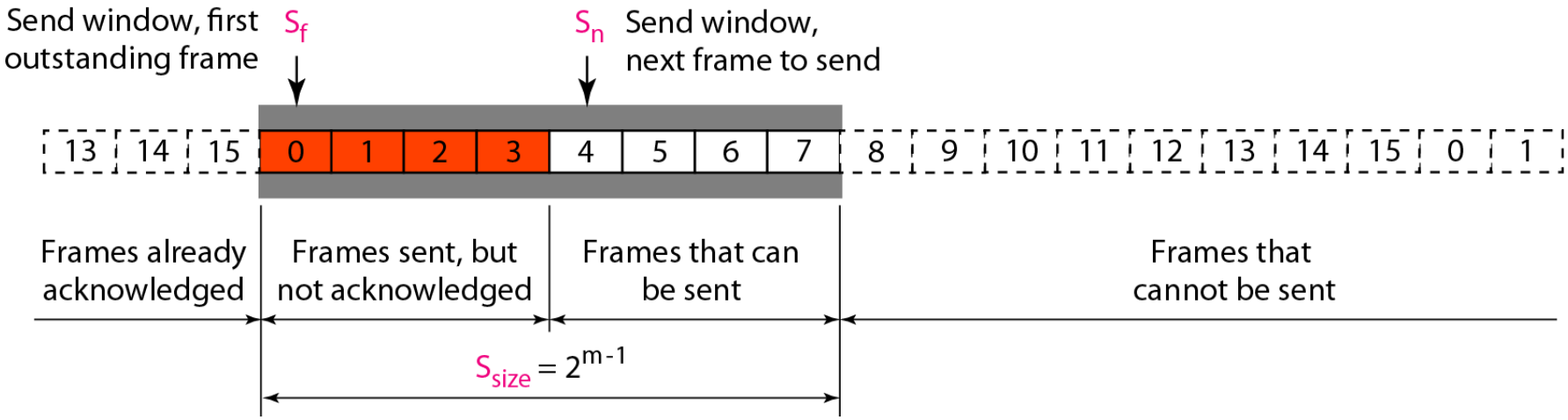
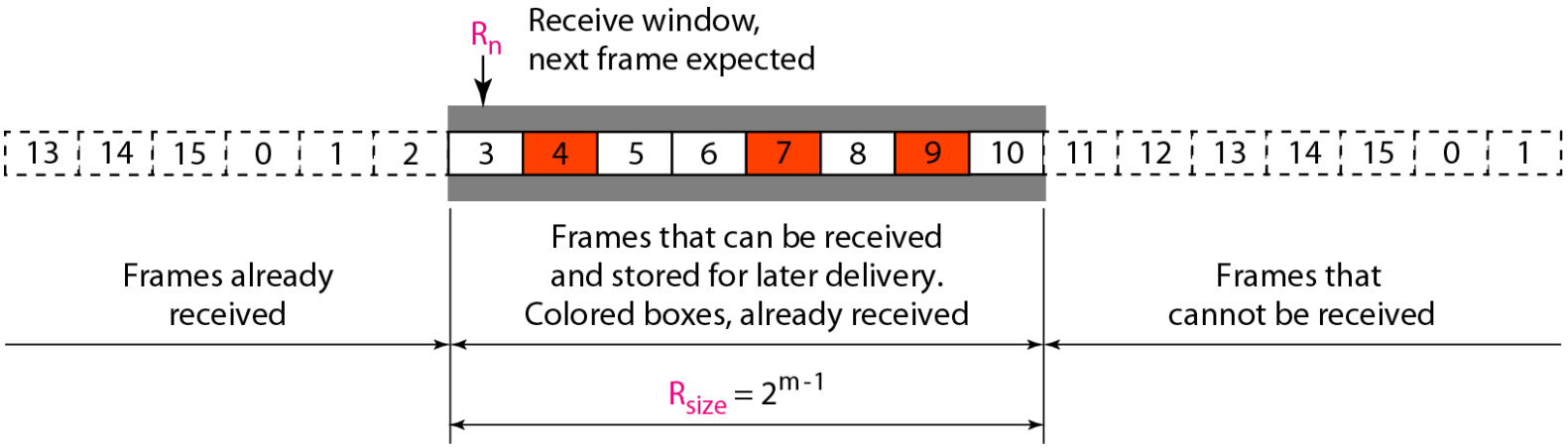


Figure 11.19 *Receive window for Selective Repeat ARQ*



- Receiver window is shifted to position (i) only when **receiver sends ACK i**
- ACK i can be sent if **all frames up to (i-1)** have been **received without error**

11.13 Selective Repeat ARQ, lost frame

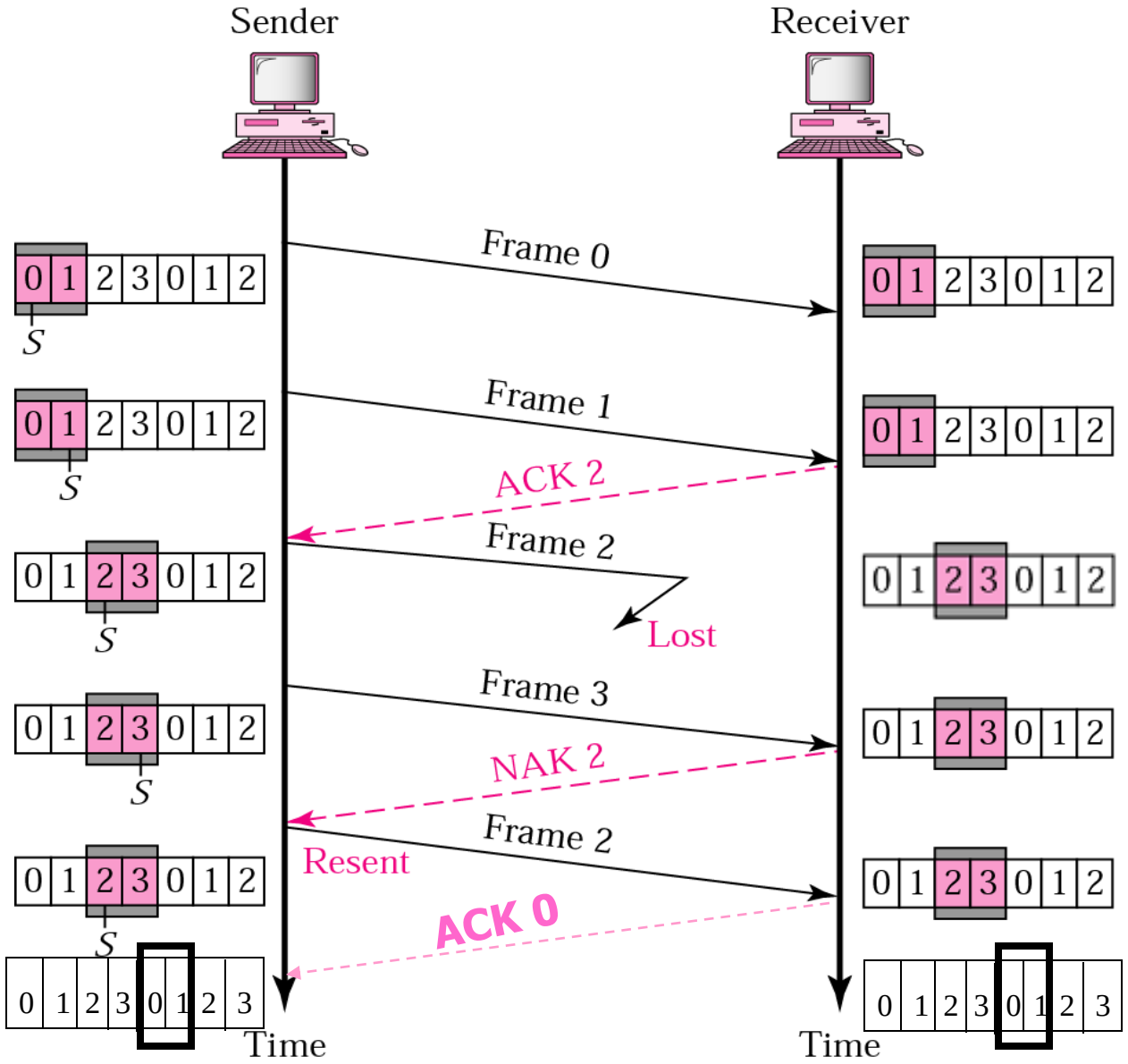


Figure 11.23 *Flow diagram for Example 11.8*

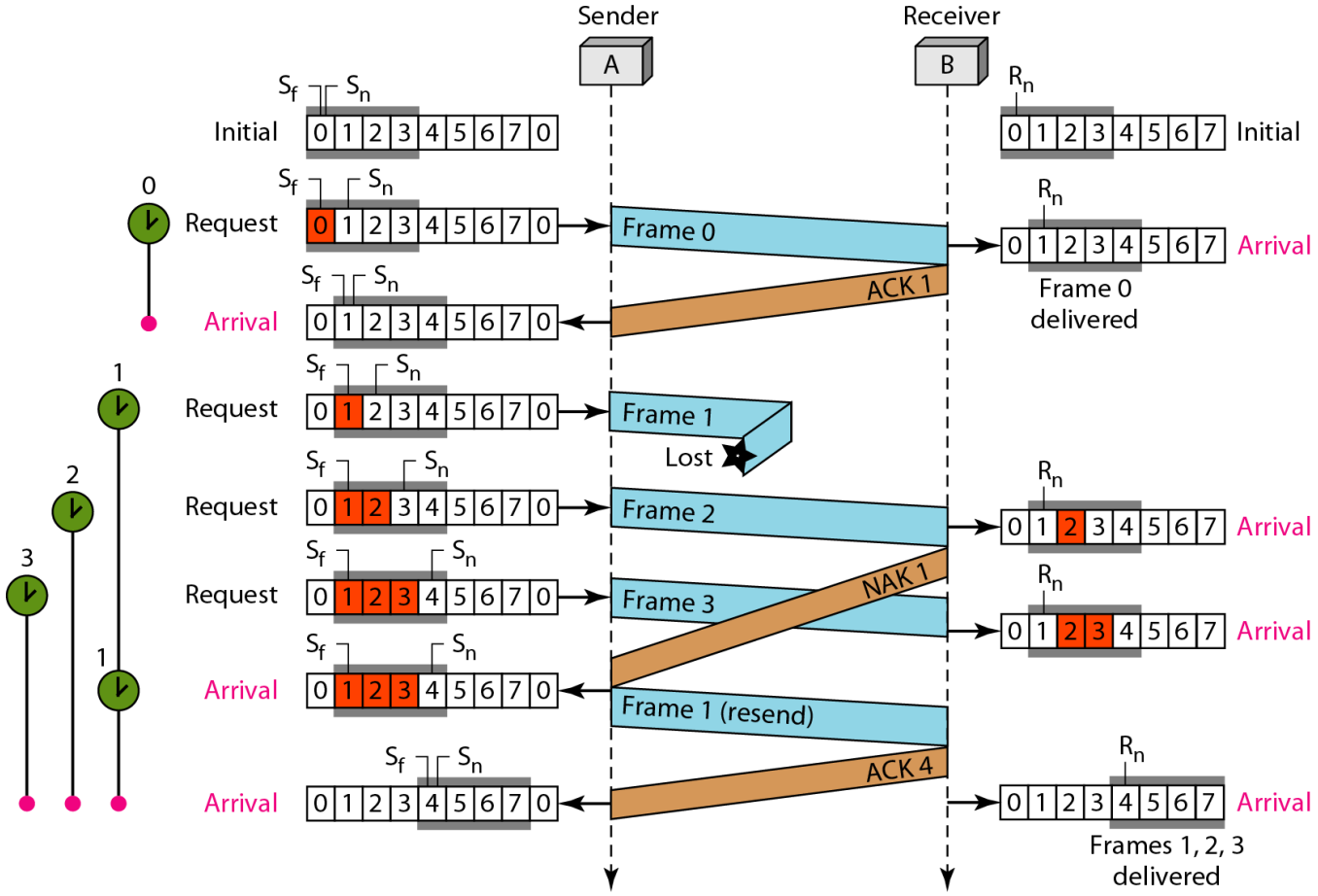
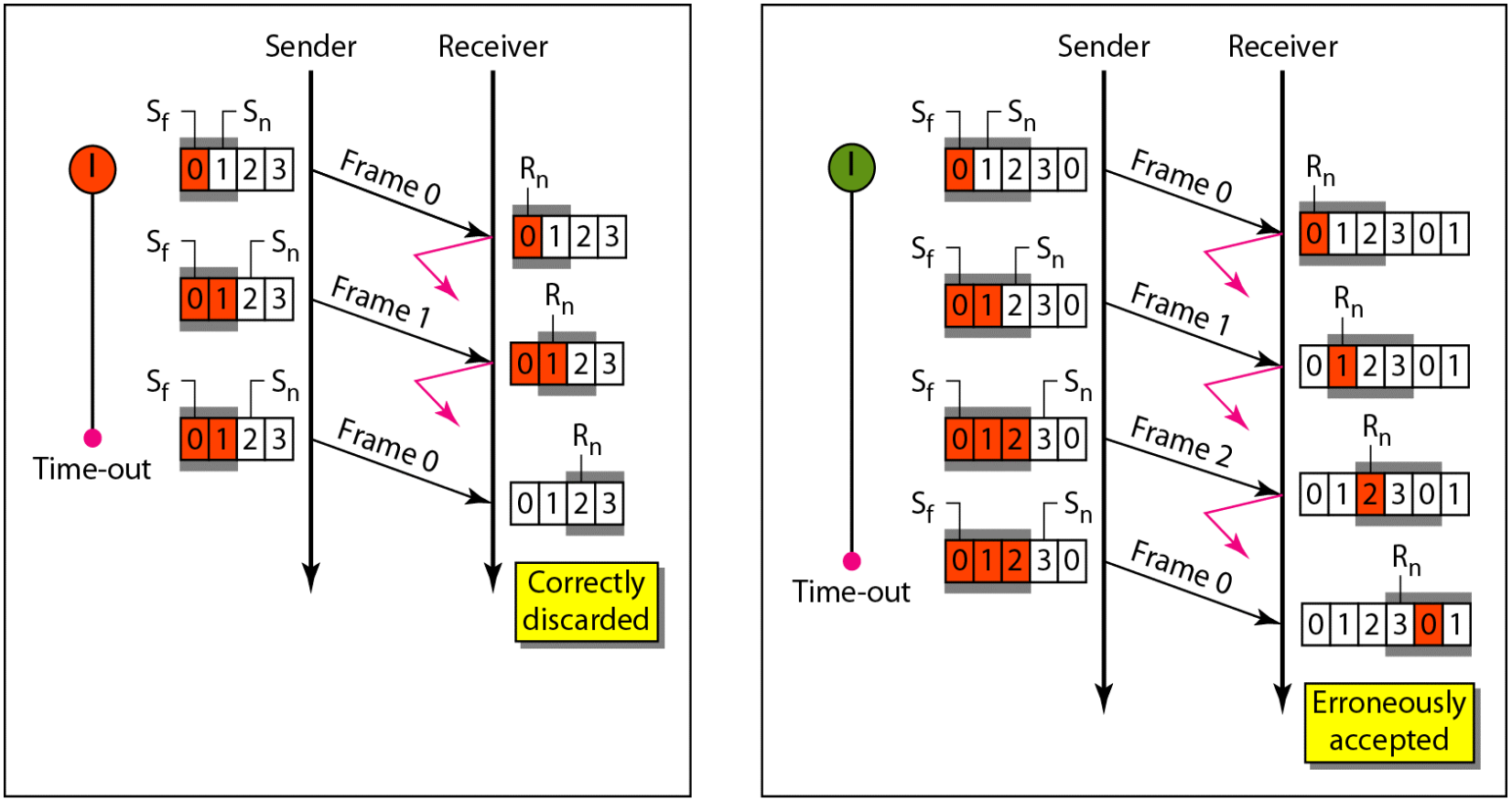


Figure 11.21 *Selective Repeat ARQ, window size*



a. Window size = 2^{m-1}

b. Window size > 2^{m-1}

Performance

- More efficient than the other two protocols because it **reduces number of retransmissions** for noisy links
- The receiver and transmitter processing logic is more complex
  - **Receiver** must be able to **reinsert** the retransmitted (lost, delayed, damaged) frame in the proper sequence after it arrives
  - **The sender** should be able to **send out of order** frame when requested by the sender using NAK
  - Needs **more memory** than Go-Back-N ARQ at the **receiver side**. The receiver memory is large if the window size is large

Pipelining

- Pipelining: beginning a task before the previous task has ended
- No pipelining in Stop-And-Wait ARQ
- Pipelining in Go-Back-N ARQ and Selective repeat ARQ (Several frames can be sent before we a sender receive any news about the previous frames)
- Pipelining increases the efficiency because it keeps the line busy instead of waiting