COMPUTER NETWORK (BCA301)

DEPARTMENT OF COMPUTER SCIENCE

PROGRAMME: BCA

CENTRAL UNIVERSITY OF ORISSA
KORAPUT

Data Link Layer

The two main functions of the data link layer are:

Data Link Control and **Media Access Control**.

Data Link Control (Flow Control & Error Control): It deals with the design and procedures for communication between two adjacent nodes: node-to-node communication.

Data link control functions include framing, flow and error control, and software-implemented protocols that provide smooth and reliable transmission of frames between nodes.

To implement data link control, we need protocols.

Media Access Control : media access control, or how to share the link.

FRAMING

- The data link layer, on the other hand, needs to pack bits into frames, so that each frame is distinguishable from another.
- Framing in the data link layer separates a message from one source to a destination, or from other messages to other destinations, by adding a sender address and a destination address.
- The destination address defines where the packet is to go; the sender address helps the recipient acknowledge the receipt.

Framing

- In fixed-size framing, there is no need for defining the boundaries of the frames; the size itself can be used as a delimiter.
- In variable-size framing, we need a way to define the end of the frame and the beginning of the next.
- Historically, two approaches were used for this purpose:

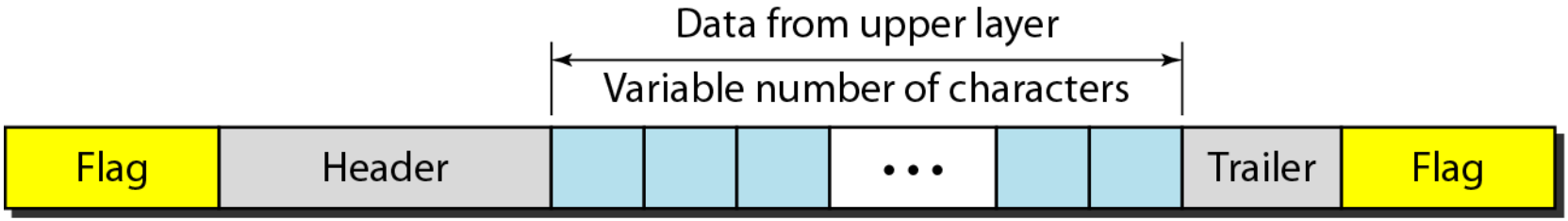
  a character-oriented approach and

  a bit-oriented approach.

Character-Oriented Protocol

- In a character-oriented protocol, data to be carried are 8-bit characters from a coding system such as ASCII.
- The header, which normally carries the source and destination addresses and other control information, and the trailer, which carries error detection or error correction redundant bits, are also multiples of 8 bits.
- To separate one frame from the next, an 8-bit (I-byte) flag is added at the beginning and the end of a frame.
- It acts as signals the start or end of a frame.
- It was popular when only text was exchanged by the data link layers.
- The flag could be selected to be any character not used for text communication.
- Any pattern used for the flag could also be part of the information.
- If this happens, the receiver, when it encounters this pattern in the middle of the data, thinks it has reached the end ofthe frame.
- Solution: byte-stuffing strategy

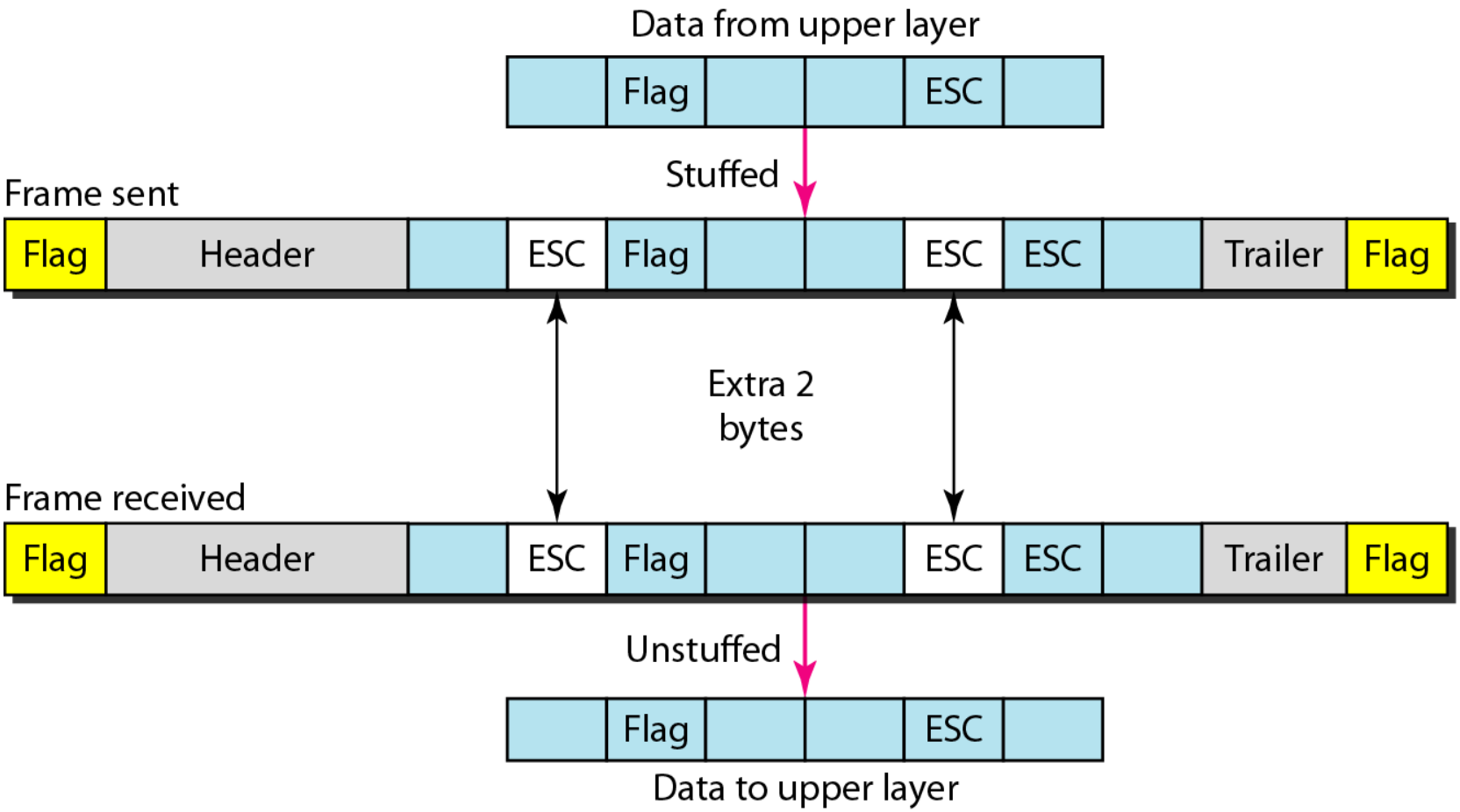
A Frame in a Character-Oriented Protocol



Byte Stuffing

- A special byte is added to the data section of the frame when there is a character with the same pattern as the flag.
- The data section is stuffed with an extra byte. This byte is usually called the escape character (ESC), which has a predefined bit pattern.
- Whenever the receiver encounters the ESC character, it removes it from the data section and treats the next character as data, not a delimiting flag.
- What happens if the text contains one or more escape characters followed by a flag?
- The receiver removes the escape character, but keeps the flag, which is incorrectly interpreted as the end of the frame.
- To solve this problem, the escape characters that are part of the text must also be marked by another escape character.
- In other words, if the escape character is part of the text, an extra one is added to show that the second one is part of the text.

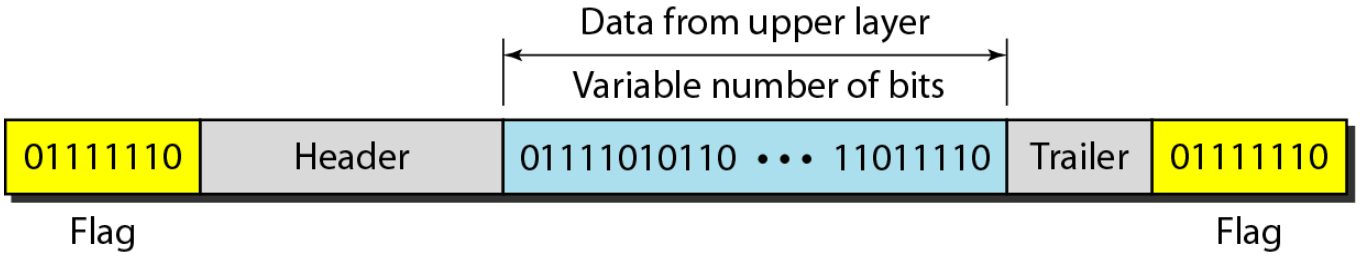
Byte Stuffing and Unstuffing



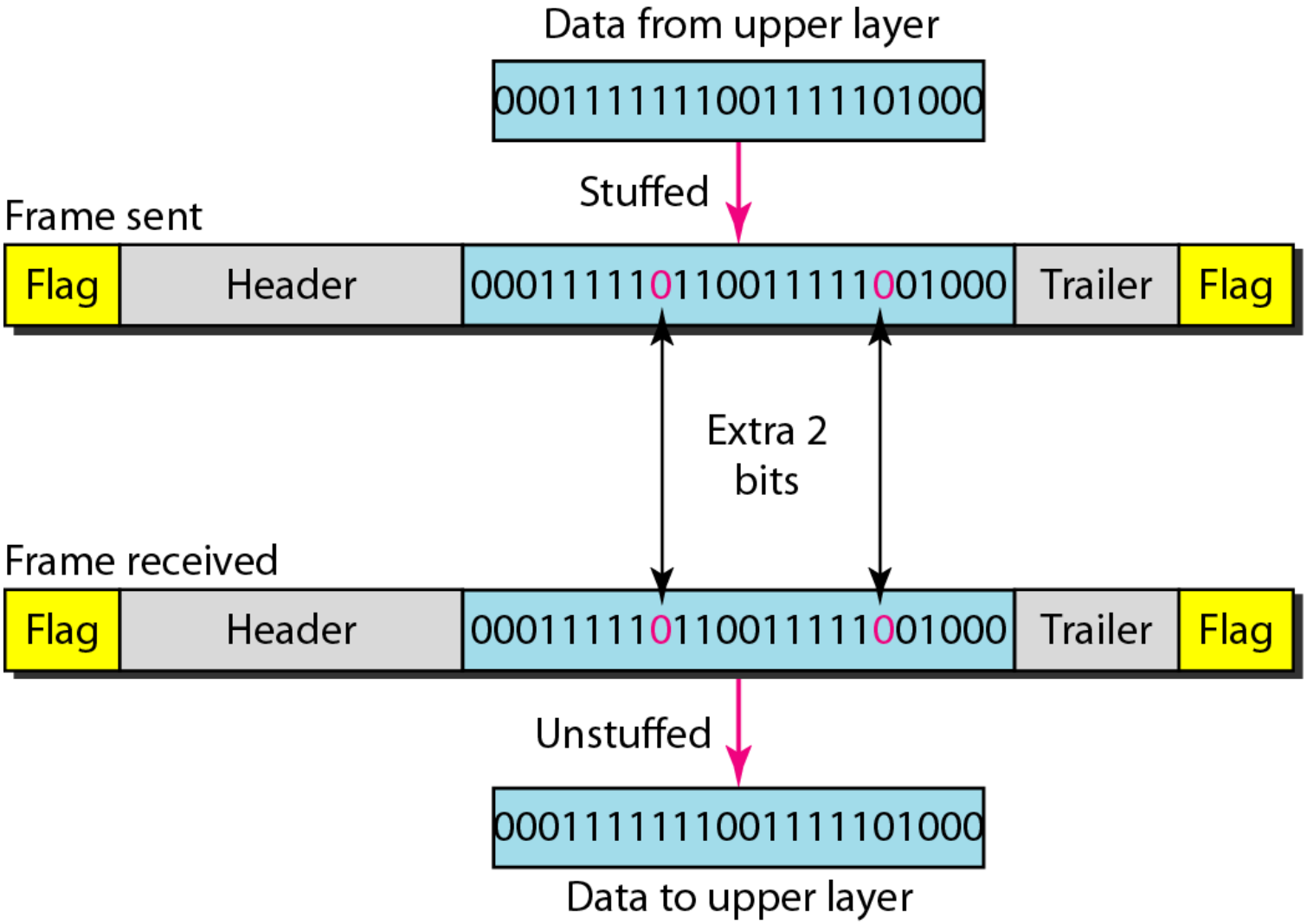
Bit-Oriented Protocol

- In a bit-oriented protocol, the data section of a frame is a sequence of bits to be interpreted by the upper layer as text, graphic, audio, video, and so on.
- It need a delimiter to separate one frame from the other.
- if the flag pattern appears in the data, we need to somehow inform the receiver that this is not the end of the frame.
- We do this by stuffing 1 single bit (instead of I byte) to prevent the pattern from looking like a flag.
- The strategy is called bit stuffing.
- In bit stuffing, if a 0 and five consecutive I bits are encountered, an extra 0 is added.
- This extra stuffed bit is eventually removed from the data by the receiver.

- The extra bit is added after one 0 followed by five 1s regardless of the value of the next bit.
- This guarantees that the flag field sequence does not inadvertently appear in the frame.



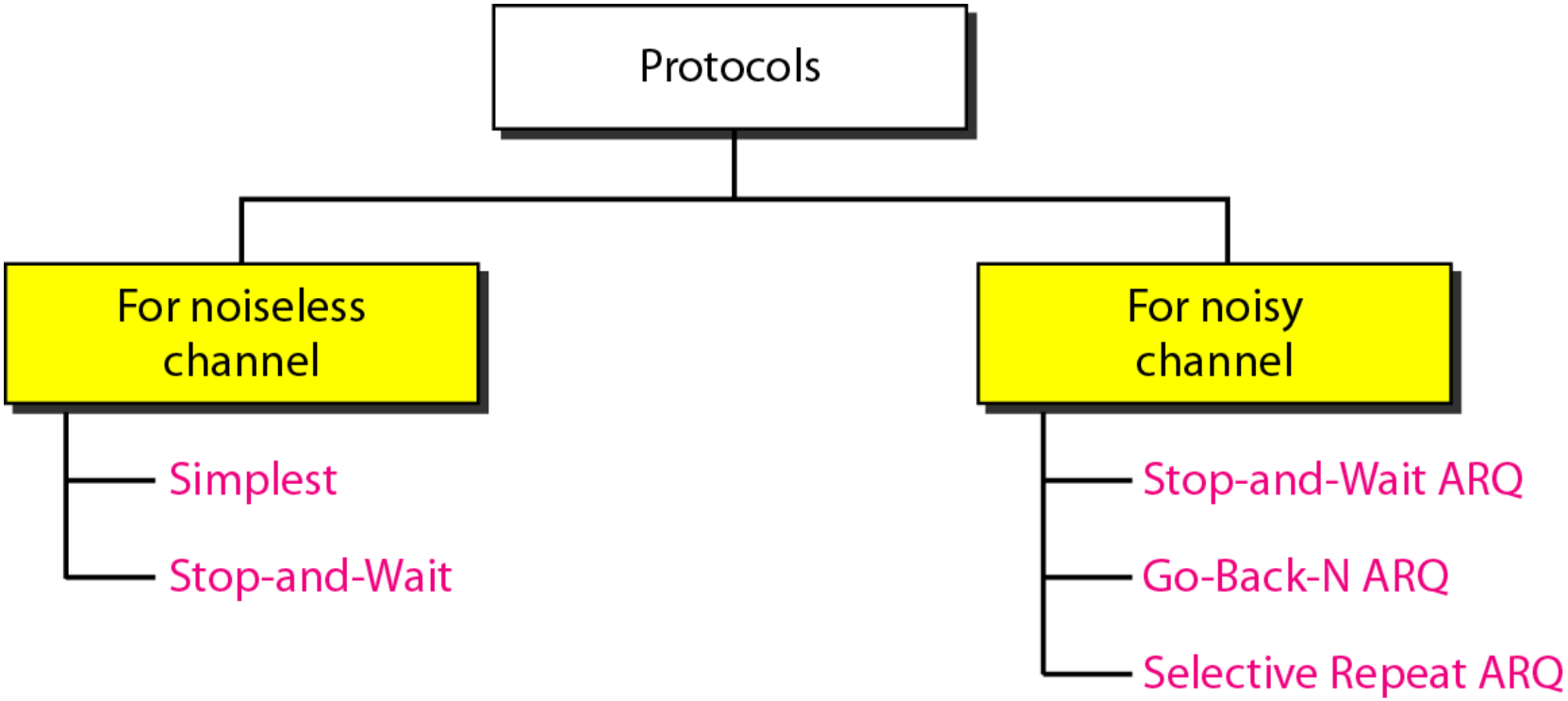
Bit Stuffing and Unstuffing



Flow Control & Error 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.
- Procedure is defined to set how much data the sender can send and receiver can receive.
- For this reason, each receiving device has a block of memory, called a buffer, reserved for storing incoming data until they are processed.
- If the buffer begins to fill up, the receiver must be able to tell the sender to halt transmission until it is once again able to receive.
- Error control is both error detection and error correction. It allows the receiver to inform the sender of any frames lost or damaged in transmission and coordinates the retransmission of those frames by the sender.

TAXONOMY OF PROTOCOLS



PROTOCOLS

- The data link layer can combine framing, flow control, and error control to achieve the delivery of data from one node to another.
- The protocols are normally implemented in software by using one of the common programming languages.
- Protocols that can be used for noiseless (error-free) channels and that can be used for noisy (error-creating) channels.
- The protocols in the first category cannot be used in real life, but they serve as a basis for understanding the protocols of noisy channels.
- All the protocols we discuss are unidirectional in the sense that the data frames travel from one node, called the sender, to another node, called the receiver.
- Although special frames, called acknowledgment (ACK) and negative acknowledgment (NAK) can flow in the opposite direction for flow and error control purposes, data flow in only one direction.

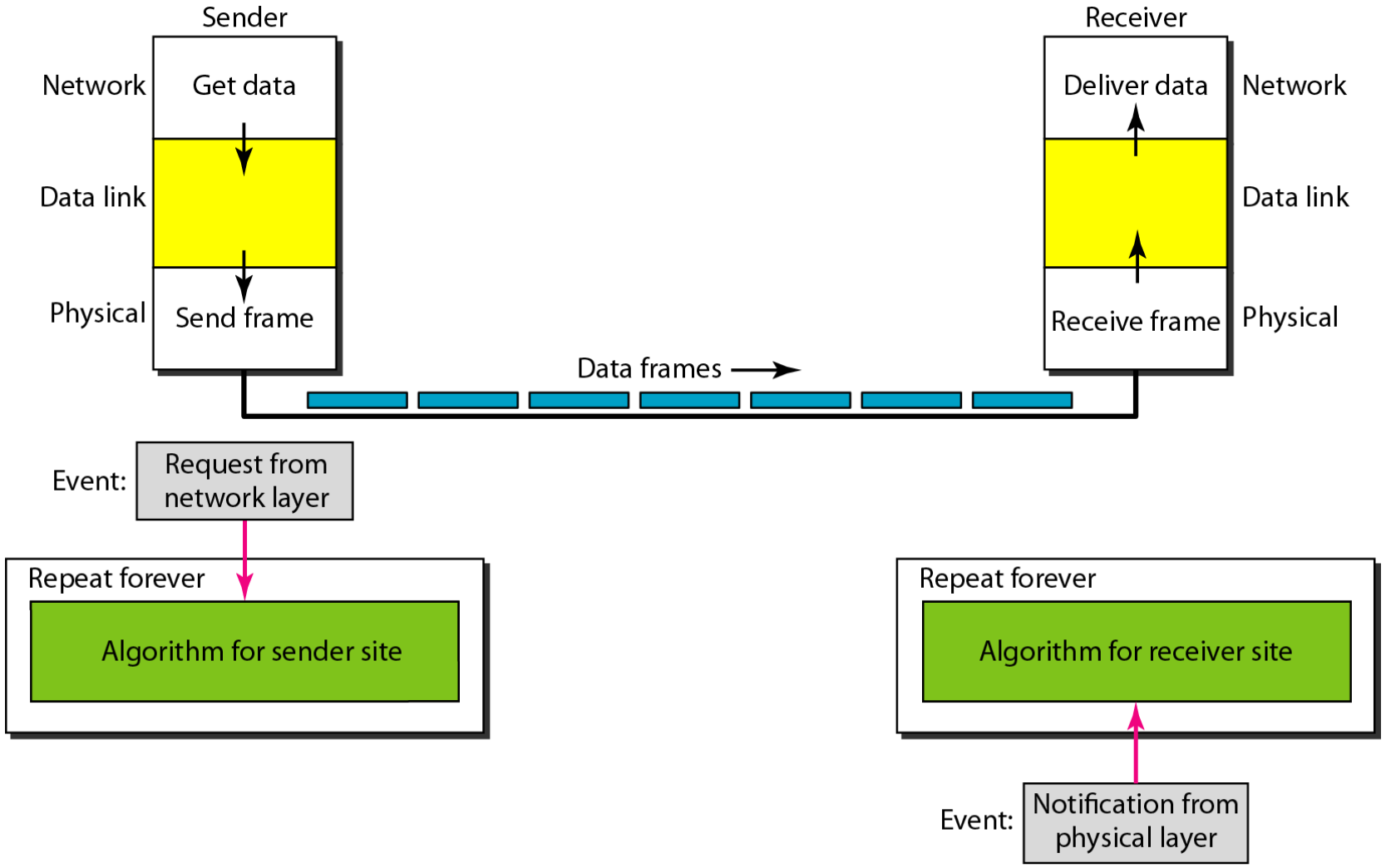
- In a real-life network, the data link protocols are implemented as bidirectional; data flow in both directions.
- In these protocols the flow and error control information such as ACKs and NAKs is included in the data frames in a technique called piggybacking.

Protocols For Noiseless Channels

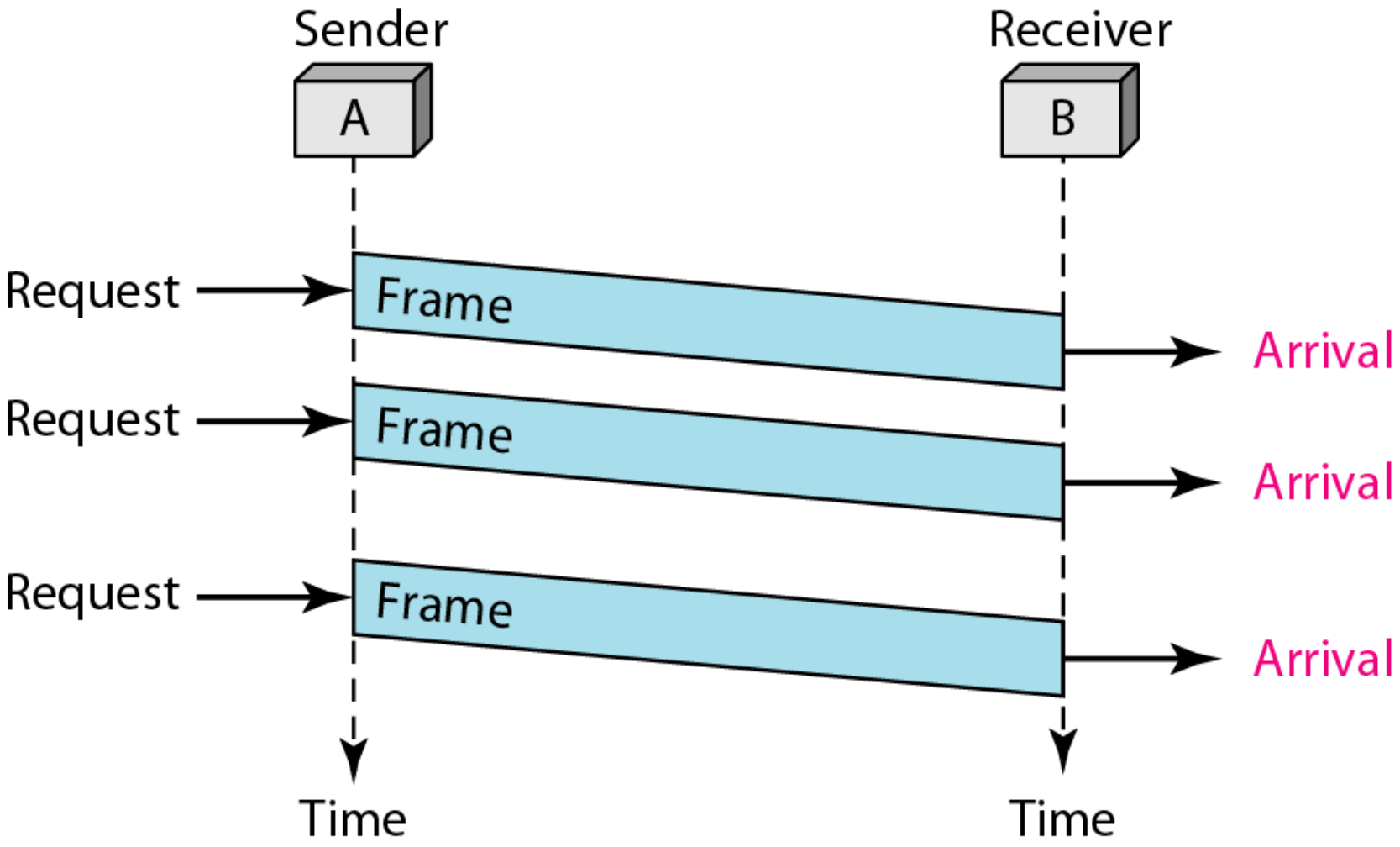
- Simplest Protocols is a protocol that does not use flow control; the second i.e Stop-and-Wait protocol is the one that does.
- Of course, neither has error control because we have assumed that the channel is a perfect noiseless channel.
- Simplest Protocols: It is a unidirectional protocol in which data frames are traveling in only one direction-from the sender to receiver.
- We assume that the receiver can immediately handle any frame it receives with a processing time that is small enough to be negligible.
- The data link layer of the receiver immediately removes the header from the frame and hands the data packet to its network layer.

- There is no need for flow control in this scheme. The data link layer at the sender site gets data from its network layer, makes a frame out of the data, and sends it.
- The data link layer at the receiver site receives a frame from its physical layer, extracts data from the frame, and delivers the data to its network layer.
- The data link layers of the sender and receiver provide transmission services for their network layers.

The Design of the Simplest Protocol with no Flow or Error Control



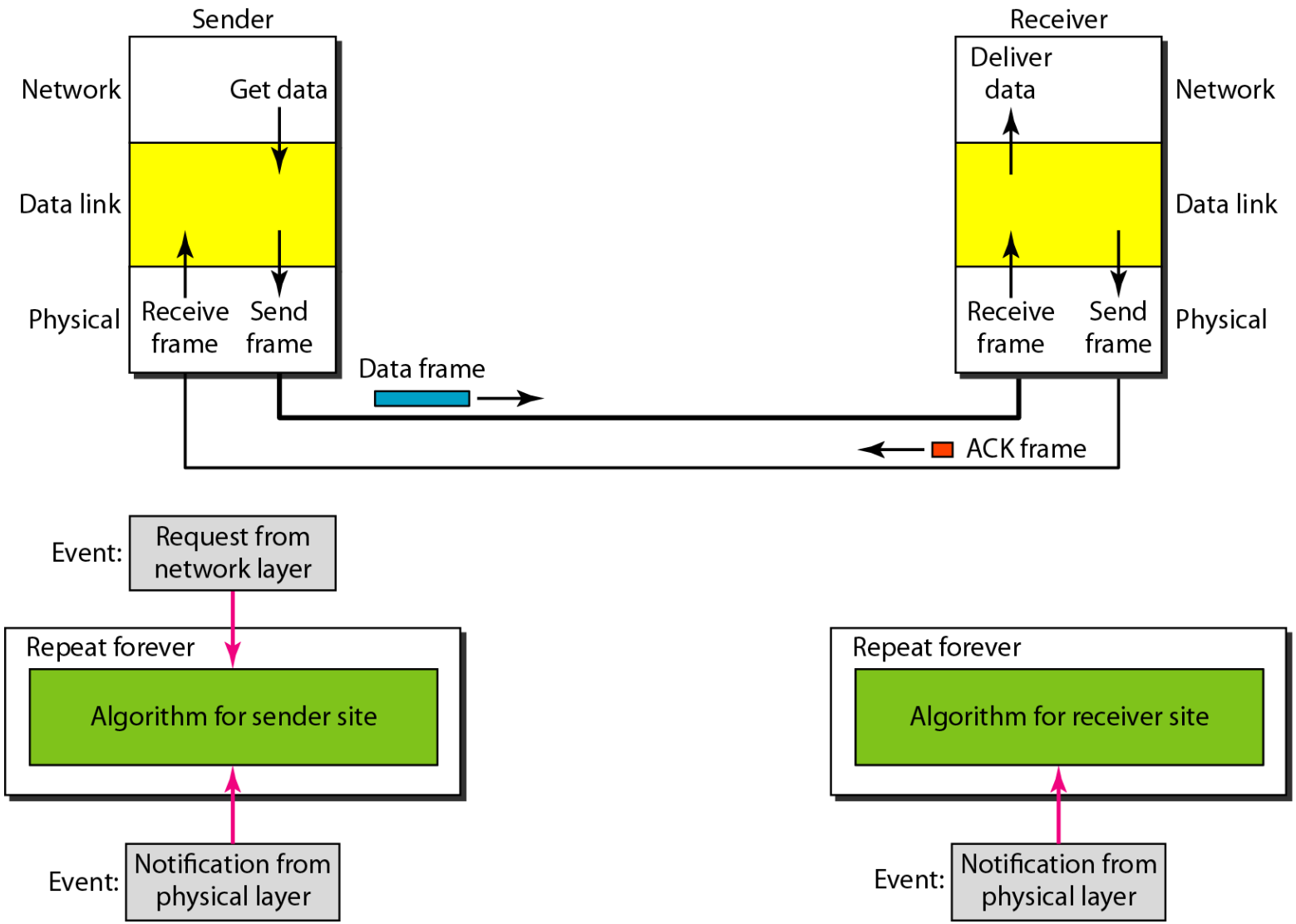
Flow Diagram



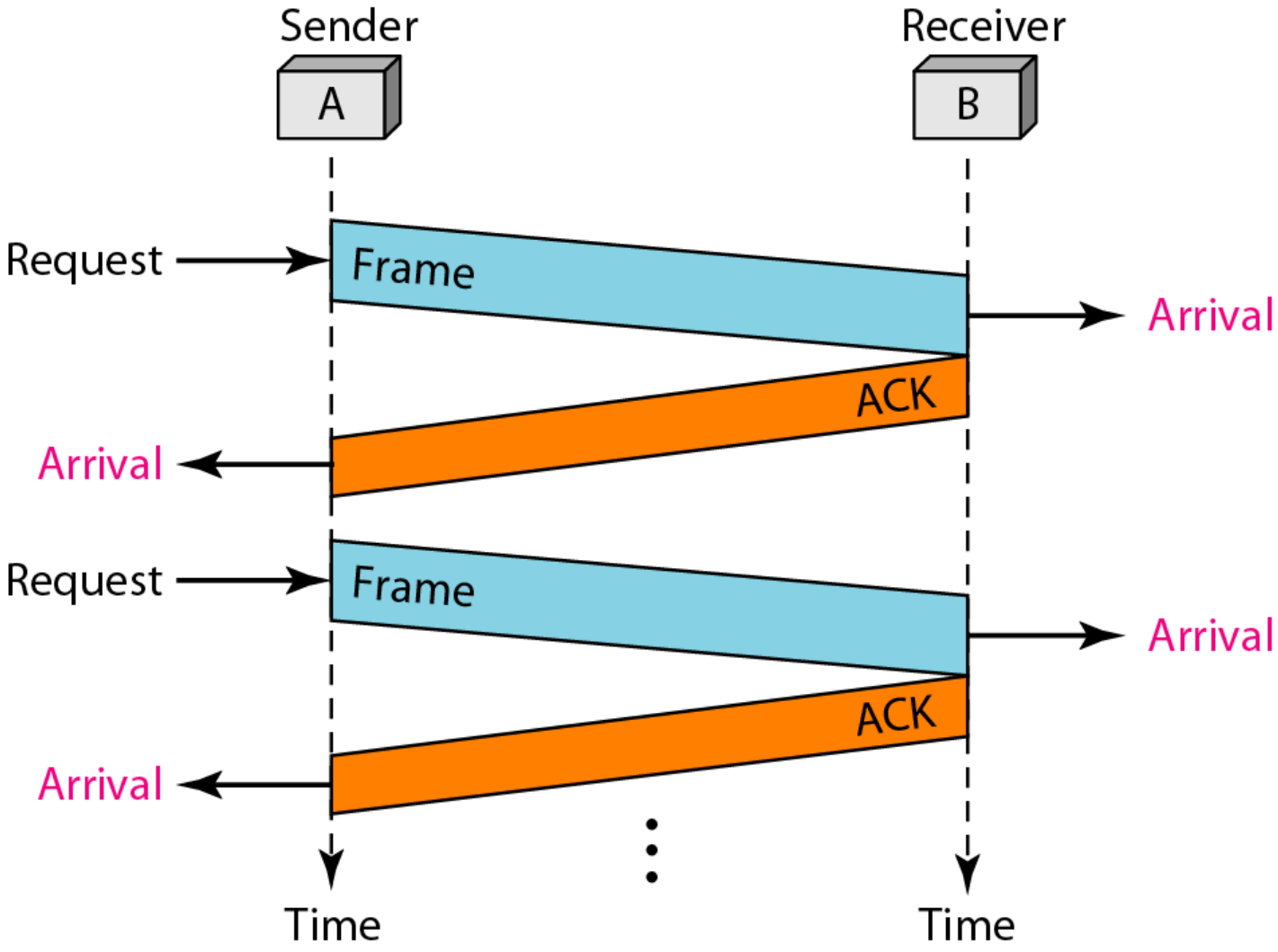
Stop-and-Wait Protocol

- The sender sends one frame, stops until it receives confirmation from the receiver and then sends the next frame.
- At any time, there is either one data frame on the forward channel or one ACK frame on the reverse channel.
- We therefore need a half-duplex link.

Design of Stop-and-Wait Protocol



Flow Diagram



Noisy Channel

- Stop-and-Wait Automatic Repeat Request:
- It adds a simple error control mechanism to the Stop-and-Wait Protocol.
- To detect and correct corrupted frames, we need to add redundancy bits to our data frame.
- When the frame arrives at the receiver site, it is checked and if it is corrupted, it is silently discarded.
- The sender keeps a copy of the sent frame. At the same time, it starts a timer.
- If the timer expires and there is no ACK for the sent frame, the frame is resent, the copy is held, and the timer is restarted.
- Since the protocol uses the stop-and-wait mechanism, there is only one specific frame that needs an ACK even though several copies of the same frame can be in the network.

Sequence Number

- Since an ACK frame can also be corrupted and lost, it too needs redundancy bits and a sequence number.
- The ACK frame for this protocol has a sequence number field.
- The protocol specifies that frames need to be numbered.
- A field is added to the data frame to hold the sequence number of that frame.
- The sequence numbers of course can wrap around.
- For example, if we decide that the field is m bits long, the sequence numbers start from 0, go to 2^m - 1, and then are repeated.

Acknowledgment Numbers

- Since the sequence numbers must be suitable for both data frames and ACK frames, we use this convention:
- The acknowledgment numbers always announce the sequence number of the next frame expected by the receiver.
- For example, if frame 0 has arrived safe and sound, the receiver sends an ACK frame with acknowledgment 1 (meaning frame 1 is expected next).
- If frame 1 has arrived safe and sound, the receiver sends an ACK frame with acknowledgment 0 (meaning frame 0 is expected).

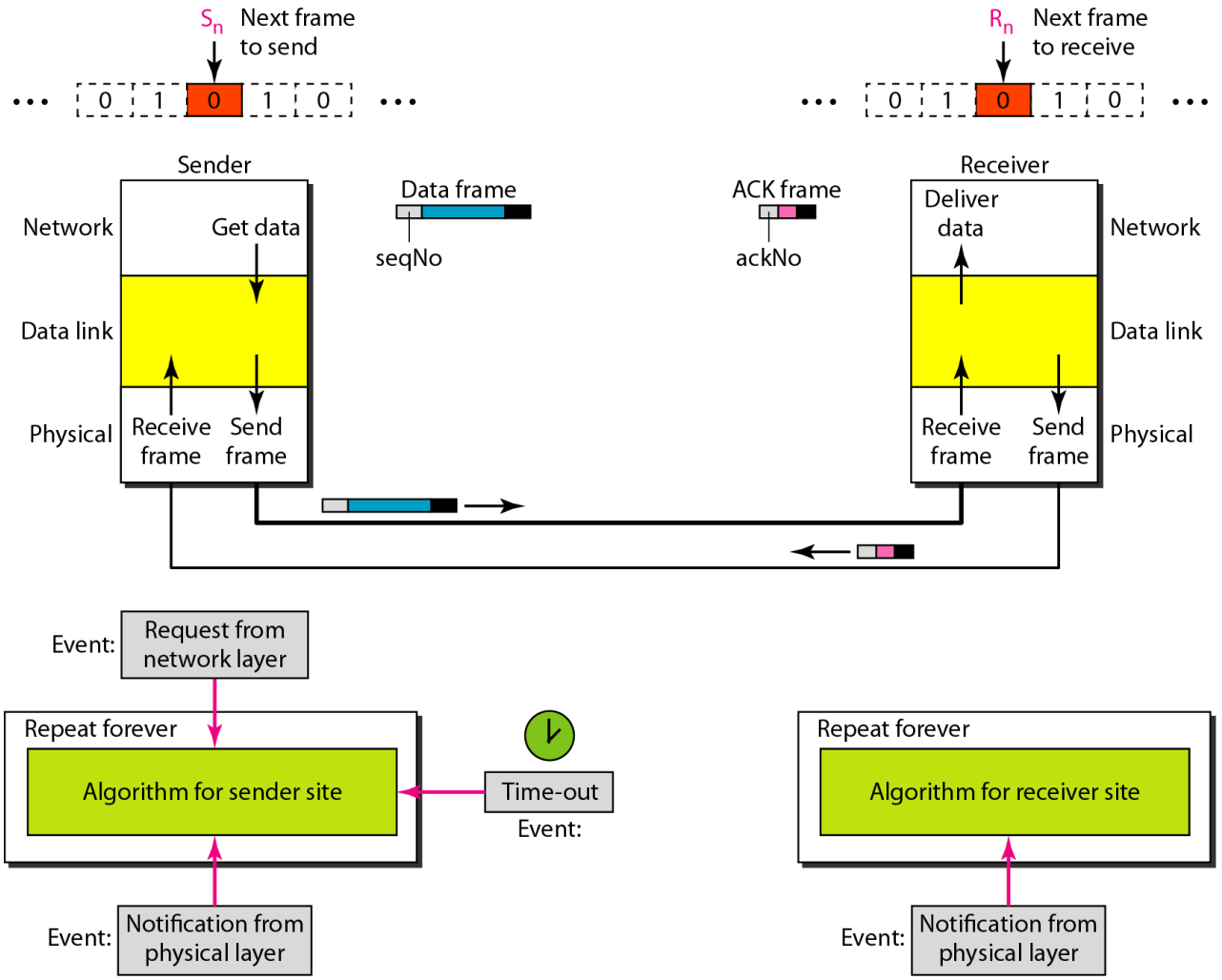
Design

- The sending device keeps a copy of the last frame transmitted until it receives an acknowledgment for that frame.
- A data frames uses a seqNo (sequence number); an ACK frame uses an ackNo (acknowledgment number).
- The sender has a control variable, which we call Sn (sender, next frame to send), that holds the sequence number for the next frame to be sent (0 or 1).
- The receiver has a control variable, which we call Rn (receiver, next frame expected), that holds the number of the next frame expected.
- When a frame is sent, the value of Sn is incremented (modulo-2), which means if it is 0, it becomes 1 and vice versa.
- When a frame is received, the value of Rn is incremented (modulo-2), which means if it is 0, it becomes 1 and vice versa.
- Three events can happen at the sender site; one event can happen at the receiver site.

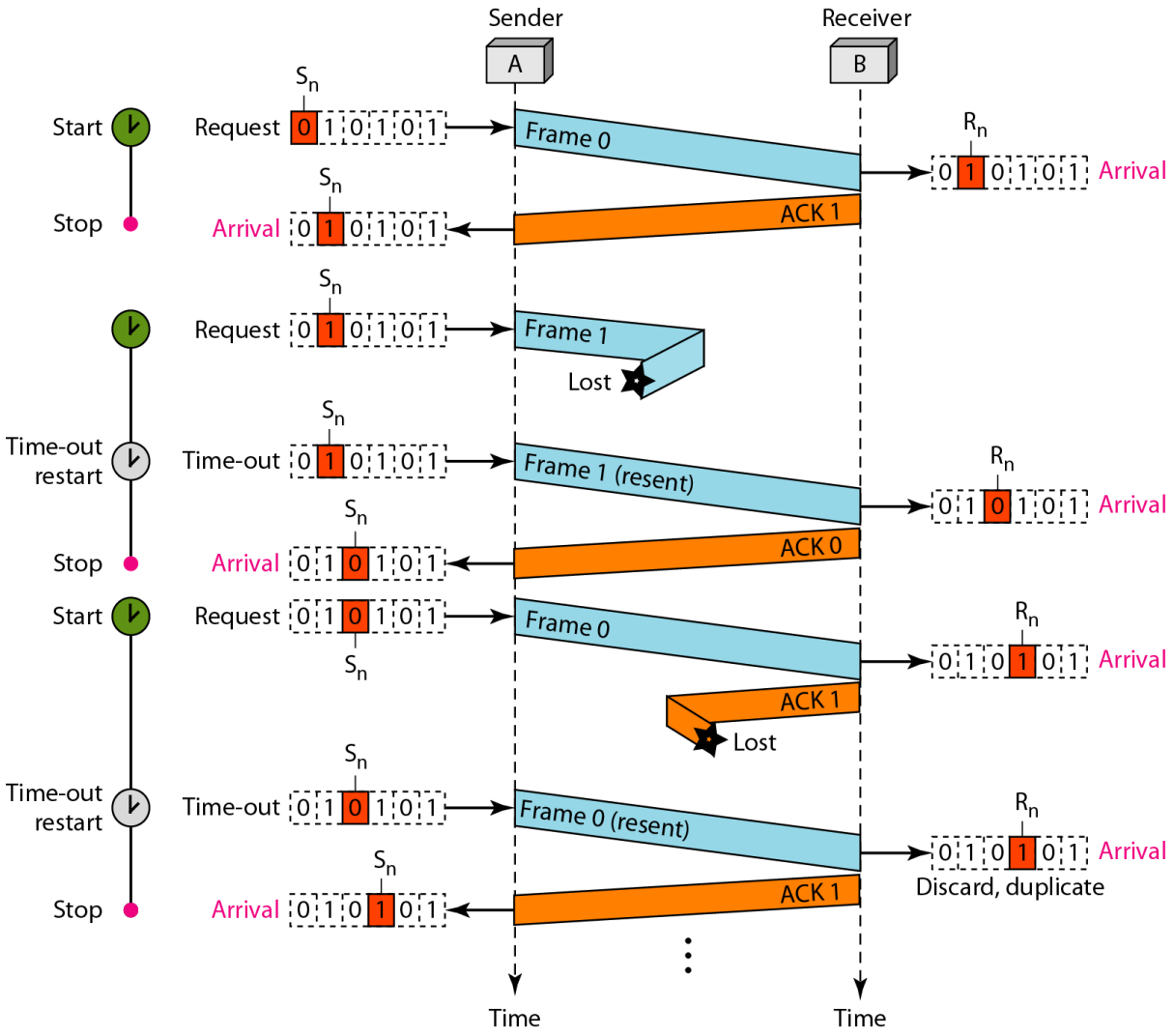
Efficiency

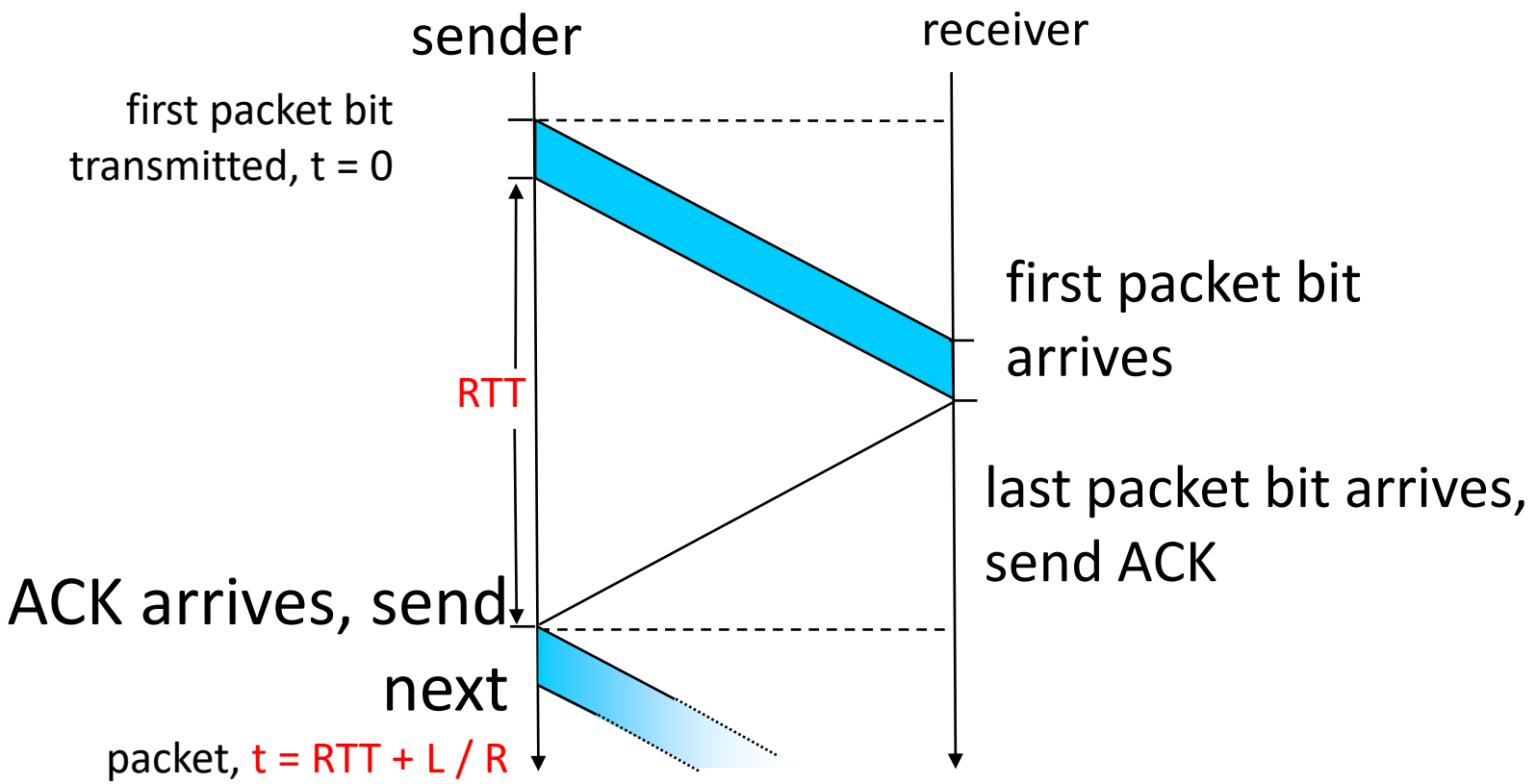
- The Stop-and-WaitARQ discussed in the previous section is very inefficient if our channel is thick and long.
- By thick, we mean that our channel has a large bandwidth; by long, we mean the round-trip delay is long.

Design of the Stop-and-Wait ARQ Protocol



Flow Diagram

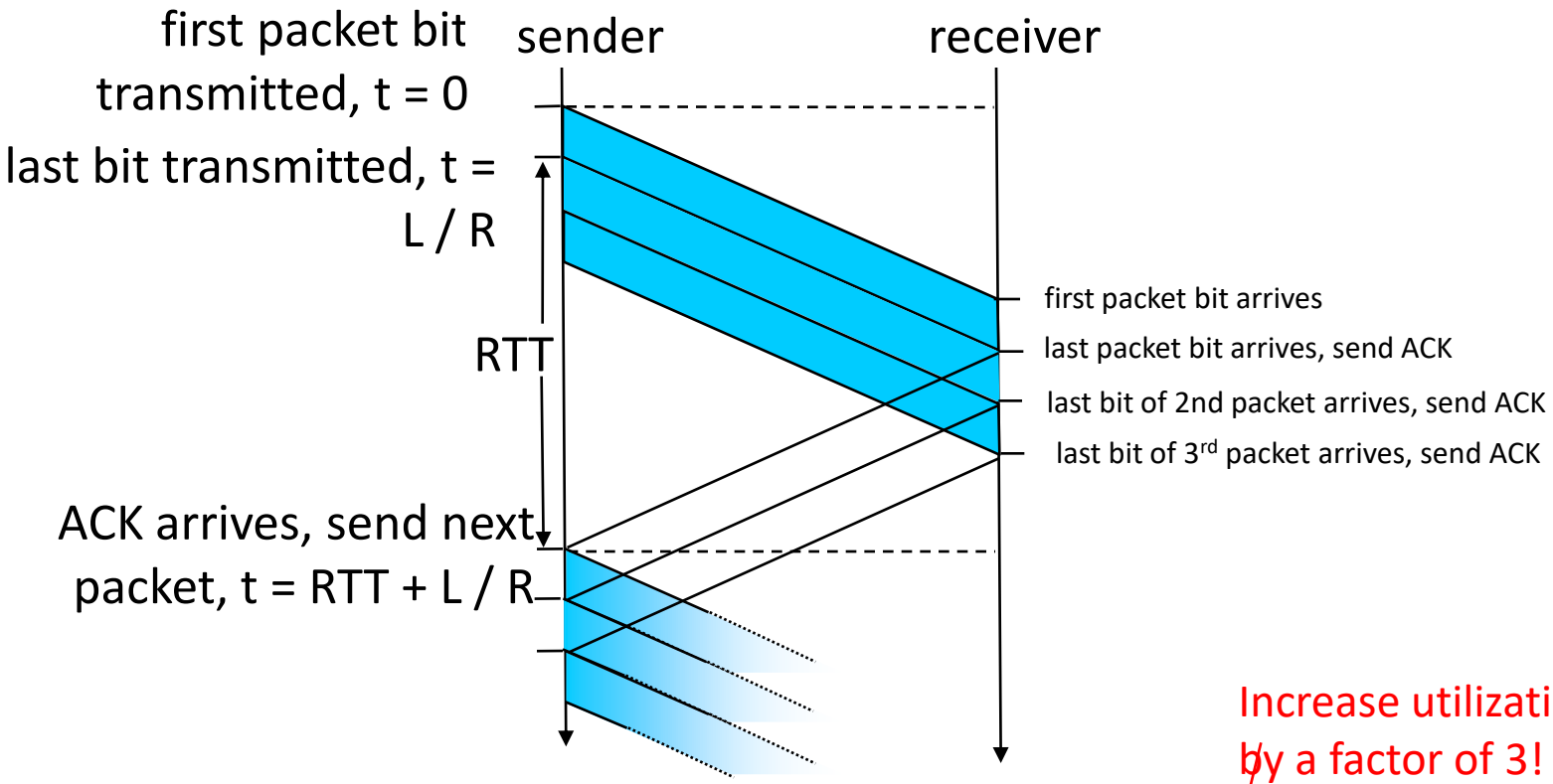




L: packet bit length

R: link bandwidth (bps)

Pipelining: Increased Utilization



Increase utilization by a factor of 3!

$$Utilization = \frac{3*L/R}{(RTT+L/R)}$$

Go-Back-N Automatic Repeat Request

- To improve the efficiency of transmission, multiple frames must be in transition while waiting for acknowledgment.
- In this protocol we can send several frames before receiving acknowledgments; we keep a copy of these frames until the acknowledgments arrive.
- Sequence Number: Frames from a sending station are numbered sequentially.
- However, because we need to include the sequence number of each frame in the header, we need to set a limit.
- If the header of the frame allows m bits for the sequence number, the sequence numbers range from 0 to 2^m - 1.
- In other words, the sequence numbers are modulo-2^m.

Sliding Windows

- The sliding window is an abstract concept that defines the range of sequence numbers that is the concern of the sender and receiver.
- In other words, the sender and receiver need to deal with only part of the possible sequence numbers.
- The range which is the concern of the sender is called the send sliding window; the range that is the concern of the receiver is called the receive sliding window.
- The send window is an imaginary box covering the sequence numbers of the data frames which can be in transit.
- In each window position, some of these sequence numbers

define the frames that have been sent; others define those that can be sent.

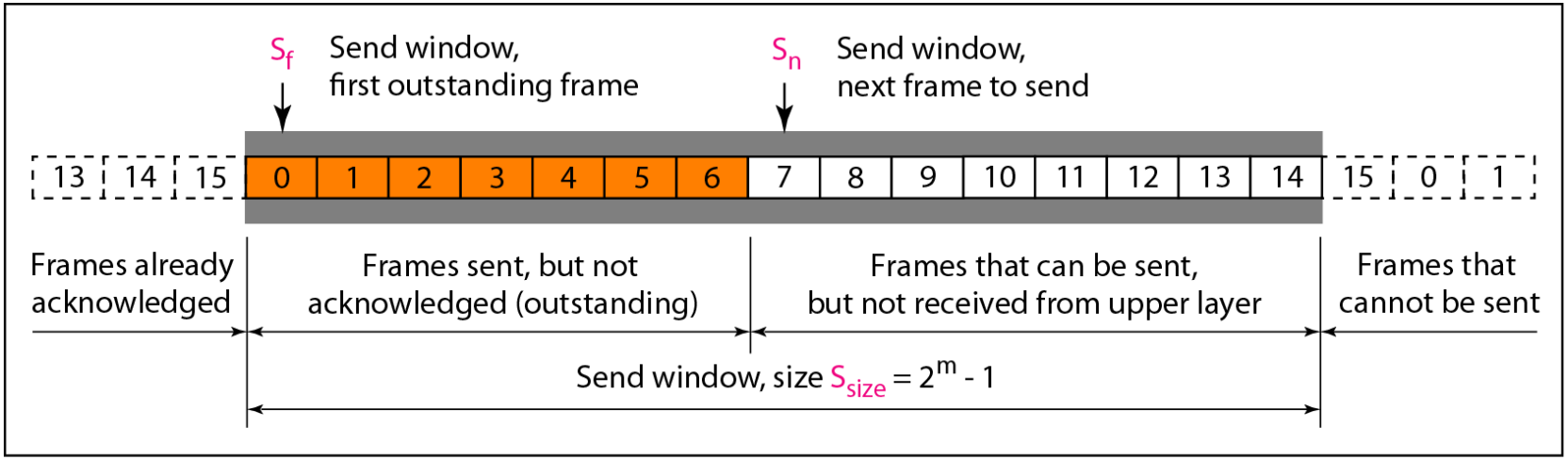
- The maximum size of the window is $2^m - 1$

- The window at any time divides the possible sequence numbers into four regions.
- The first region, from the far left to the left wall of the window, defines the sequence numbers belonging to frames that are already acknowledged. The sender does not worry about these frames and keeps no copies of them.
- The second region, colored in defines the range of sequence numbers belonging to the frames that are sent and have an unknown status.
- The sender needs to wait to find out if these frames have been received or were lost. We call these outstanding frames.
- The third range, white in the figure, defines the range of sequence numbers for frames that can be sent; however, the corresponding data packets have not yet been received from the network layer.
- Finally, the fourth region defines sequence numbers that cannot be used until the window slides, as we see next.

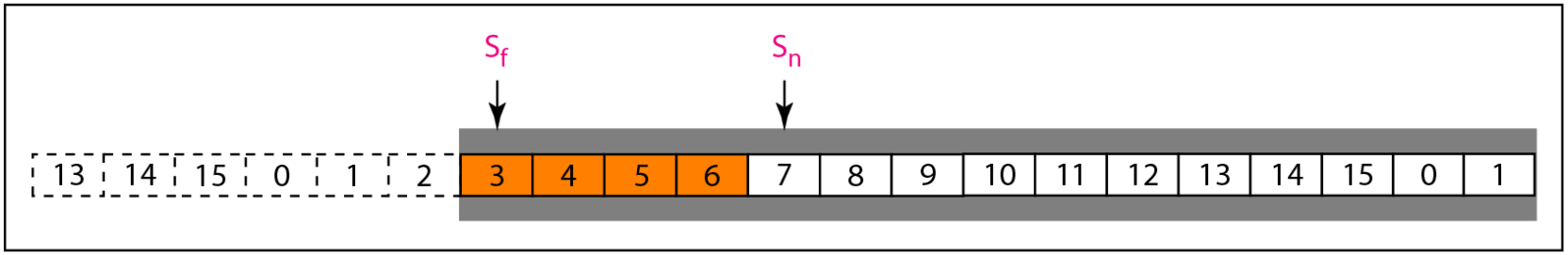
- The window itself is an abstraction; three variables define its size and location at any time.
- We call these variables Sf(send window, the first outstanding frame), Sn (send window, the next frame to be sent), and Ssize (send window, size).
- The variable Sf defines the sequence number of the first (oldest) outstanding frame.
- The variable Sn holds the sequence number that will be assigned to the next frame to be sent.
- Finally, the variable Ssize defines the size of the window, which is fixed in our protocol.

- We need only one variable Rn (receive window, next frame expected) to define this abstraction.
- The sequence numbers to the left of the window belong to the frames already received and acknowledged; the sequence numbers to the right of this window define the frames that cannot be received.
- Any received frame with a sequence number in these two regions is discarded.
- Only a frame with a sequence number matching the value of Rn is accepted and acknowledged.
- The receive window also slides, but only one slot at a time.
- When a correct frame is received (and a frame is received only one at a time), the window slides.

Send Window For Go-Back-N ARQ

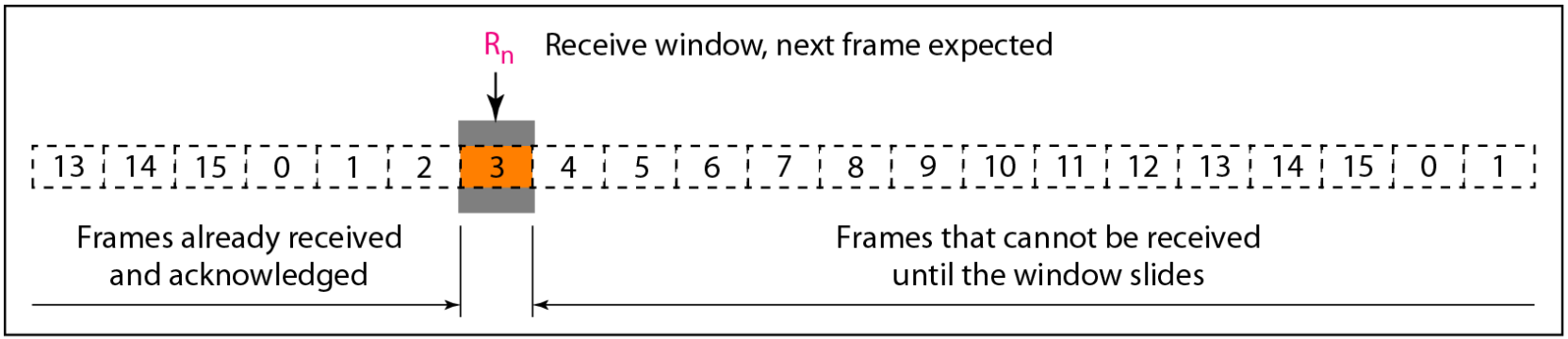


a. Send window before sliding

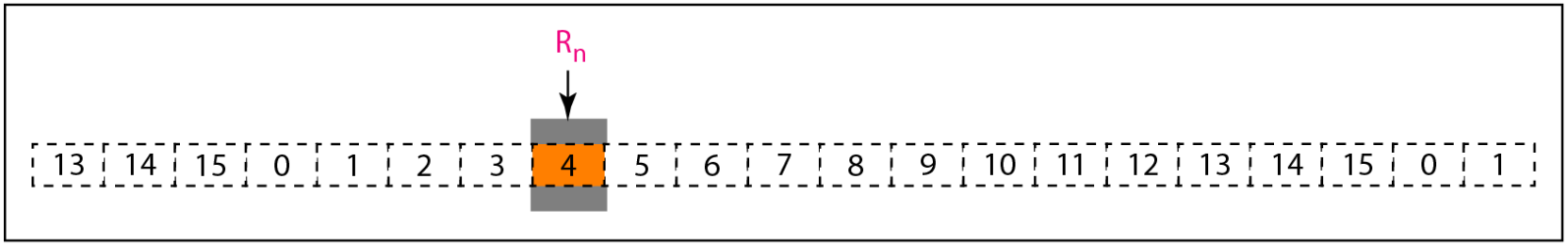


b. Send window after sliding

Send Window For Go-Back-N ARQ

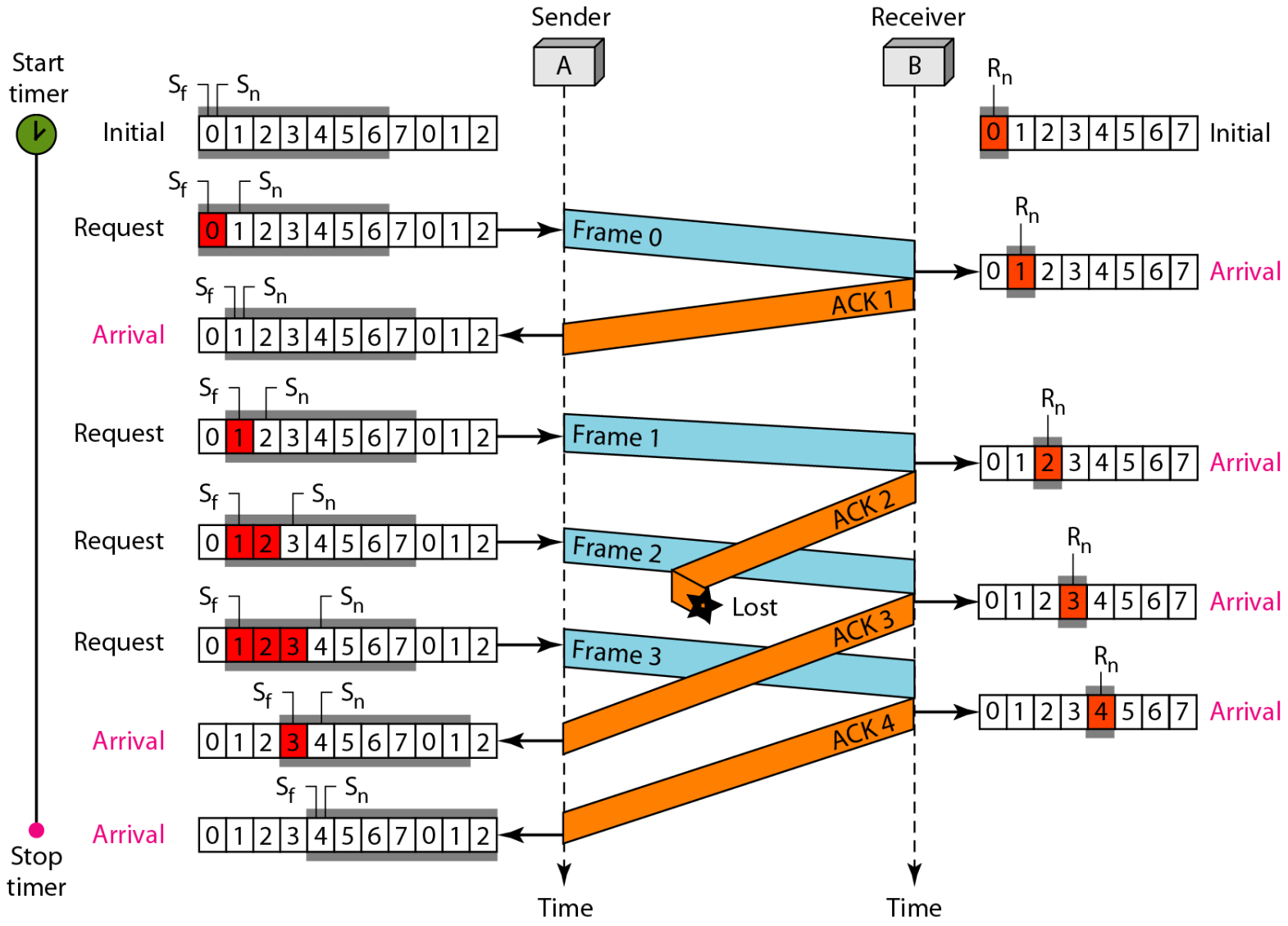


a. Receive window

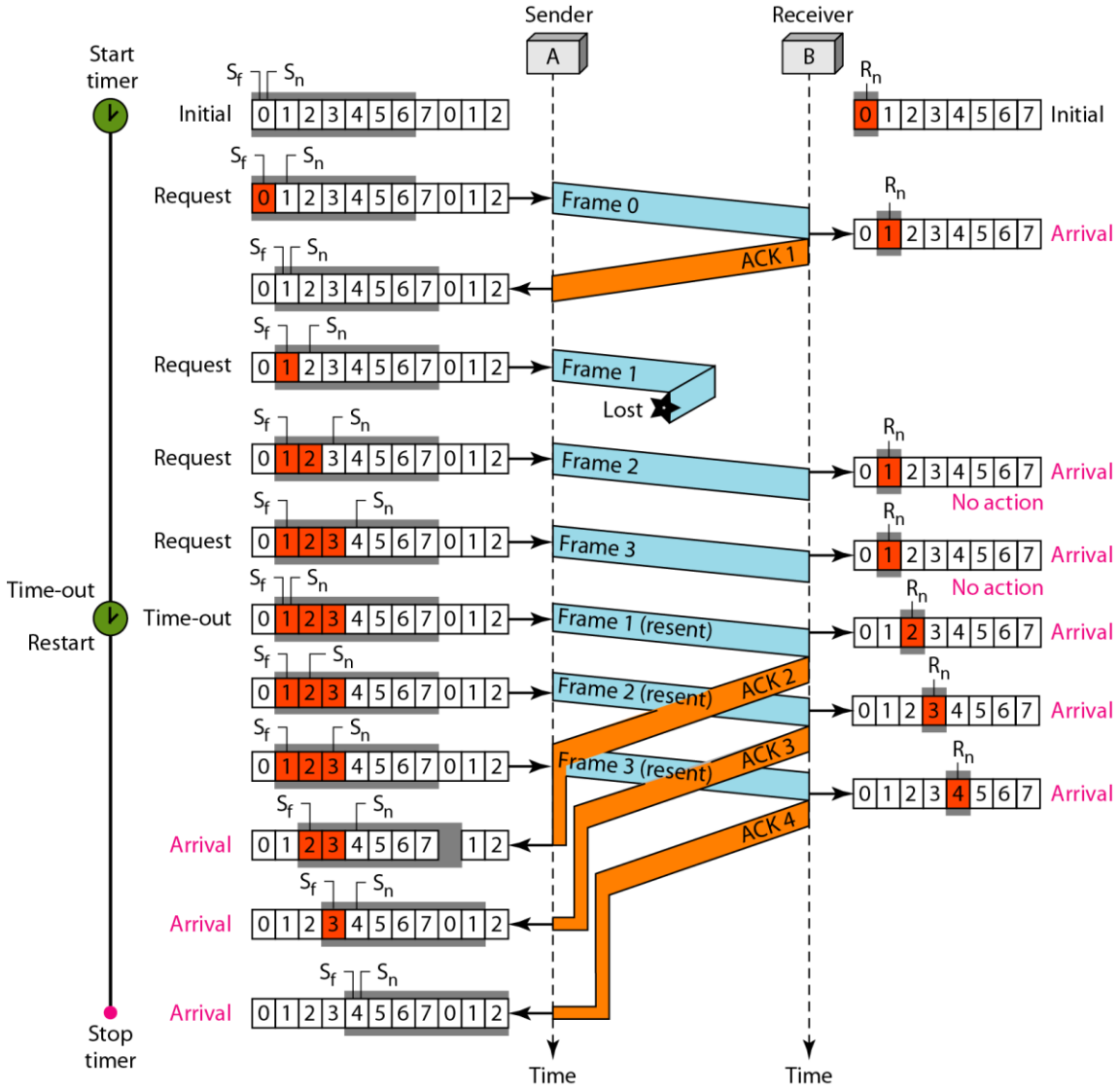


b. Window after sliding

Flow Diagram



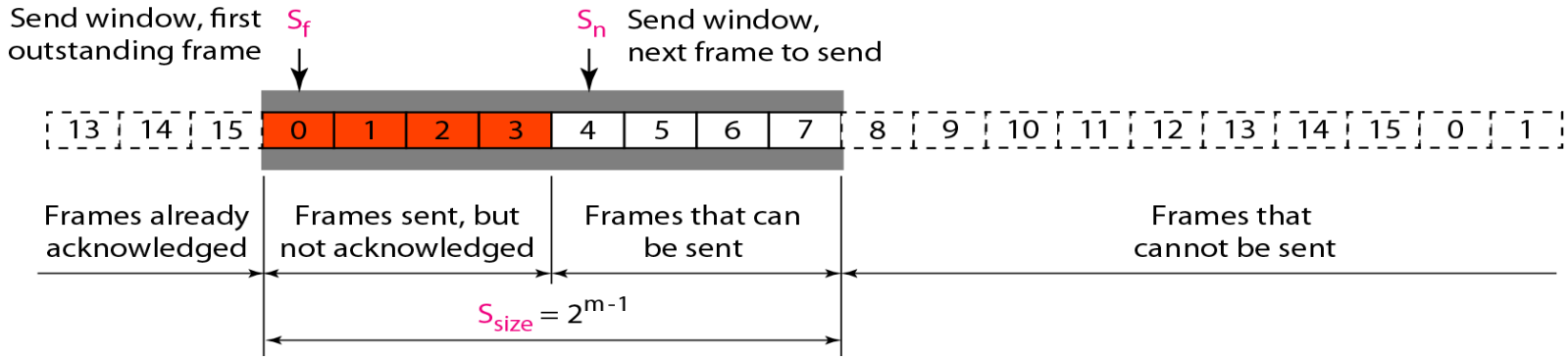
Flow Diagram



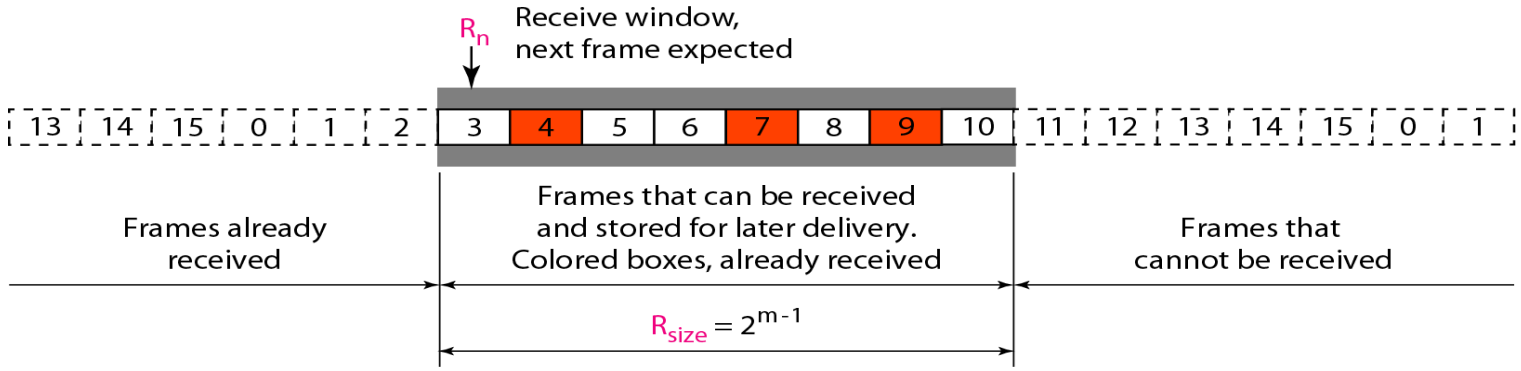
Selective Repeat ARQ

- Problem with Go-back-N:
  - Sender: resend many packets with a single lose
  - Receiver: discard many good received (out-of-order) packets
  - Very inefficient when N becomes bigger (in high-speed network)
- Solution: Receiver *individually* acknowledges all correctly received pkts
  - buffers pkts, as needed, for eventual in-order delivery to upper layer
- sender only resends pkts for which ACK not received
  - sender keeps timer for each unACKed pkt
- sender window
  - N consecutive seq #'s
  - again limits seq #s of sent, unACKed pkts

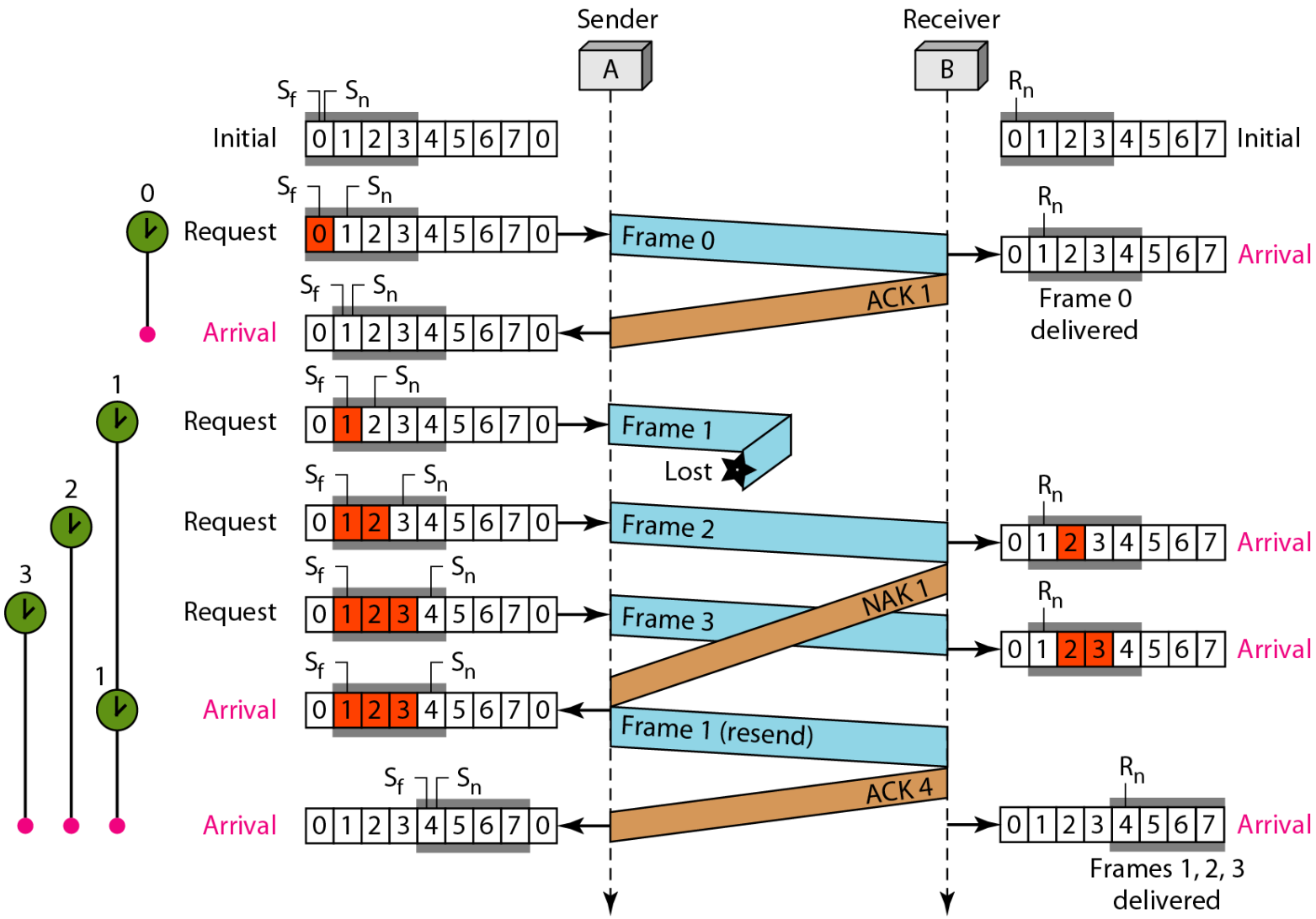
Send window for Selective Repeat ARQ



Receive window for Selective Repeat ARQ



Flow diagram



REFRENCES

- “ DATA COMMUNICATIONS AND NETWORKING ”, Behrouz A. Forouzan And Sophia Chung Fegan , Fourth Edition , McGraw-Hill