DATA LINK LAYER

Ref: Forouzan, Data communications and networking.

By:

Abdul Ghofir, M.Kom.

Data Link Layer

- The two main functions of the data link layer are data link control and media access control.
- **Data link control** functions include framing, flow and error control, and software implemented protocols that provide smooth and reliable transmission of frames between nodes.

Framing

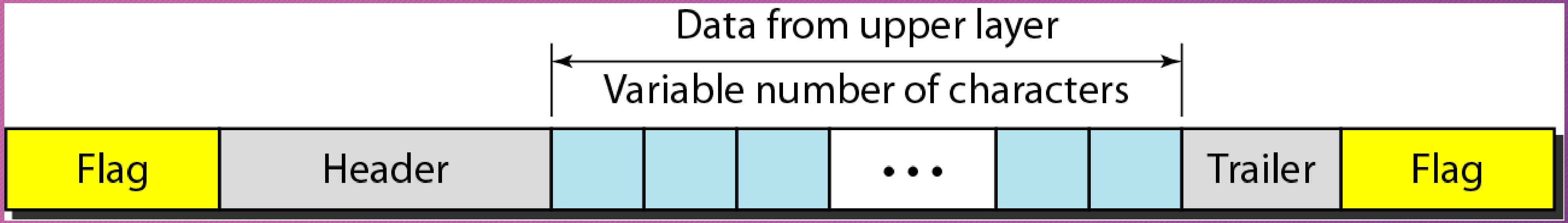
- Data transmission in the physical layer means moving bits in the form of a signal from the source to the destination.
- The data link layer needs to pack bits into frames, so that each frame is distinguishable from another.
- Framing in the data link layer separates a message from one source to a destination, or from other messages to other destinations, by adding a sender address and a destination address.

Framing

1. Fix-Size 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.
   - An example of this type of framing is the ATM wide-area network, which uses frames of fixed size called cells.
2. Variable-Size Framing
   - 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 (1-byte) flag is added at the beginning and the end of a frame.
- The flag, composed of protocol-dependent special characters, signals the start or end of a frame.



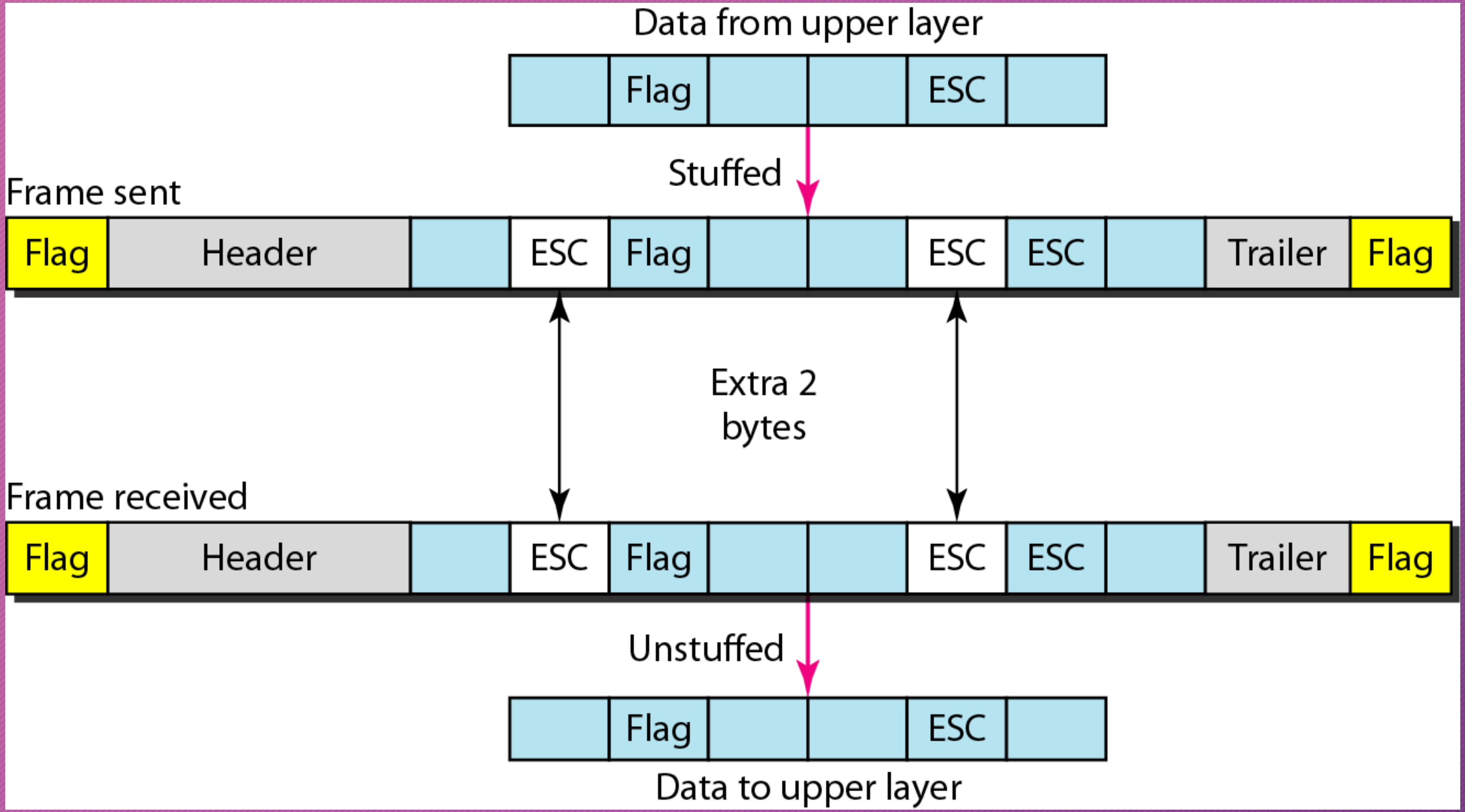
A frame in a character-oriented protocol

Character Oriented protocol

- Character-oriented framing 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.
- Now, however, we send other types of information such as graphs, audio, and video.
- To fix this problem, a byte-stuffing strategy was added to character-oriented framing.
- In byte stuffing (or character 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.

Byte Stuffing

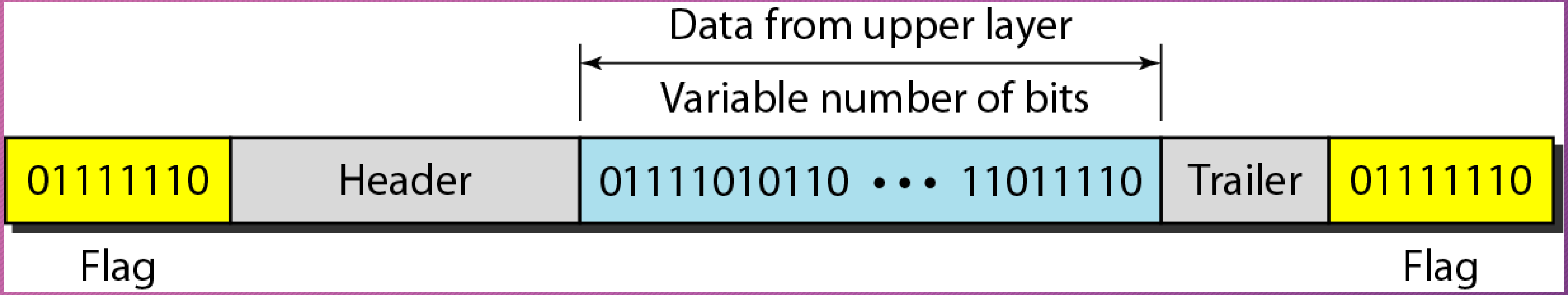
Byte stuffing is the process of adding 1 extra byte whenever there is a flag or escape character in the text.



Byte stuffing and unstuffing

Bit-Oriented Protocols

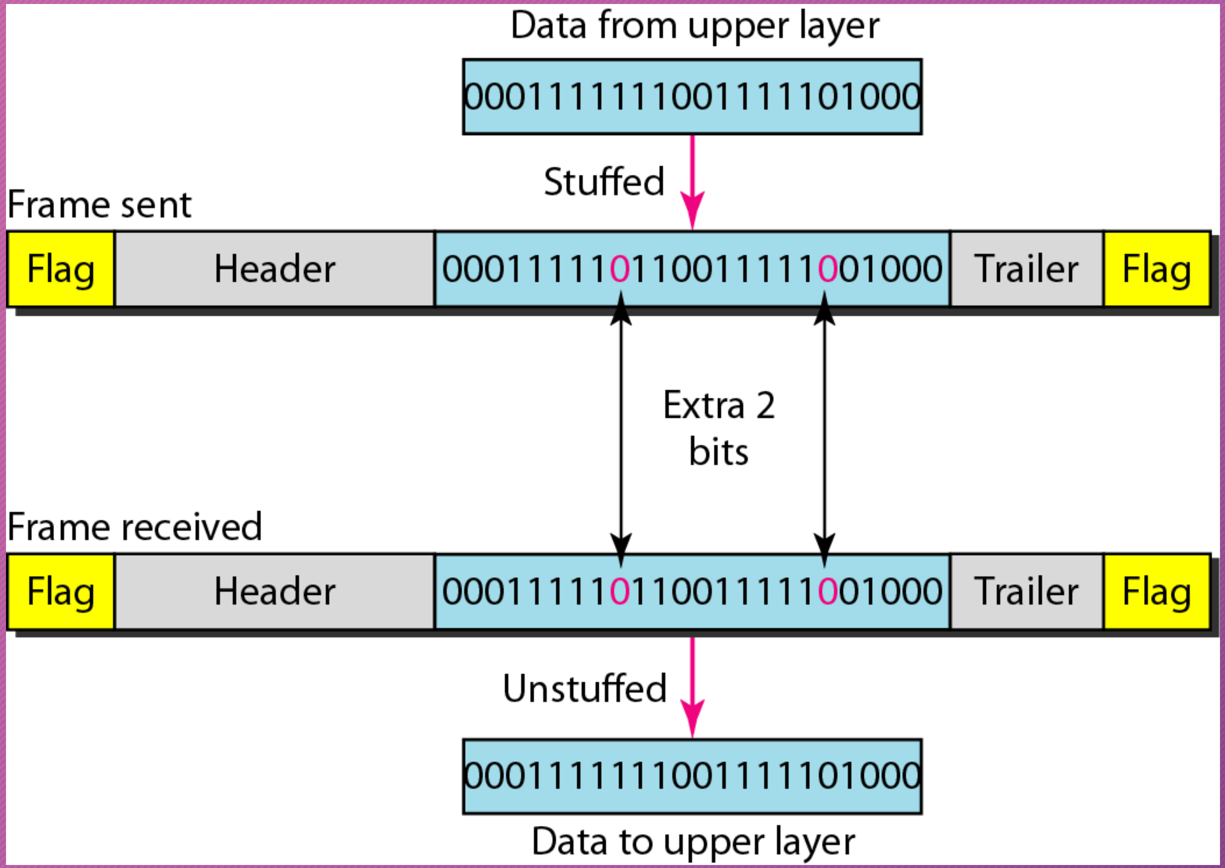
- In a bit-oriented protocol, the data section of a frame is a sequence of bits to be interpreted by the upper layer as text, graphic, audio, video, and so on.
- Most protocols use a special 8-bit pattern flag 01111110 as the delimiter to define the beginning and the end of the frame.



A frame in a bit-oriented protocol

Bit-Oriented Protocols

Bit stuffing is the process of adding one extra 0 whenever five consecutive 1s follow a 0 in the data, so that the receiver does not mistake the pattern 0111110 for a flag.



Bit stuffing and unstuffing

Flow and Error Control

The most important responsibilities of the data link layer are flow control and error control. Collectively, these functions are known as data link 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.

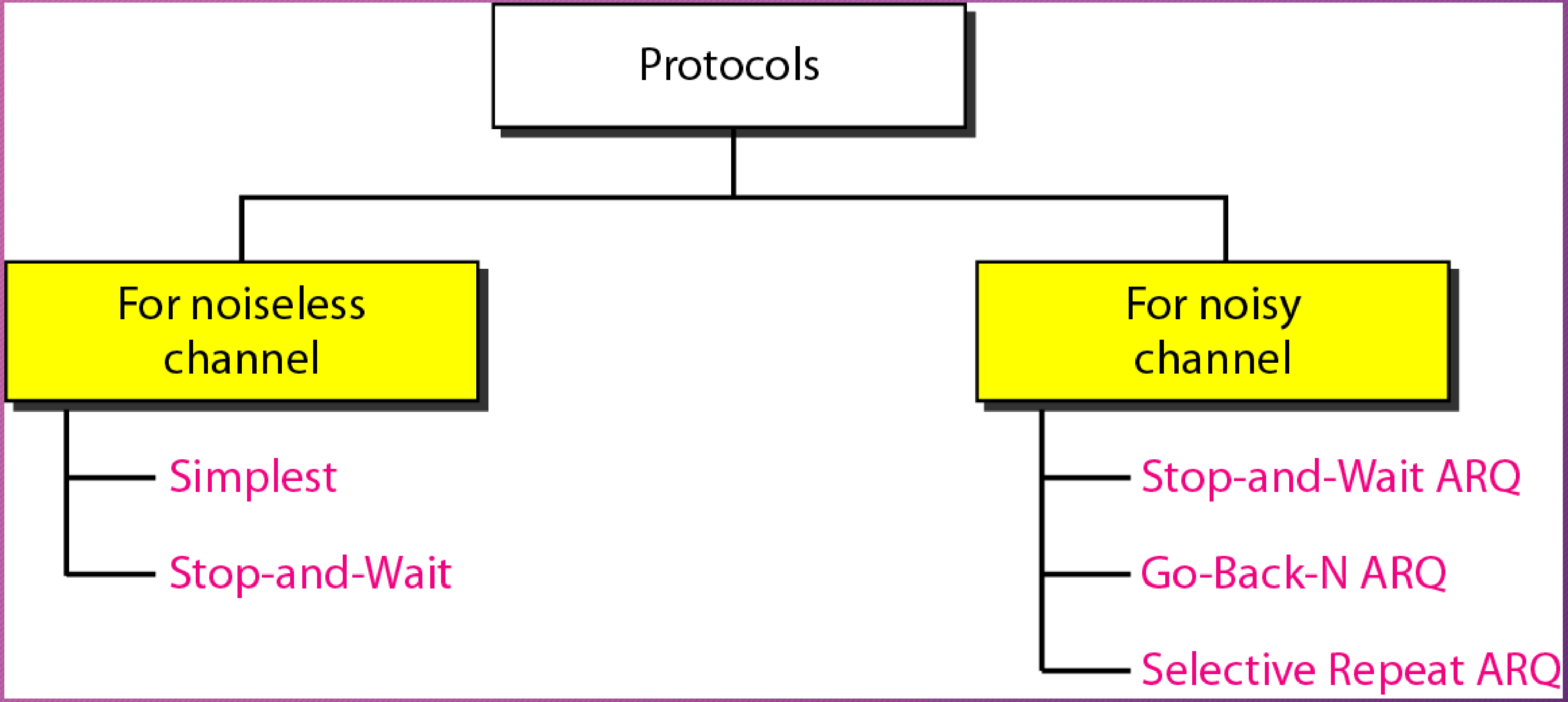
Error Control

Error control in the data link layer is based on automatic repeat request, which is the retransmission of data.

Protocols

- How 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 divided into those that can be used for noiseless (error-free) channels and those that can be used for noisy (error-creating) channels.

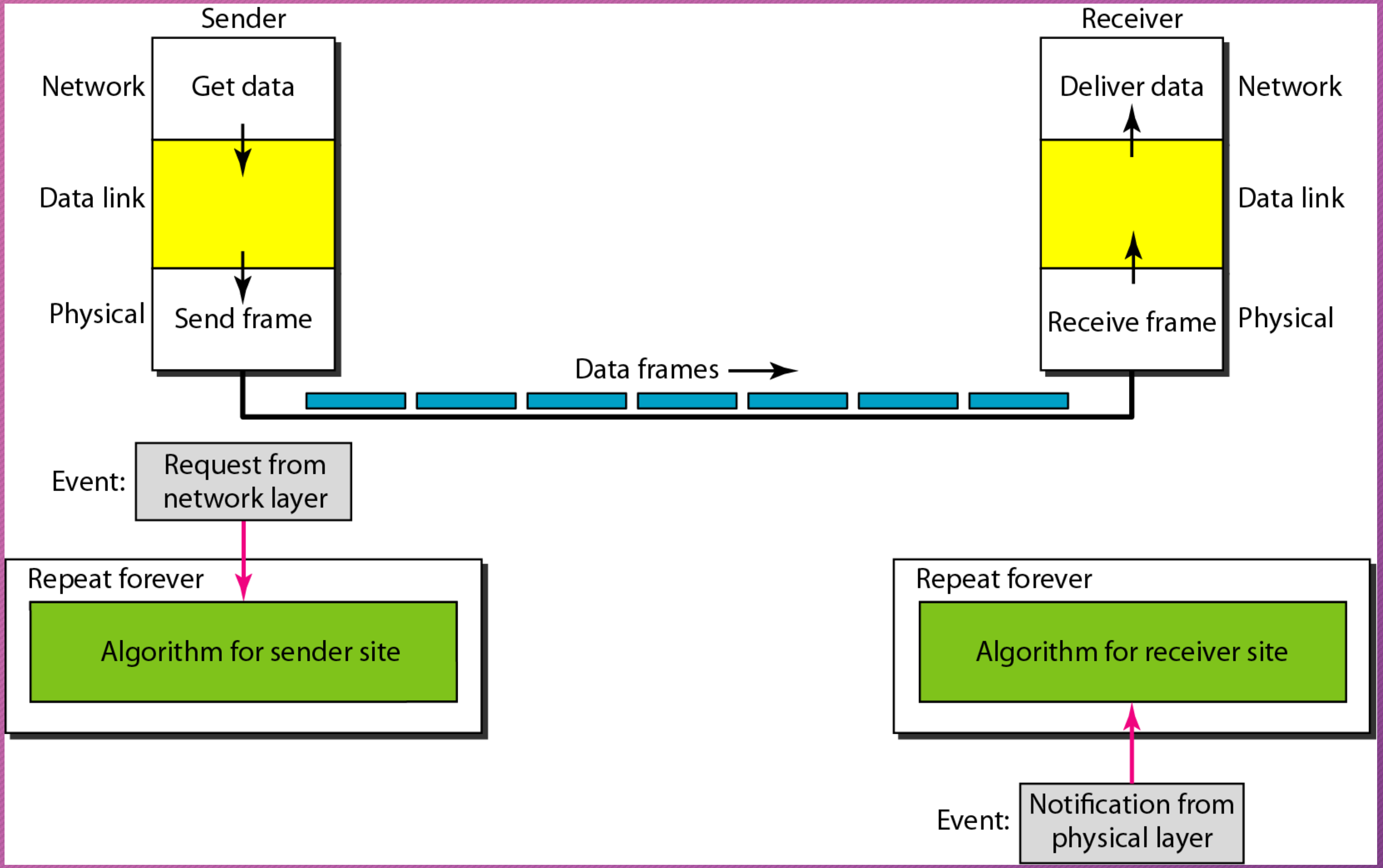
Protocols


Protocols
For noiseless channel
Simplest
Stop-and-Wait
For noisy channel
Stop-and-Wait ARQ
Go-Back-N ARQ
Selective Repeat ARQ

Noiseless Channels

Let us first assume we have an ideal channel in which no frames are lost, duplicated, or corrupted. We introduce two protocols for this type of channel:

- Simplest Protocol
- Stop-and-Wait Protocol

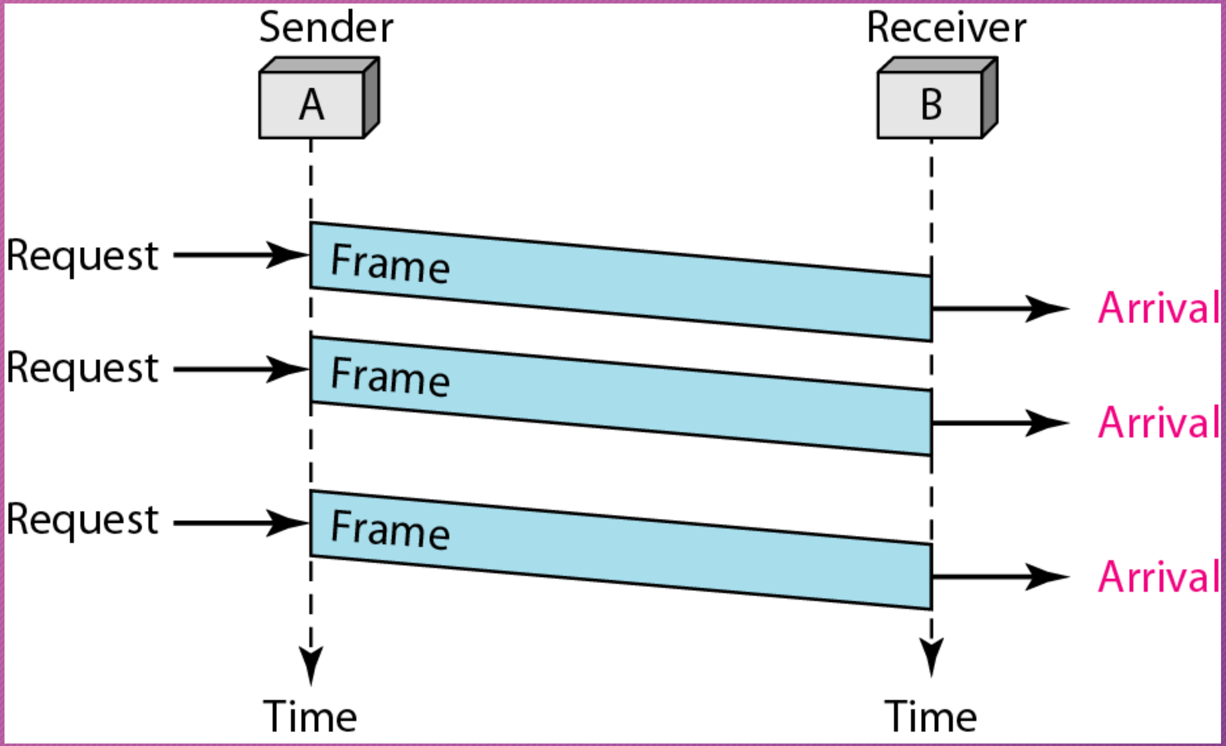


The design of the simplest protocol with no flow or error control

Noiseless Channels
(Simplest Protocol)

Example:

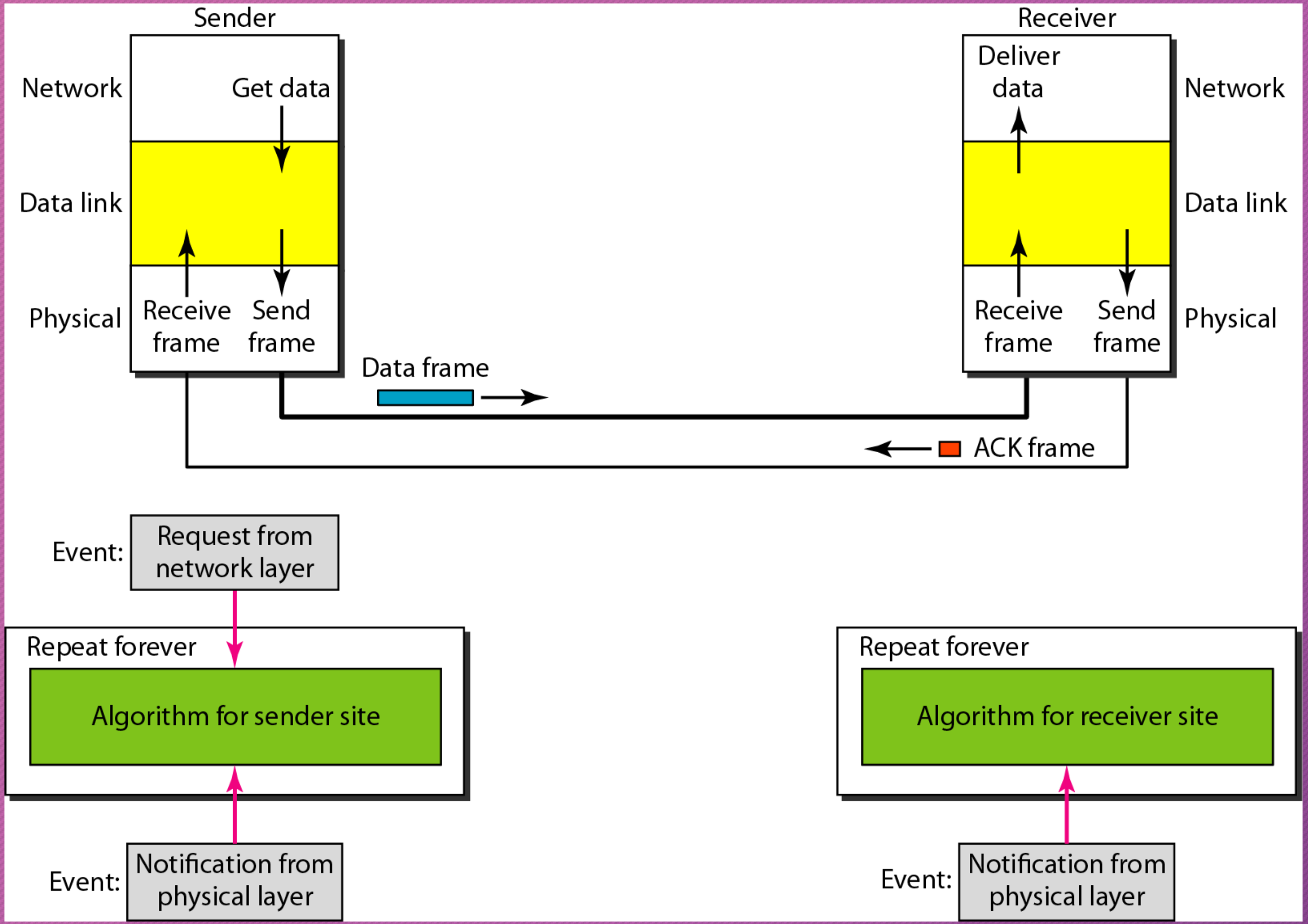
A communication using (simplest) protocol. The sender sends a sequence of frames without even thinking about the receiver. To send three frames, three events occur at the sender site and three events at the receiver site. Note that the data frames are shown by tilted boxes; the height of the box defines the transmission time difference between the first bit and the last bit in the frame.



Flow diagram of the example

Noiseless Channels
(Stop-and-wait Protocol)

- If data frames arrive at the receiver site faster than they can be processed, the frames must be stored until their use.
- Normally, the receiver does not have enough storage space, especially if it is receiving data from many sources.
- To prevent the receiver from becoming overwhelmed with frames, it somehow need to tell the sender to slow down.

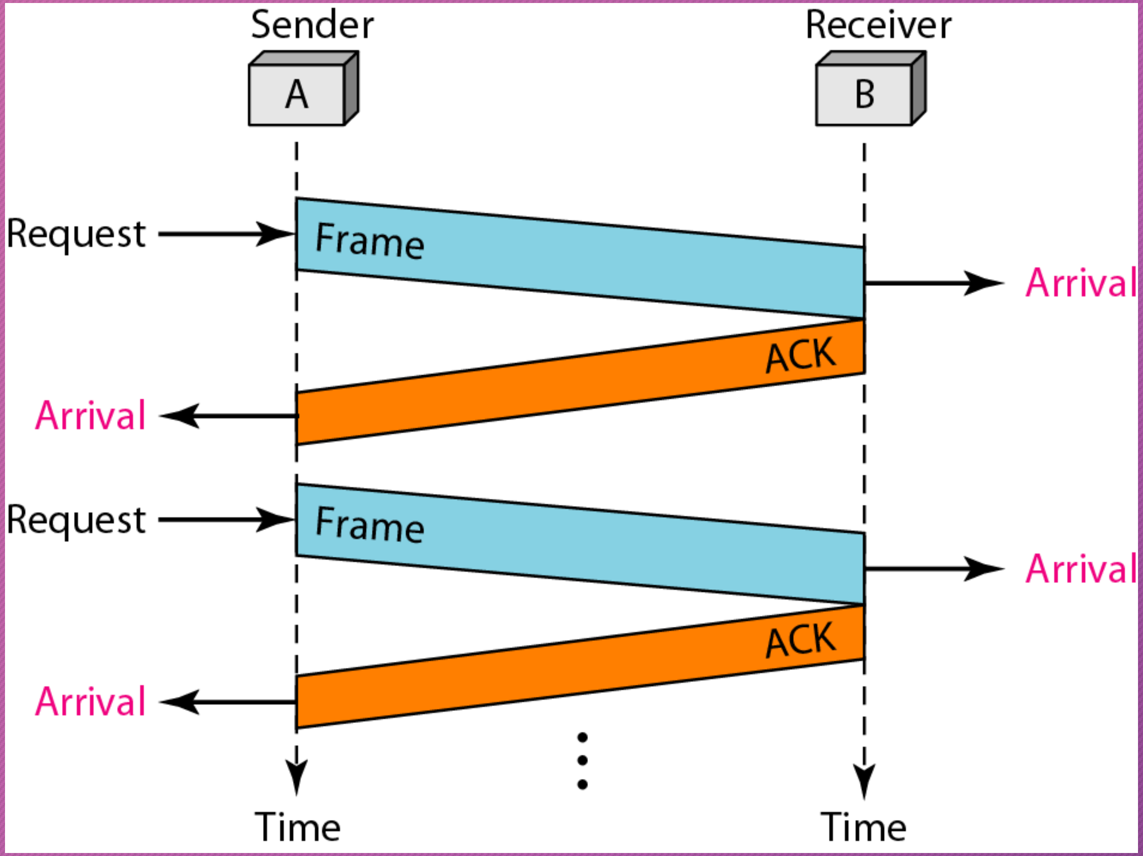


Design of Stop-and-Wait Protocol

Noiseless Channels
(Stop-and-wait Protocol)

Example:

A communication using (stop-and-wait) protocol. The sender sends one frame and waits for feedback from the receiver. When the ACK arrives, the sender sends the next frame. Note that sending two frames in the protocol involves the sender in four events and the receiver in two events.



Flow diagram of example

Noisy Channels

- Although the Stop-and-Wait Protocol gives us an idea of how to add flow control to its predecessor, noiseless channels are nonexistent.
- We can ignore the error (as we sometimes do), or we need to add error control to our protocols.
- There are three protocols in this section that use error control.

Noisy Channels
(Stop-and-Wait Automatic Repeat Request)

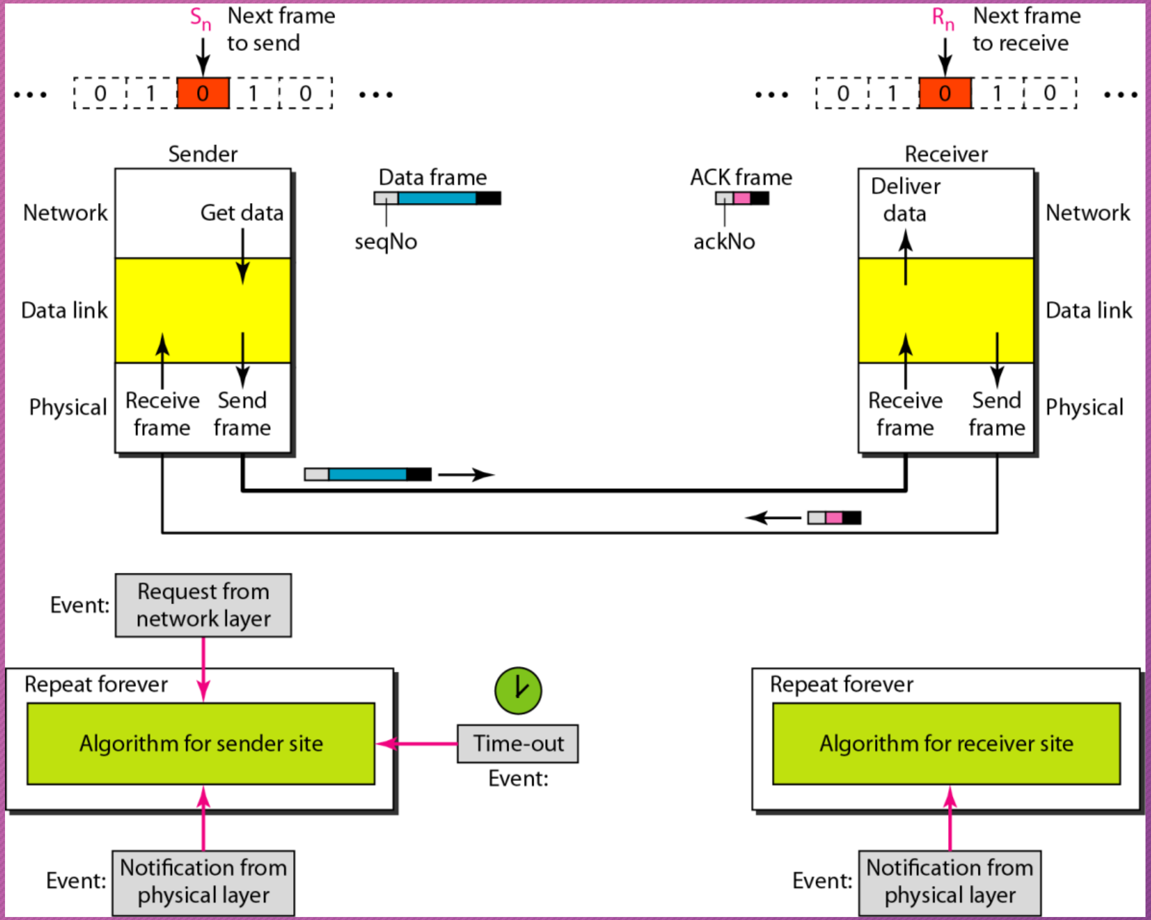
Error correction in Stop-and-Wait ARQ is done by keeping a copy of the sent frame and retransmitting of the frame when the timer expires.

Noisy Channels
(Stop-and-Wait Automatic Repeat Request)

In Stop-and-Wait ARQ, we use sequence numbers to number the frames.
The sequence numbers are based on modulo-2 arithmetic.

Noisy Channels
(Stop-and-Wait Automatic Repeat Request)

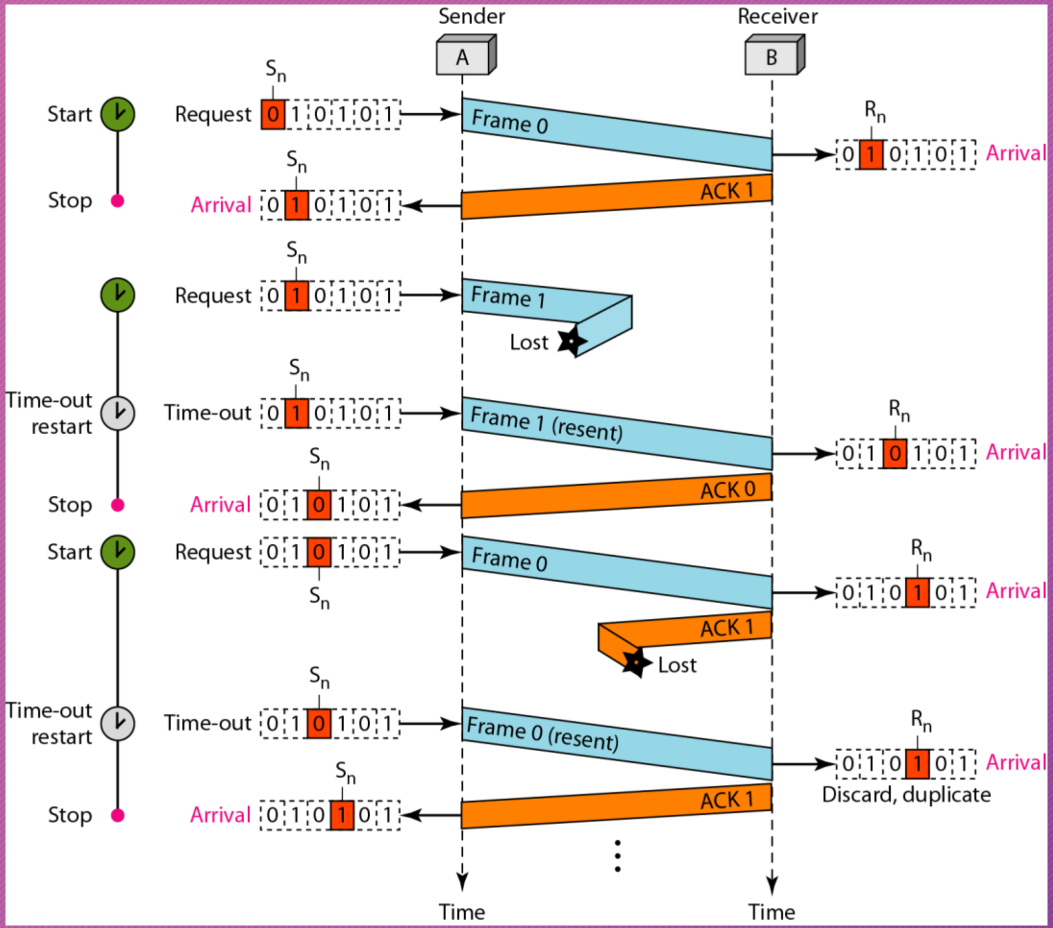
In Stop-and-Wait ARQ, the acknowledgment number always announces in modulo-2 arithmetic the sequence number of the next frame expected.



Design of the Stop-and-Wait ARQ Protocol

Noisy Channels
(Stop-and-Wait Automatic Repeat Request)

Example 1:
Frame 0 is sent and acknowledged. Frame 1 is lost and resent after the time-out. The resent frame 1 is acknowledged and the timer stops. Frame 0 is sent and acknowledged, but the acknowledgment is lost. The sender has no idea if the frame or the acknowledgment is lost, so after the time-out, it resends frame 0, which is acknowledged.



Flow diagram for Stop-and-Wait ARQ

Task 1

1. Frame 0 is sent and acknowledged. Then, frame 1 sent and acknowledged too. Next, frame 0 lost and resent after the time is out. The resent 0 is acknowledged but the acknowledgement is lost. The sender has no idea if the frame or the acknowledgement is lost, so after the time out, it resent frame 0 which is acknowledged. Please create the flow diagram!
2. Frame 1 is sent and acknowledged, but the acknowledgement never come to the sender until the time is out. Then frame 1 is resent and acknowledged. Next, frame 0 is sent and acknowledged. Frame 1 is sent and acknowledged. Frame 0 is lost and resent after the time is out. The resent 0 is acknowledged but the acknowledgement is lost. The sender has no idea if the frame or the acknowledgement is lost, so after the time out, it resent frame 0 which is acknowledged. Please create the flow diagram!