Data Link Layer

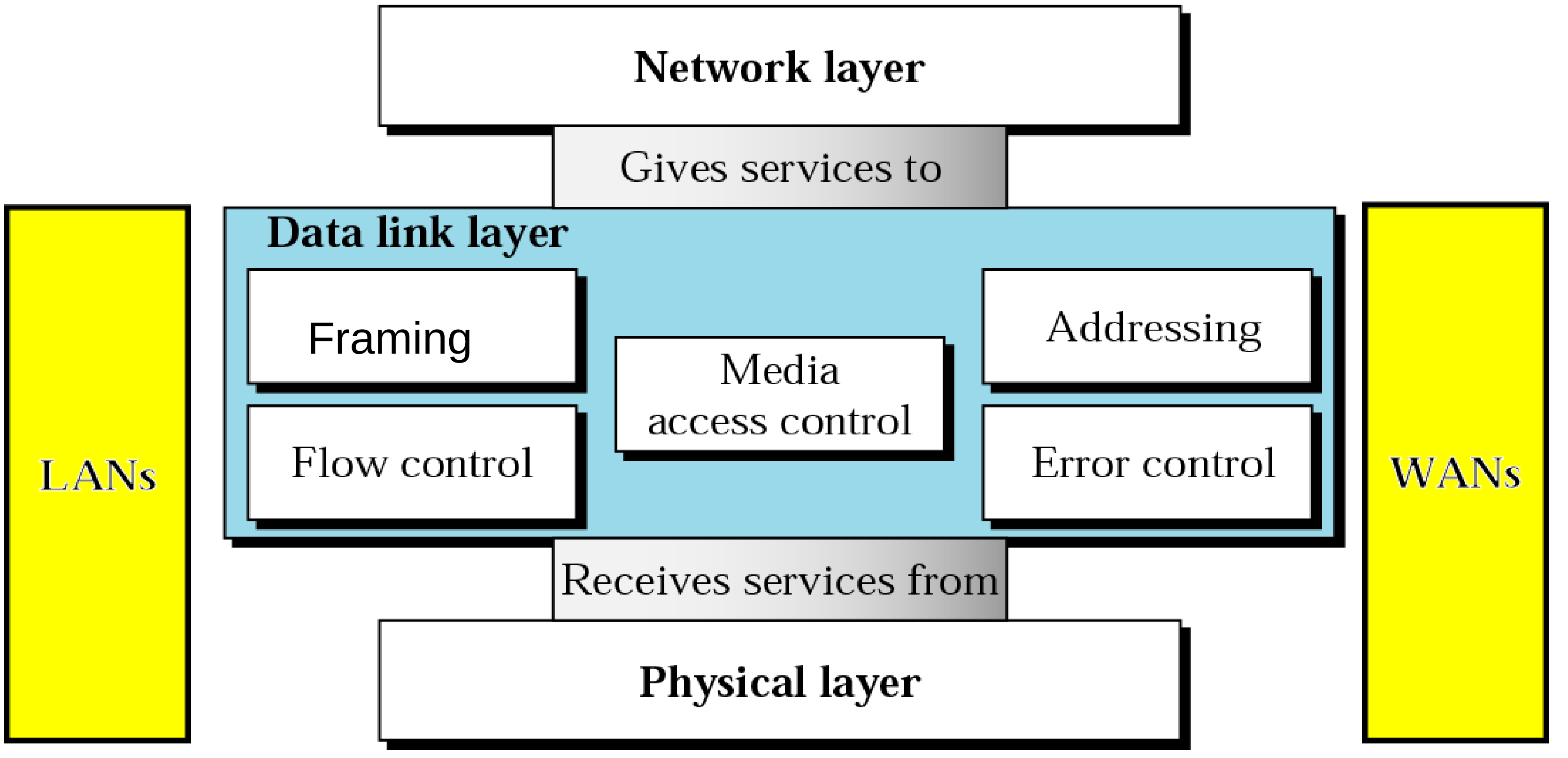
Sachin Gajjar
sachin.gajjar@nirmauni.ac.in

Reading Material for this topic

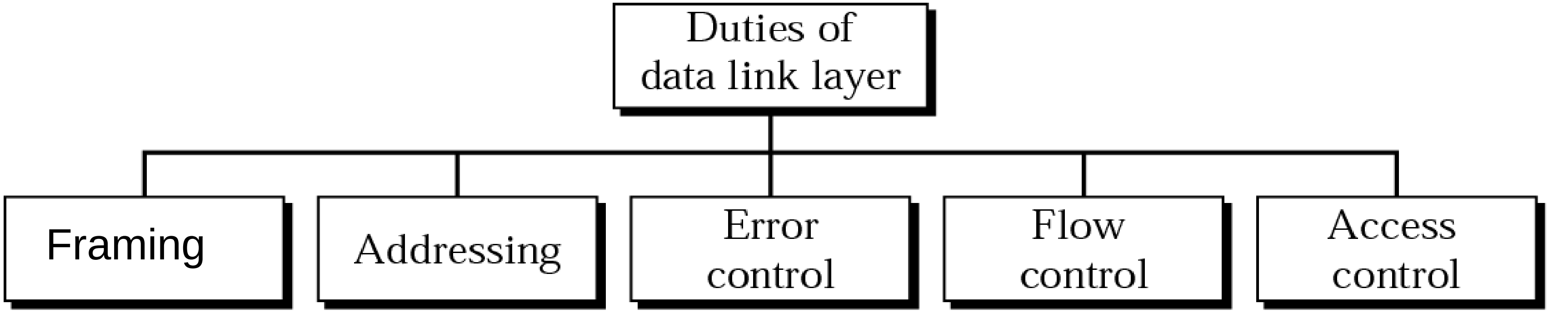
- Computer Networks by Andrew S Tanenbaum
  - Chapter 3 Data Link Layer, Topic: 3.1, 3.3, 3.4

- DATA COMMUNICATIONS AND NETWORKING, Fourth Edition by Behrouz A. Forouzan, Tata McGraw-Hill
  - Chapter 11, Topic 11.3, 11.4, 11.5

Position of the data-link layer



Data link layer functions



Data Link Layer Design Issues

- Providing Services to the Network Layer
- Framing (raw bits at physical layer)
- Error Control
- Flow Control

Services Provided to the Network Layer

- Unacknowledged connectionless service
- Acknowledged connectionless service
- Acknowledged connection-oriented service

Unacknowledged connectionless service

- No logical connection established
- Send frames. No ack by rx
- If lost, no attempt to recover by this layer
- Recovery is left up to the higher layers (transport)
- Good for reliable lines such as fiber,
- for real time where late is worse than lost frames
- Used in Ethernet LAN

Acknowledged connectionless service

- No connection established
- Frames are sent and each is individually ack
- Frames are resent if no ack within time period
- Useful for unreliable channels such as wireless eg. 802.11 (WiFi)
- ACK = optimization but overhead

Acknowledged connection-oriented service

- Most sophisticated
- Reliable service
- 3 phases : Connection establishment/data transfer/connection release
- All frames are received in the right order
- Used in long, unreliable links like satellite channel or a long-distance telephone circuit

Framing

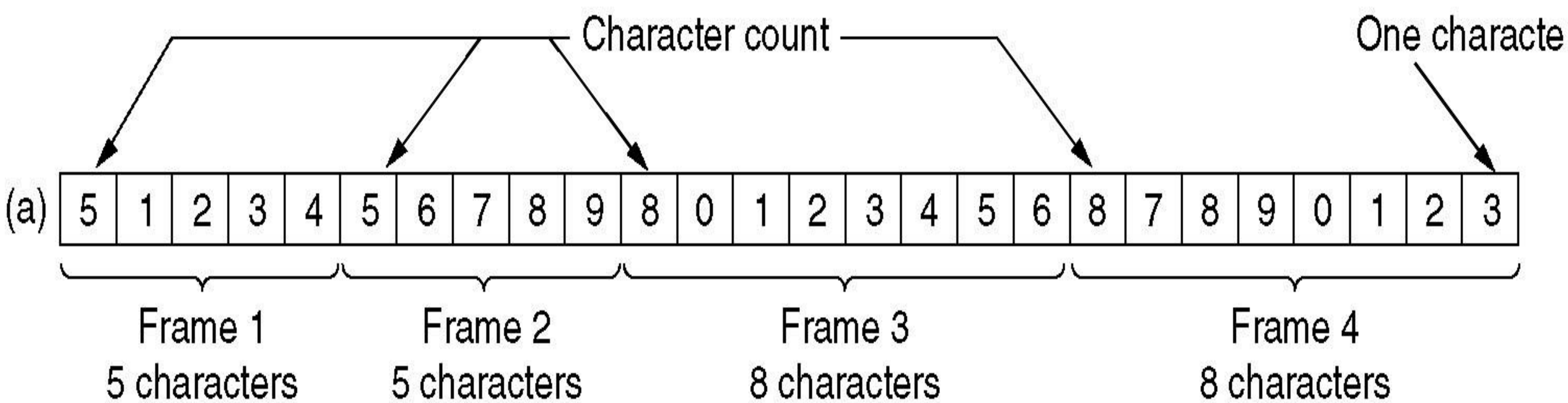
- Network layer gives packet, Physical layer wants bit stream
- physical layer only accept a raw bit stream and deliver to destination
- For long bit stream if error occurs during transmission entire stream has to be sent again.
- Therefore, it is broken to smaller frames
- Use parity/checksum with frames for error detection
- If error detected by Receiver, discard frame or send error report

Framing Methods

- Breaking bit stream into frames
- Receiver to find the start of new frames while using little of the channel bandwidth
  - Character count.
  - Flag bytes with byte stuffing.
  - Starting and ending flags, with bit stuffing.

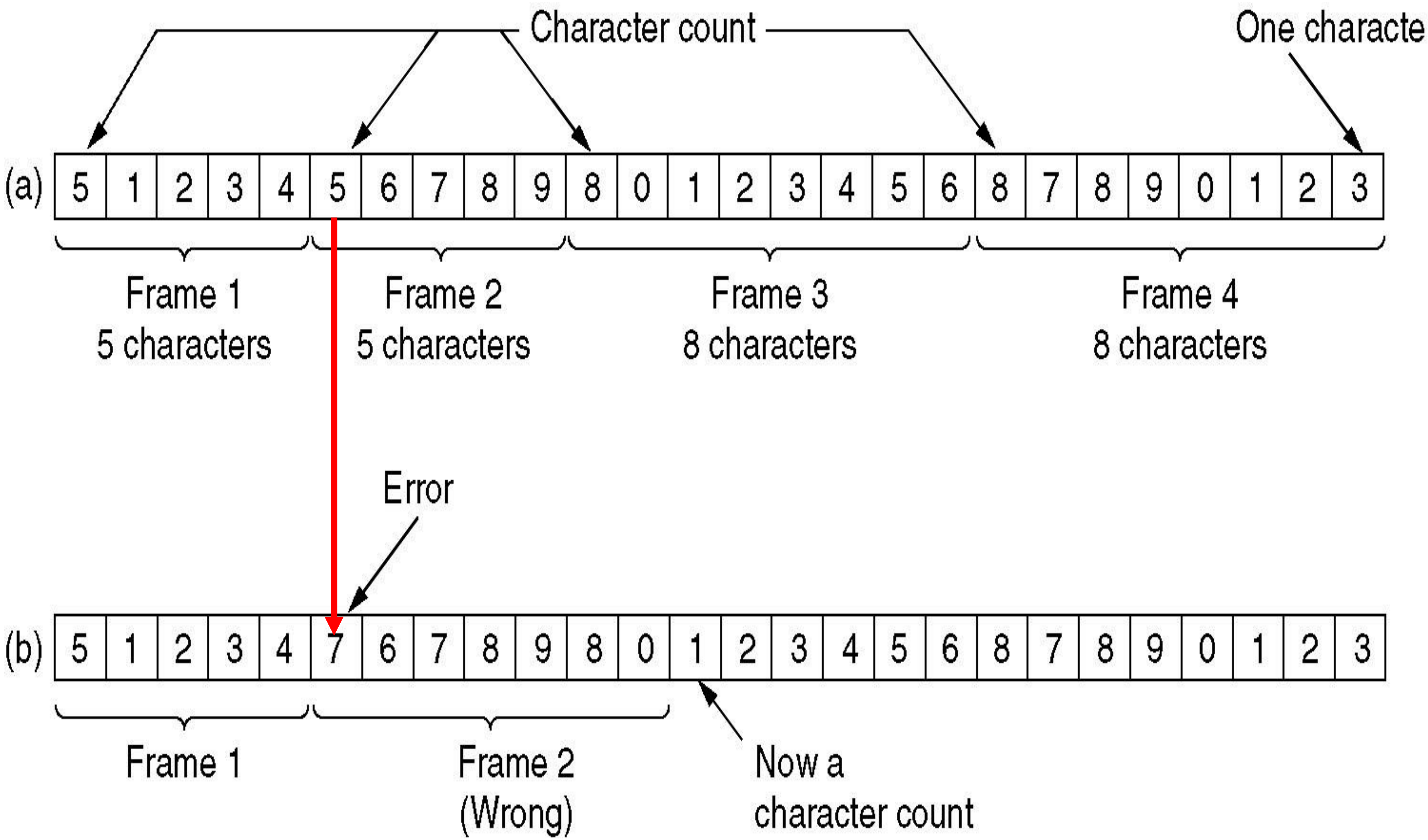
Character count

- First byte (header) indicates how many characters are in the frame including the first byte.
- Rx data link layer sees and knows character to follow



Character count

(a) Without errors. (b) With one error.



Problems with Character Count

- If first byte of any frame gets corrupted, rx goes out of sequence, has no way to find where beginning of next frame is.
- Checksum may not match, but from which character to ask retransmit of the frame from
- This method is rarely used any more

Flag bytes with byte stuffing

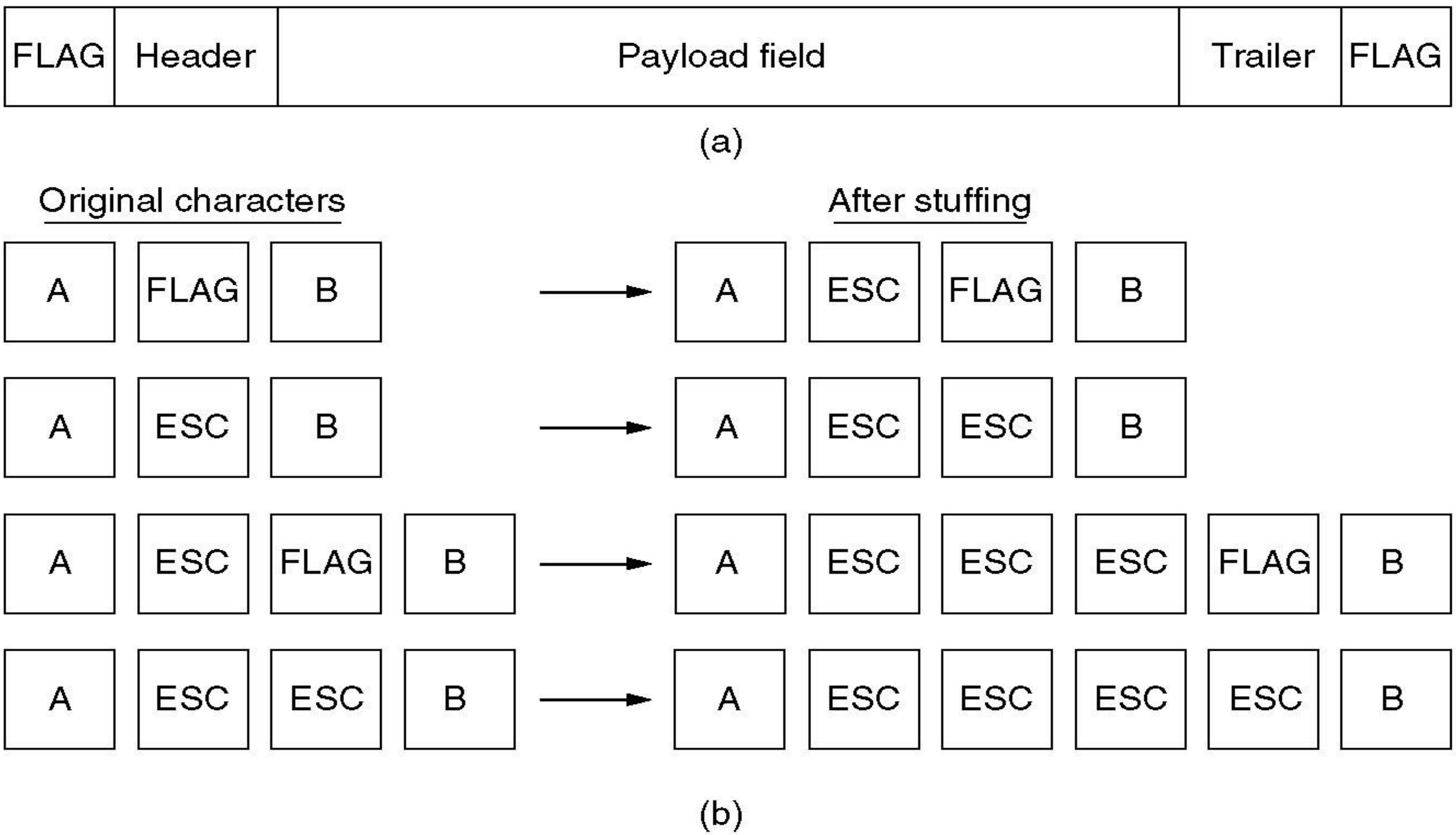
- gets around problem of resynchronization after an error
- has each frame start and end with special bytes
- same flag byte = starting and ending delimiter
- If rx ever loses synchronization, it can search for flag byte to find the end of the current frame
- Two consecutive flag bytes indicate the end of one frame and start of the next one.
- used in PPP (Point-to-Point Protocol)

FLAG	Header	Payload field	Trailer	FLAG

Problem and Solution

- flag byte's bit pattern occurs in the data
- tx's data link layer insert escape byte (ESC) just before each ''accidental'' flag byte in data.
- Rx’s data link layer removes escape byte before the data are given to the network layer
- This is byte stuffing or character stuffing.
- If ESC in data put extra ESC
- One ESC = ESC stuffing
- Two ESC = ESC in data

Flag bytes with byte stuffing



(a) A frame delimited by flag bytes.

(b) Four examples of byte sequences before and after stuffing.

Problem but no solution

- Tied to 8 bit characters – bytes
- If UNICODE character coding is used it requires 16 bit characters
- The data representation has to be in multiples of number of bits of Flag, ESC
- Suppose ESC=1001 Flag=0110, then my data has to be in multiples of 4 bits

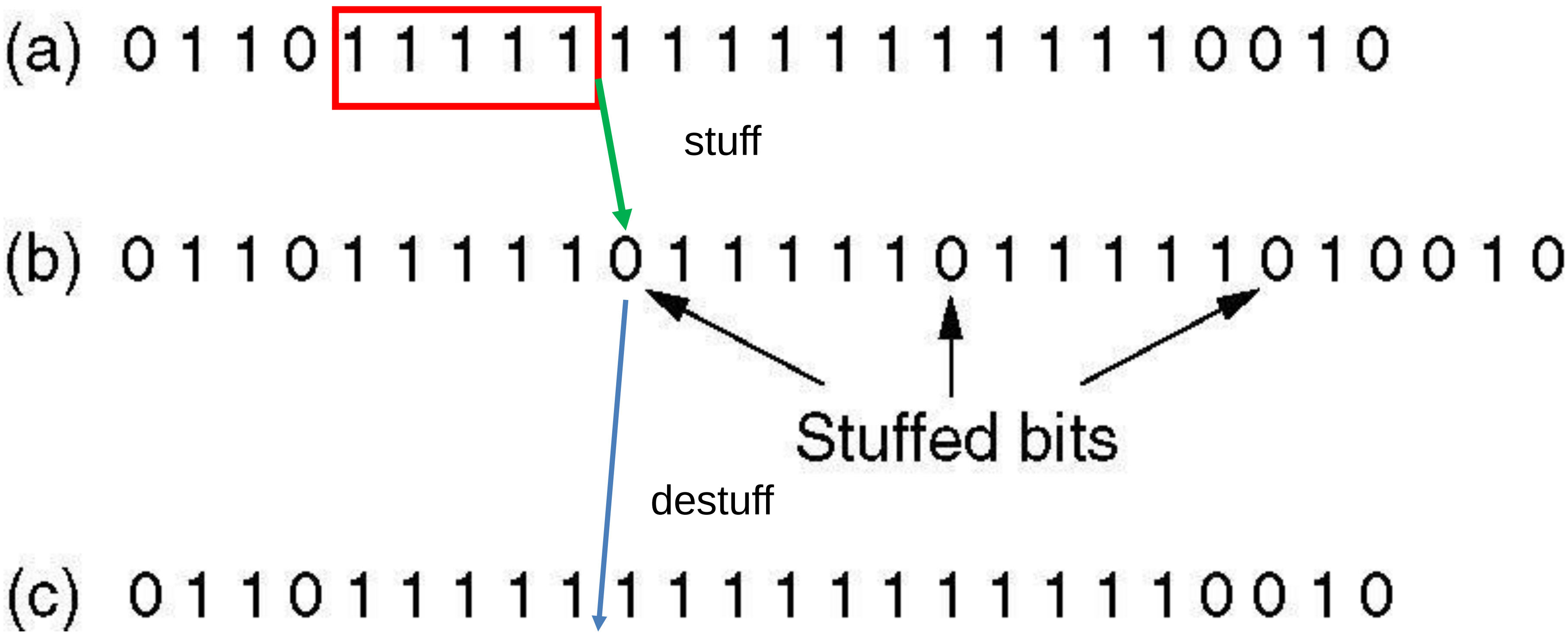
Bit Framing and Stuffing

- allows data frames to contain an arbitrary number of bits
- allows character codes with an arbitrary number of bits per character.
- Each frame begins/ends with a special bit pattern, 01111110 (flag byte)
- Used in HDLC (Highlevel Data Link Control)
- USB (Universal Serial Bus) uses bit stuffing
- Eg. 01111110-1010110100-01111110

Bit Framing and Stuffing

- When sender's data link layer encounters five consecutive 1s in data, it automatically stuffs a 0 bit into outgoing bit stream.
- This bit stuffing is analogous to byte stuffing

Bit Framing and Stuffing



Bit stuffing

(a) The original data.

(b) The data as they appear on the line.

(c) The data as they are stored in receiver’s memory after destuffing.

Bit Framing and Stuffing

- Boundary between two frames can be unambiguously recognized by flag pattern
- If receiver loses track of where it is, scan input for flag sequence since they can only occur at frame boundaries and never within data

Final note on framing

- use combination of character count with one of the other methods for extra safety.
- count field is used to locate the end of the frame.
- if at that position there is delimiter and checksum is correct the frame is accepted as valid.
- Otherwise, the input stream is scanned for the next delimiter.

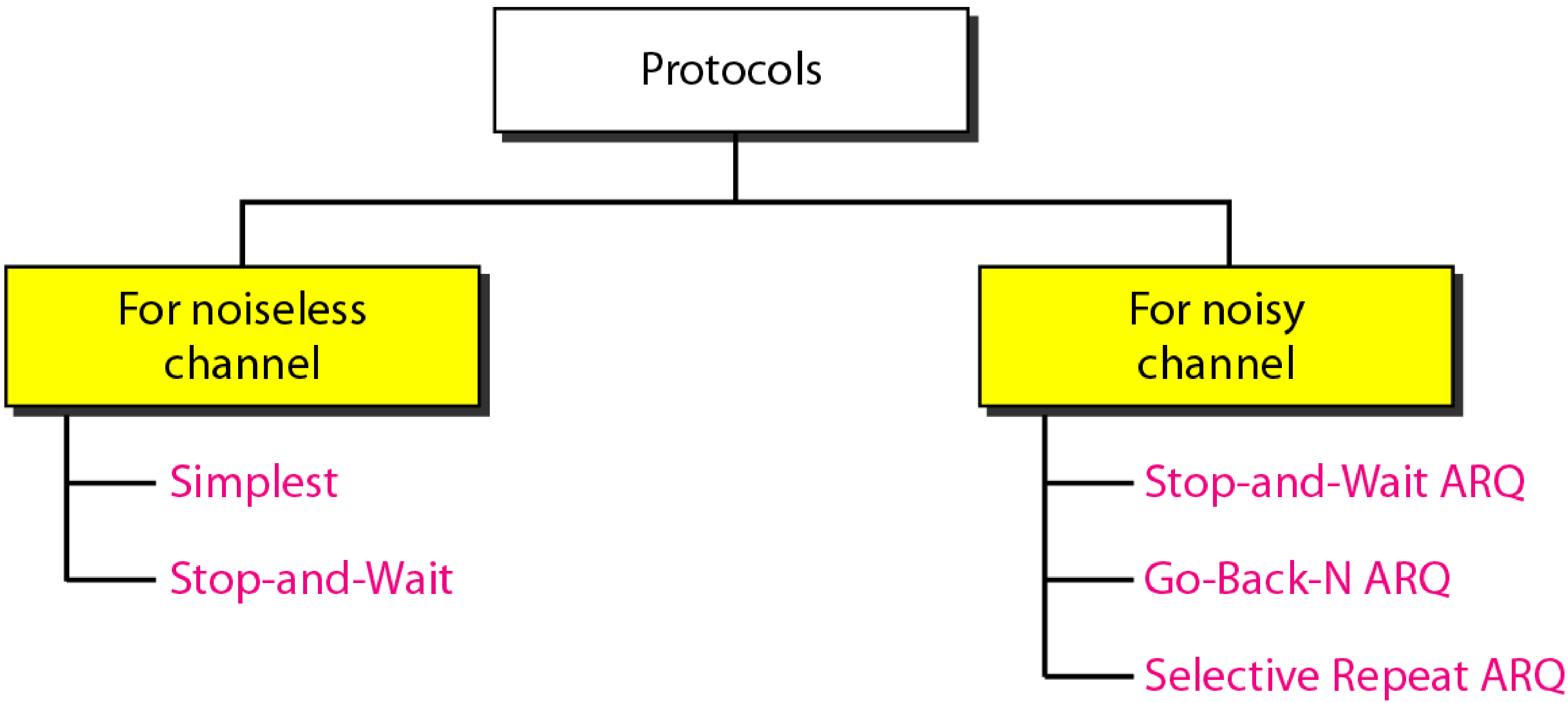
Error Control

- error detection and error correction.
- Allows rx to inform tx of frames lost/damaged in transmission
- Coordinates retransmission of those frames by tx
- In data link layer error detection and retransmission.
- Any time an error is detected by receiver, specified frames are retransmitted.
- This is called automatic repeat request (ARQ).

Flow Control

- Fast sender – slow receiver
- feedback-based flow control,
  - rx sends information to sender giving it permission to send more data or its condition
- rate-based flow control,
  - protocol has a built-in mechanism that limits rate at which senders may transmit data,
  - without using feedback from receiver
  - rate-based schemes are never used in data link layer

DATA LINK LAYER PROTOCOLS



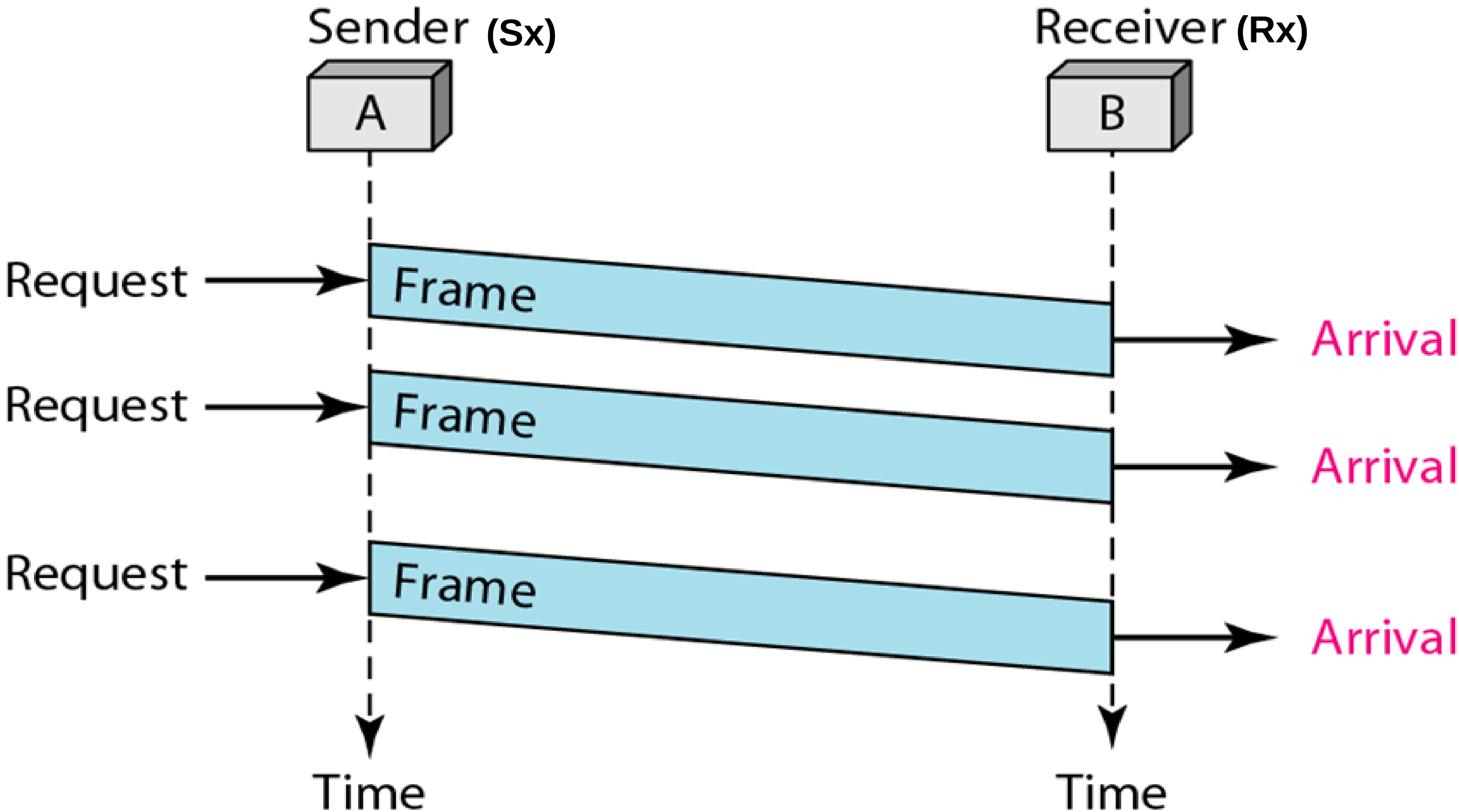
Piggybacking

- In a real network data link protocols are bidirectional
- flow and error control information such as ACKs and NAKs is included in the data frames

Simplest Protocol

- Data = unidirectional
- Both tx/rx always ready with Infinite buffer space
- So no flow control
- Processing time is ignored
- Channel is best, no noise
- No frame damage or lost
- So no error control
- May be used to compare performance of other algorithm

Simplest Protocol – Implemented at Data Link Layer

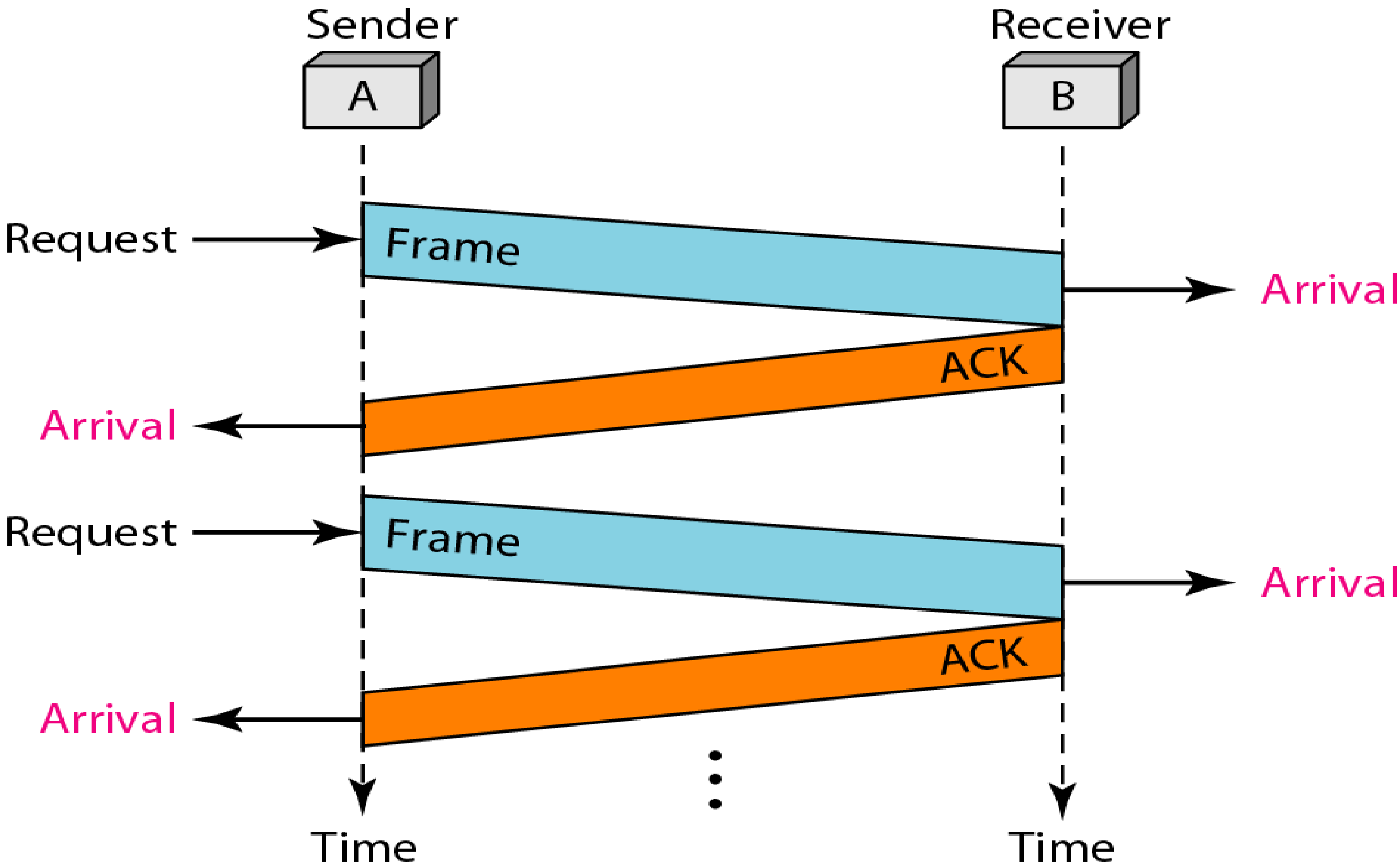


- Sx side DLL procedure -- gets request to send frame from NW layer
- Rx side DLL procedure -- gets notification of arrival of frame from PHY layer

Simplex Stop-and-Wait Protocol

- If it gets frames at rate faster than it can process than frames must be stored in buffer
- Receiver has limited buffer
- Must have flow control, so feedback from rx
- Data = unidirectional
- Only ACK from rx
- Sx sends frame, waits for ack
- If ack comes than only send next one
- Half duplex link between sx/rx

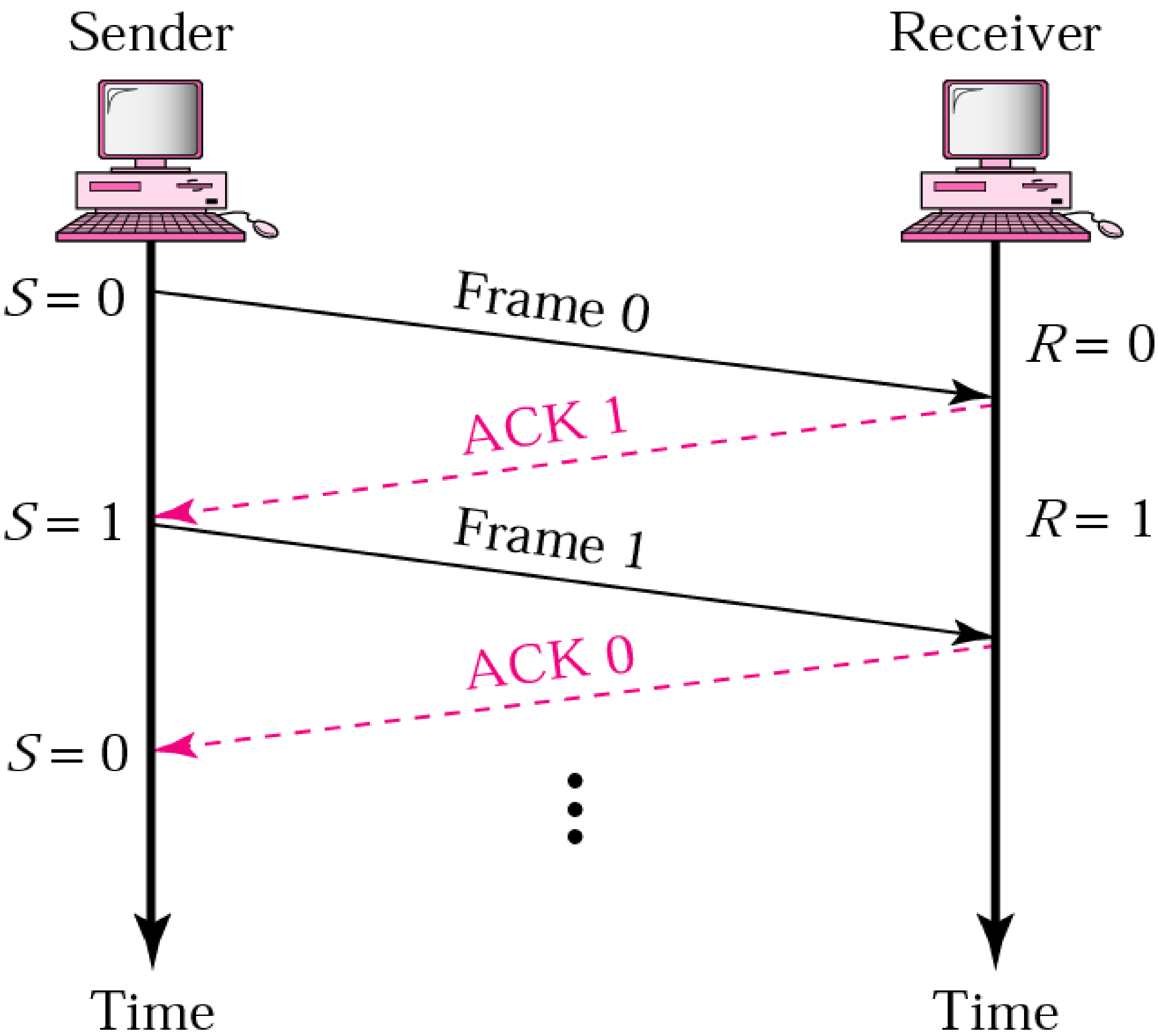
Simplex Stop-and-Wait Protocol



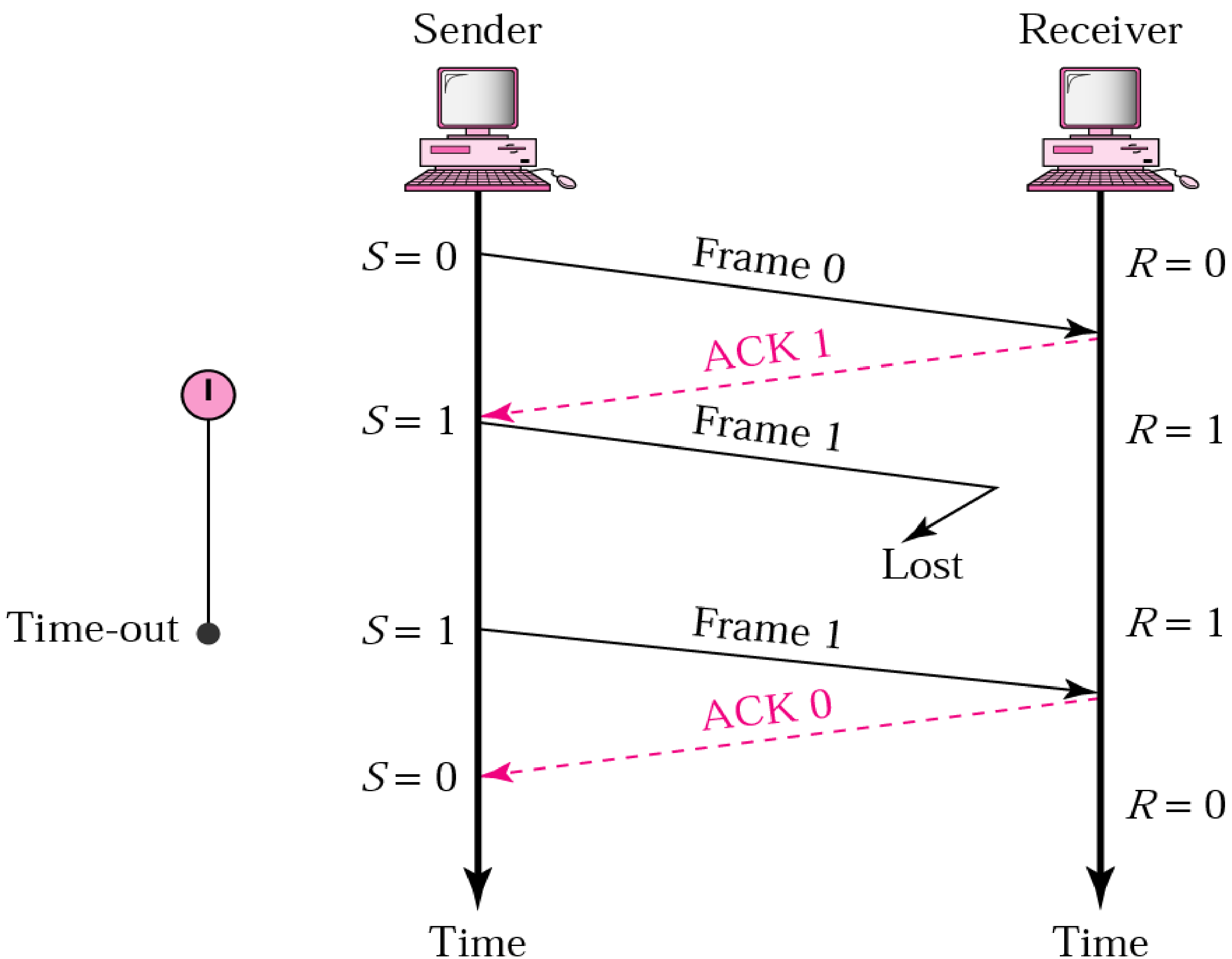
Stop and Wait for noisy channel

- Stop-and-Wait Automatic Repeat Request (ARQ)
- Add bits for error detection
- Corrupted frame discarded at rx
- ACK for correct frame
- Sx has timer, starts when frame send, keeps copy of frame
- If ACK doesnot come in time period, frame is resend
- Sequence numbers for frames/ACK = 0/1
- Frame 0 sx → rx
- ACK 1 rx→sx

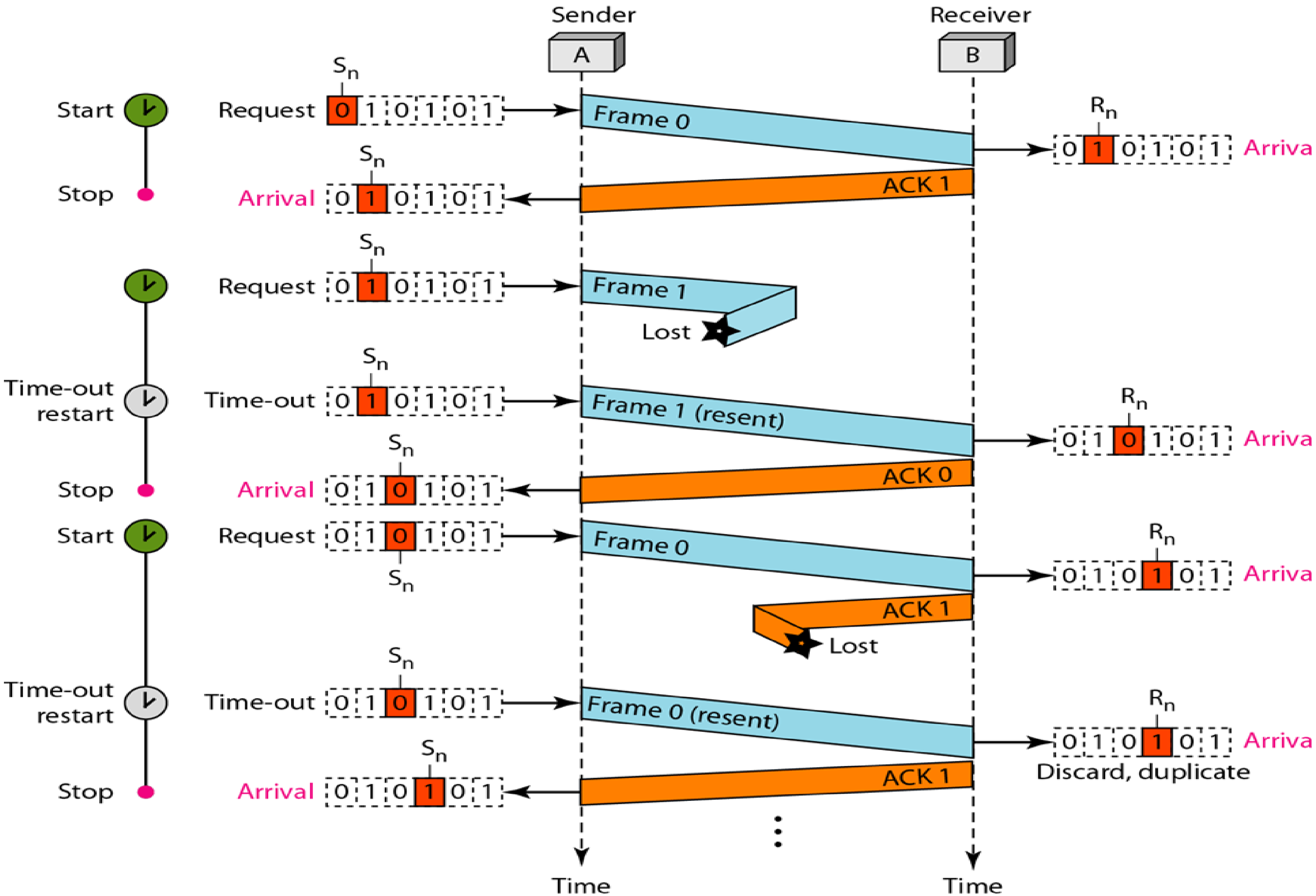
Stop and Wait for noisy channel - Normal operation



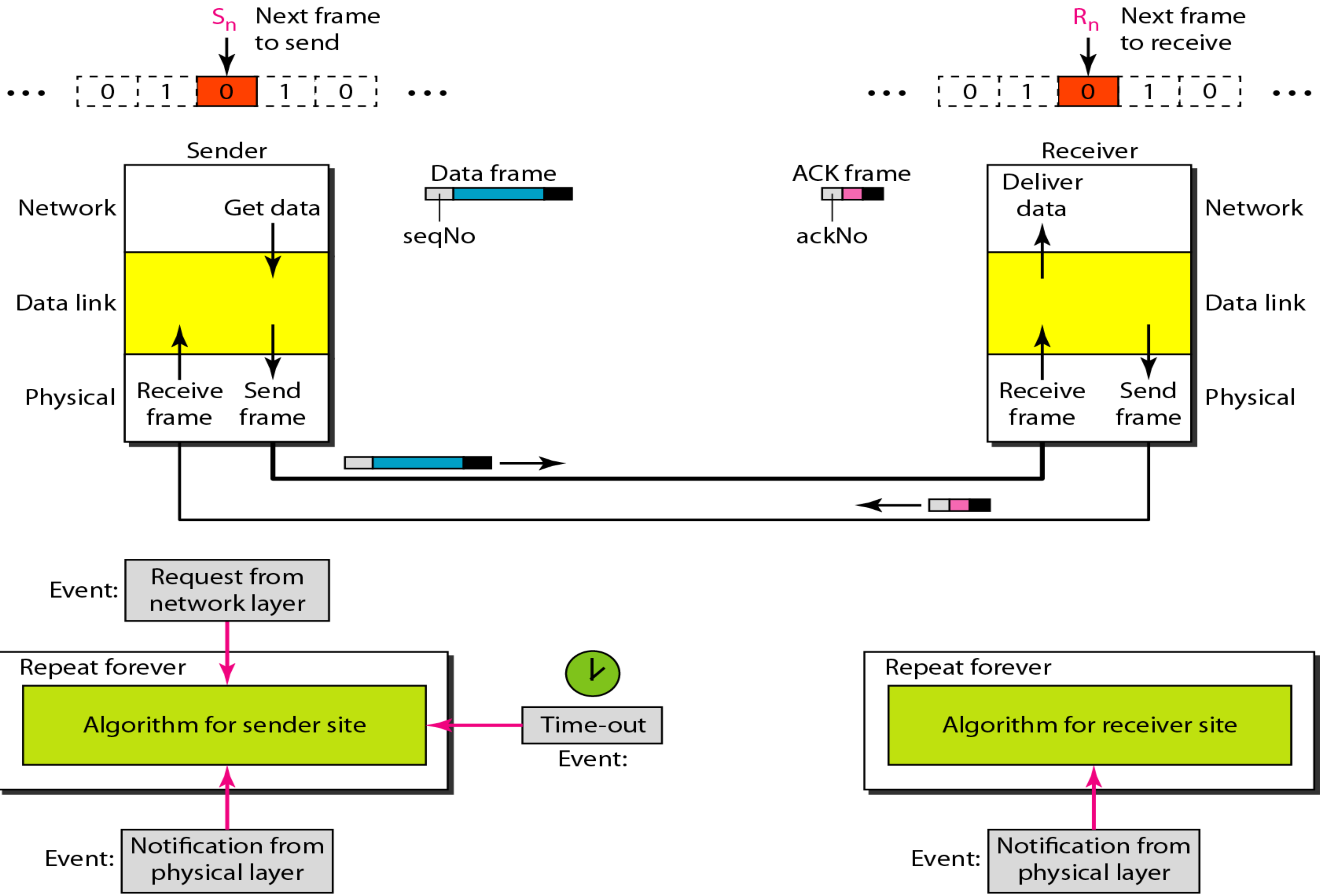
Stop-and-Wait ARQ, lost frame



Stop-and-Wait ARQ – All operations



Design of the Stop-and-Wait ARQ Protocol



Sender-site algorithm for Stop-and-Wait ARQ

```
Sn = 0;                              // Frame 0 should be sent first
canSend = true;                      // Allow the first request to go
while(true)                          // Repeat forever
{
  WaitForEvent();                    // Sleep until an event occurs
  if(Event(RequestToSend) AND canSend)
  {
     GetData();
     MakeFrame(Sn);                           //The seqNo is Sn
     StoreFrame(Sn);                          //Keep copy
     SendFrame(Sn);
     StartTimer();
     Sn = Sn + 1;
     canSend = false;
  }
  WaitForEvent();                             // Sleep
```

(continued)

Sender-site algorithm for Stop-and-Wait ARQ **(continued)**

```
    if(Event(ArrivalNotification)      // An ACK has arrived
    {
      ReceiveFrame(ackNo);             //Receive the ACK frame
      if(not corrupted AND ackNo == S_n) //Valid ACK
        {
          Stoptimer();
          PurgeFrame(S_{n-1});         //Copy is not needed
          canSend = true;
        }
     }

    if(Event(TimeOut)                  // The timer expired
    {
     StartTimer();
     ResendFrame(S_{n-1});             //Resend a copy check
    }
}
```

Receiver-site algorithm for Stop-and-Wait ARQ Protocol

```
Rn = 0;                        // Frame 0 expected to arrive first
while(true)
{
  WaitForEvent();              // Sleep until an event occurs
  if(Event(ArrivalNotification))  //Data frame arrives
  {
     ReceiveFrame();
     if(corrupted(frame));
        sleep();
     if(seqNo == Rn)              //Valid data frame
     {
      ExtractData();
       DeliverData();             //Deliver data
       Rn = Rn + 1;
     }
      SendFrame(Rn);              //Send an ACK
  }
}
```

Example

Assume that, in a Stop-and-Wait ARQ system, the bandwidth of the line is 1 Mbps, and 1 bit takes 20 ms to make a round trip. What is the bandwidth-delay product? If the system data frames are 1000 bits in length, what is the utilization percentage of the link?

Solution

The bandwidth-delay product is

$$\boxed{(1 \times 10^6) \times (20 \times 10^{-3}) = 20{,}000 \text{ bits}}$$

Example (continued)

System can send 20,000 bits during the time it takes for data to go from sender to receiver and then back again.

However, the system sends only 1000 bits.

Link utilization = 1000/20,000 x 100= or 5 %.

For a link with a high bandwidth or long delay, use of Stop-and-Wait ARQ wastes the capacity of the link.

Example

What is the link utilization in previous example if we have a protocol that can send up to 15 frames before stopping and worrying about the acknowledgments?

Solution

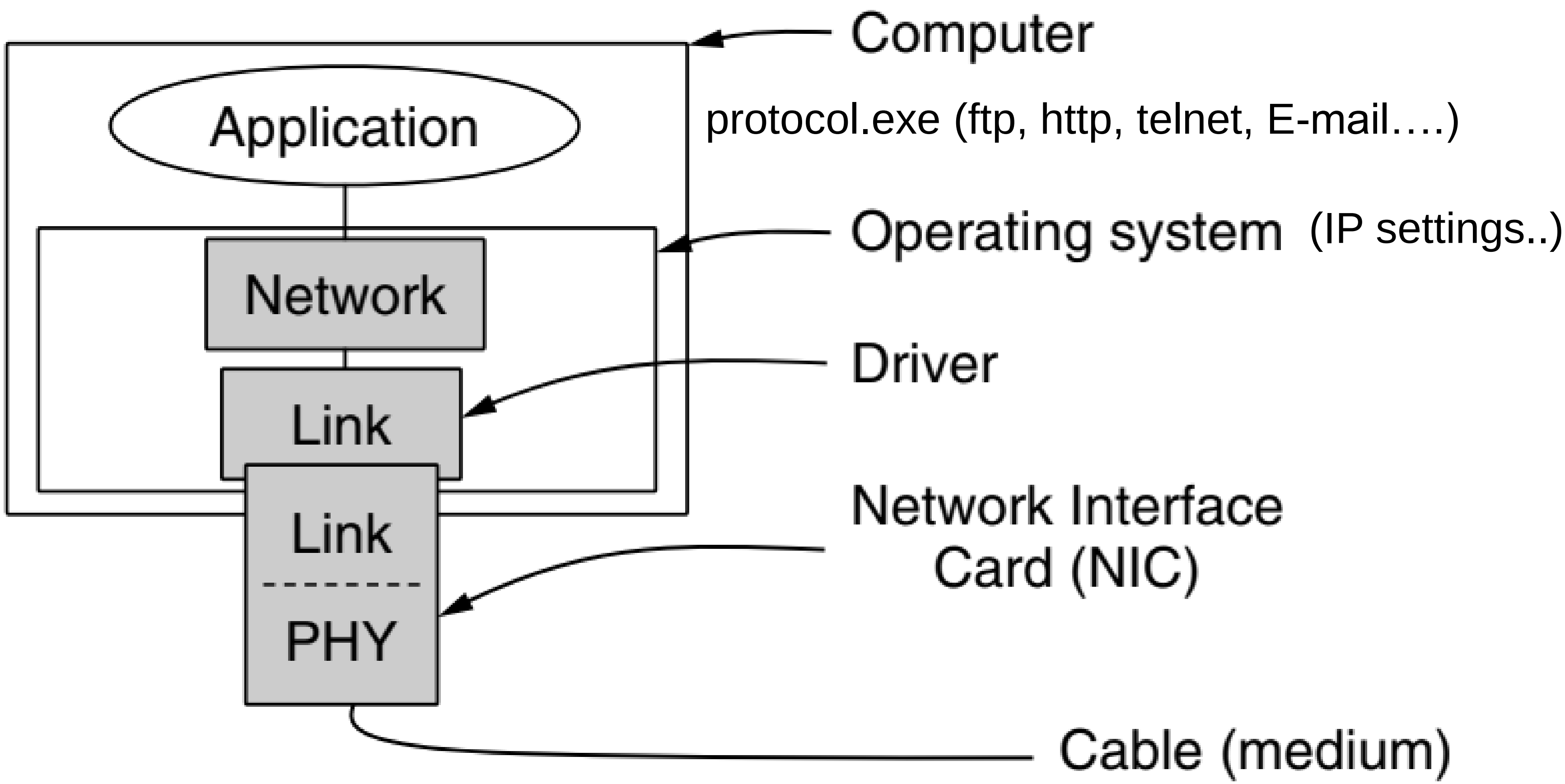
bandwidth-delay product = 20,000 bits.

System can send up to 15 frames (15,000 bits) during a round trip.

Link utilization = 15,000/20,000 = 75 %

If there are damaged frames, utilization percentage is much less because frames have to be resent.

Implementation of the physical, data link, network and application layers



Thank You!