

# **Data Communications And Networking** fourth Edition

Forouzan

## **Chapter 11**

### **Data Link Control**

# Data Link Control

The two main functions of the data link layer are :-

- 1-data link control (deals with the design and procedures for communication between two adjacent nodes: node-to-node communication).
- 2-media access control (deals how share the link).

**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. protocol :- is a set of rules that need to be implemented in software and run by the two nodes involved in data exchange at the data link layer.

**Data transmission** in the physical layer means moving bits in the form of a signal from the **source** to the **destination**.

The physical layer **provides** bit synchronization to ensure that the sender and receiver use the same bit durations and timing.

11

-

1

FRAMING

11-1

FRAMING



The data link layer needs to pack bits into **frames**, so that each frame is distinguishable from another. Our postal system practices a type of framing. The simple act of inserting a letter into an envelope separates one piece of information from another; the envelope serves as the delimiter.



## Topics discussed in this section:

**Fixed-Size Framing** – no boundaries for frames, size used as the delimiter

**Variable-Size Framing** – define beginning and end of frame, character oriented or bit oriented approach

11.5

**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.

11.6

## Fixed-Size Framing

Frames can be of fixed or variable size. **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.

ATM: Asynchronous Transfer Mode( connection oriented, high-speed network technology that is used in both LAN and WAN over optical fiber and operates upto gigabit speed.

11.7

## Variable-Size Framing

We concerns with variable-size framing, prevalent in local area networks.

In variable-size framing, we need a way to define the end of the frame and the beginning of the next.



Two approaches were used for this purpose:  
a character-oriented approach and a bit  
oriented approach.

11.8

## Character-Oriented Protocols

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.

11.9

**Figure 11.1** A frame in a character-oriented protocol



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. Figure 11.1 shows the format of a frame in a character-oriented protocol

11.10

## Bit-Oriented Protocols

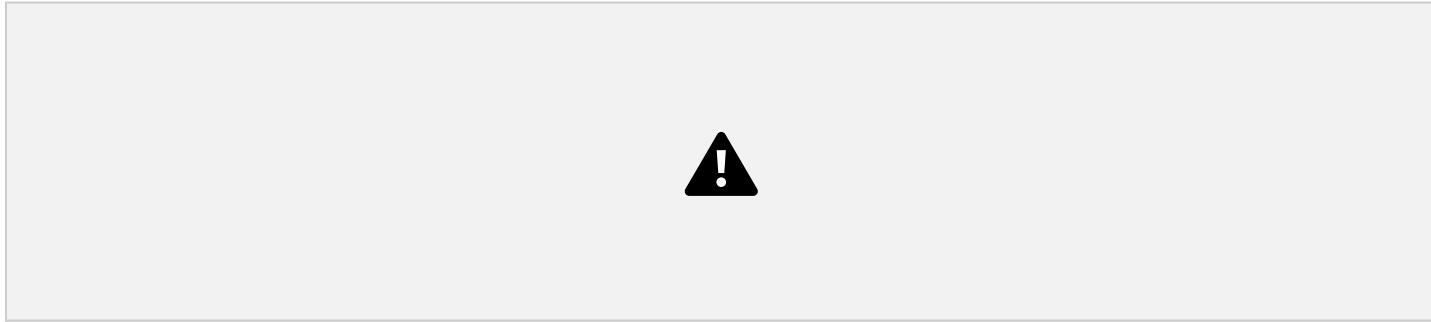
**In a bit-oriented protocol**, the data section of a frame is a **sequence of bits to be interpreted** by the upper layer as text, graphic, audio, video, and so on. However, in addition to headers (and possible trailers), we still need

a delimiter to separate one frame from the other.

**Most protocols** use a special 8-bit pattern flag **01111110** as the delimiter to define the beginning and the end of the frame, as shown in Figure 11.3.

11.11

**Figure 11.3** A frame in a bit-oriented protocol



**11.12**

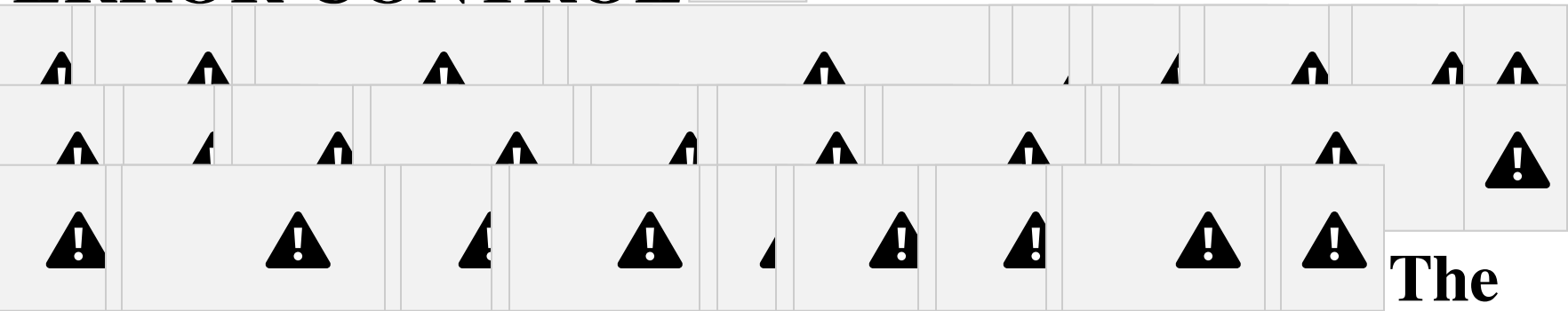


**11-2**



**FLOW AND**

# ERROR CONTROL



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

Topics discussed in this section:

---

Flow Control

11.13

**Flow control refers to a set of procedures used to restrict the amount of data that the sender can send before waiting for acknowledgment.**

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.

11.14

**Error control** is both error detection and error correction. It allows the receiver to **tell** the sender of any frames lost or damaged in transmission and coordinates the retransmission of those frames by the sender.

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

11.15

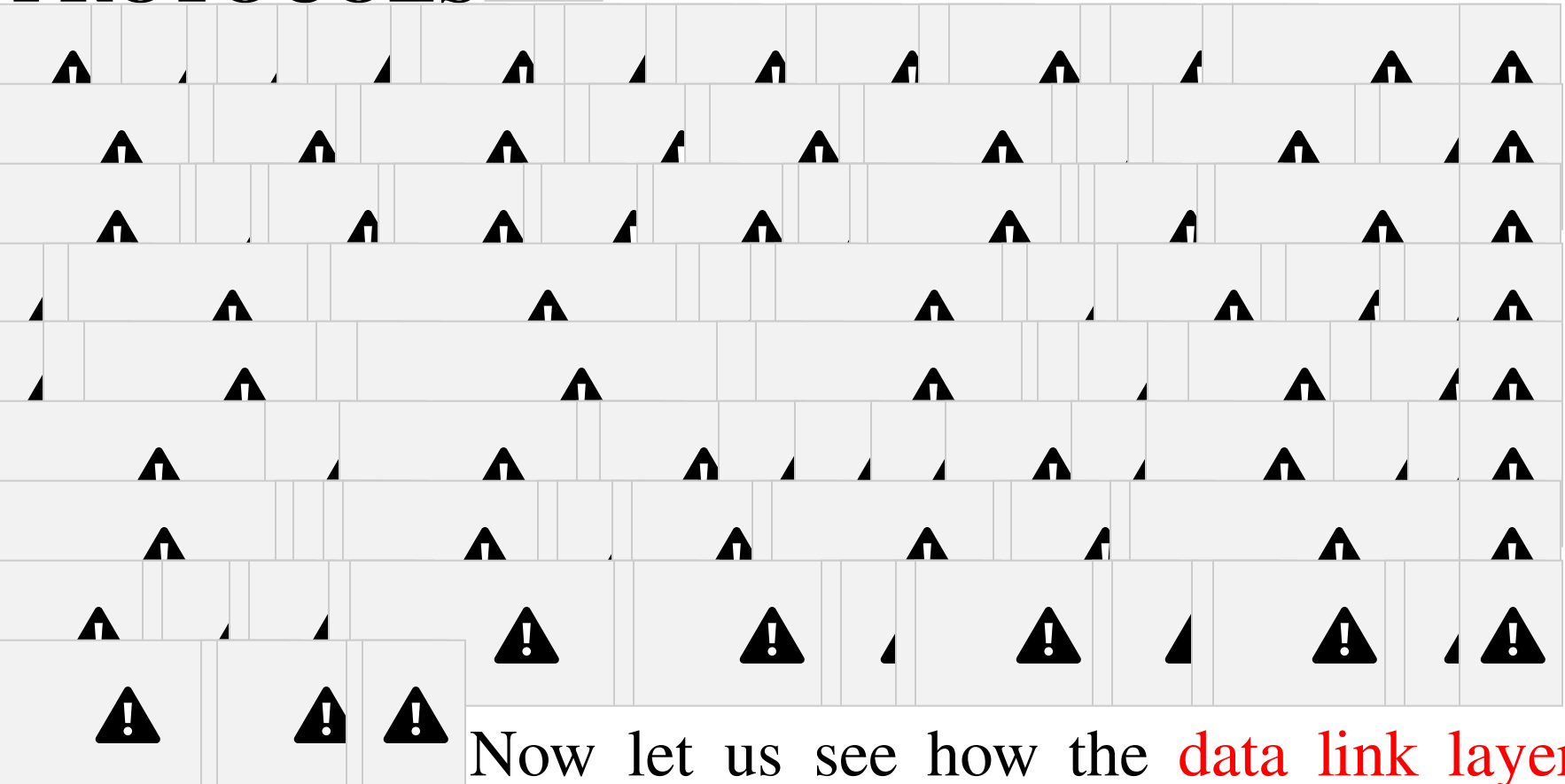




11-3



# PROTOCOLS



Now let us see 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. To make our discussions language-free, we have written in pseudocode a version of each protocol that concentrates mostly on the procedure **instead of** deeply the details of language rules.

11.16

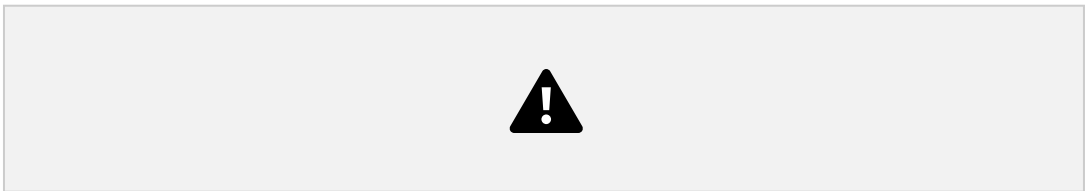
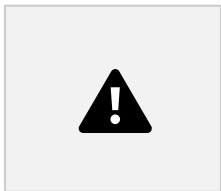
**Figure 11.5** classification of protocols discussed in this chapter



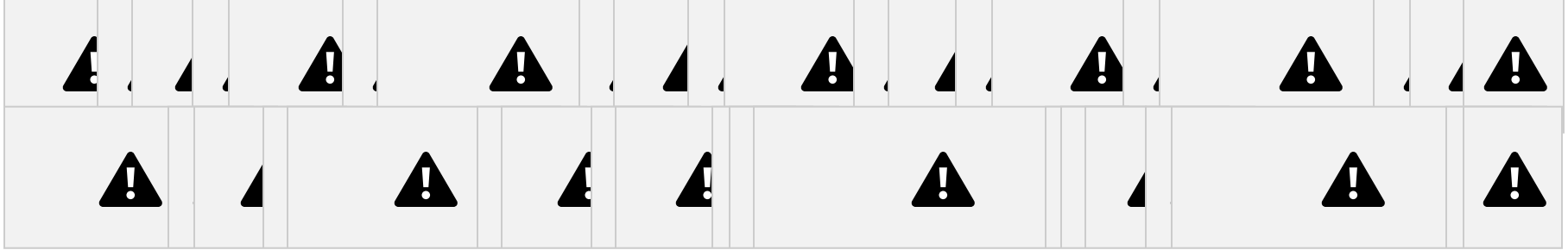
**11.17**



**11-4**



**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.



Topics discussed in this

section:

---

**Simplest Protocol – has no flow or error control**

**Stop-and-Wait Protocol – sender sends one frame, stops until it receives agree from receiver and then sends the next frame**

11.18

## **1-Simplest Protocol**

- Our first protocol, which we call the Simplest Protocol, is one

that has no flow or error control. Like other protocols, it is a **unidirectional** protocol in which **data frames are traveling in only one direction**-from the sender to receiver.

- 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 network layer, which can also accept the packet immediately.
- In other words, the receiver can never be fill out with incoming frames.

11.19

**Figure 11.6** The design of the simplest protocol with no flow or error control



**11.20**



**11.21**





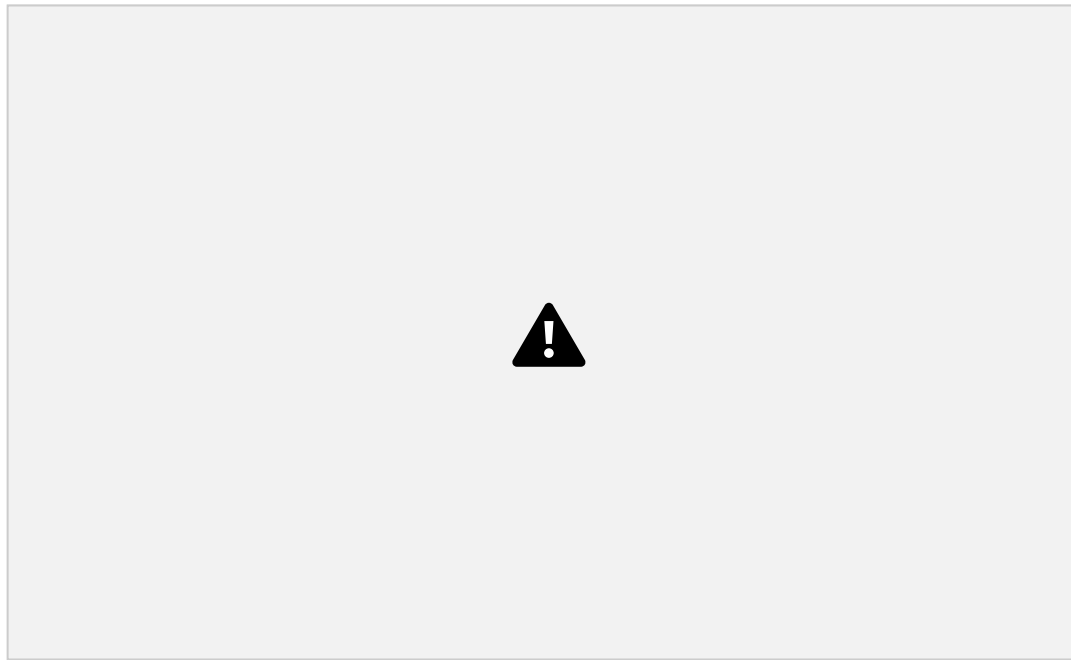
11.22

## Example 11.1

Figure 11.7 shows an example of communication using this protocol. It is very

simple. 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.

**Figure 11.7** Flow diagram for Example 11.1



11.23

## 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.

- We need to tell the sender to slow down. There must be feedback from the receiver to the sender.
- The sender sends one frame, **stops** until it receives agreement the receiver (okay to go ahead), **and then** sends the next frame. **We still have unidirectional communication for data frames**, but auxiliary ACK frames (simple tokens of acknowledgment) travel from the other direction. **We add flow control** to our previous protocol.

11.24

**Figure 11.8** Design of Stop-and-Wait Protocol

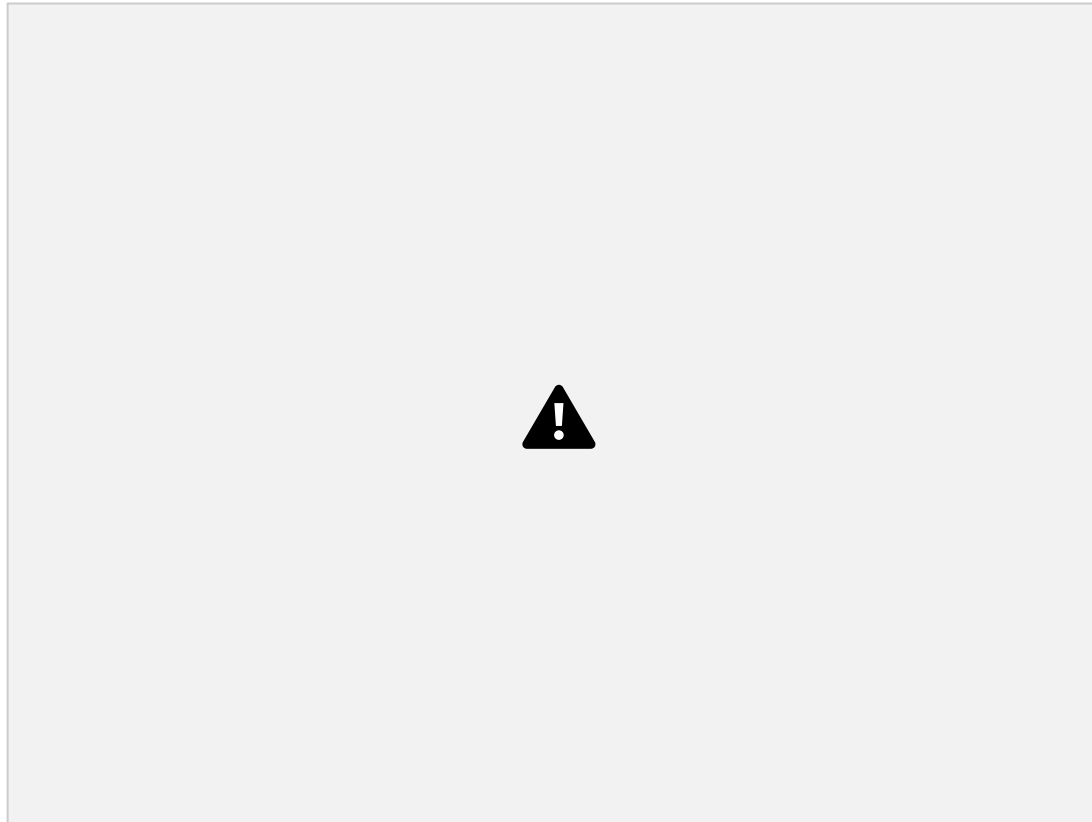


11.25

**Example 11.2**

Figure 11.7 shows an example of communication using this protocol. It is still very simple. 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.**

**Figure 11.7** Flow diagram for Example 11.2





11-5



# 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 discuss three protocols in this section that use error control.



## **Topics discussed in this section:**

---

**Stop-and-Wait Automatic Repeat  
Request (ARQ) Go-Back-N Automatic Repeat  
Request  
Selective Repeat Automatic Repeat Request**

11.27

**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.**

**In Stop-and-Wait ARQ, we use sequence numbers to number the frames.**

**The sequence numbers are based on modulo-2 arithmetic.**

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

11.28

**Figure 11.10** Design of the Stop-and-Wait ARQ Protocol





11.29

## Example 11.3

Figure 11.11 shows an example of **Stop-and-Wait ARQ**. 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(second ) 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.

**Figure 11.11** Flow diagram for Example 11.3



11.30

# Pipelining

- In networking and in other areas, a task is often begun before the previous task has ended. This is known as **pipelining**. There is **no** pipelining in Stop-and-Wait ARQ because we need to wait for a frame to reach the destination and be acknowledged before the next frame can be sent.
- Pipelining improves the efficiency of the transmission if the number of bits in transition is large with respect to the bandwidth-delay .

11.31

## Go-Back-N Automatic Repeat Request

- To improve the efficiency of transmission (filling the pipe), multiple frames must be in transition while waiting for acknowledgment. In other words, we need to let more than one frame be **outstanding** to keep the channel busy while the sender is 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, **Thus need sequence number for frames.**

11.32

**For example**, if  $m$  is 4, the only sequence

numbers are **0** through **15** inclusive. However, we can repeat the sequence.



**Note**

**In the Go-Back-N Protocol, the sequence numbers are modulo  $2^m$ , where  $m$  is the size of the sequence number field in bits.**



11.34

**The send window is an abstract concept defining an imaginary box of size  $2^m - 1$  with**

**three variables:  $S_f$ ,  $S_n$ , and  $S_{size}$ .**

**$S_f$**  defines the sequence number of the first (oldest) outstanding frame.

**$S_n$**  holds the sequence number that will be assigned to the next frame to be sent.

**$S_{size}$**  defines the size of the window, which is fixed in our protocol.

**The send window can slide one or more slots when a valid acknowledgment arrives.**

11.35

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



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.

11.36

**Figure 11.14** Design of Go-Back-N ARQ





11.37

# Go-Back-N ARQ **Versus** Stop-and- Wait ARQ

- there is a similarity between Go-Back-N ARQ and Stop-and-Wait ARQ. We can say that the Stop and-Wait ARQ Protocol is **actually** a Go-Back-N ARQ in which there are only two sequence numbers and the send window size is 1. In other words,  $m = 1$ ,  $2^m - 1 = 1$ . In Go-Back-N ARQ, we said that the addition is modulo-  $2^m$  ;  
In Stop-and-Wait ARQ it is 2, which is the same as  $2^m$  when  $m = 1$ .

# Selective Repeat Automatic Repeat Request

- Go-Back-N ARQ simplifies the process at the receiver site. The receiver keeps track of **only one** variable, and there is no need to buffer out-of-order frames; they are simply discarded. However, this protocol is very **inefficient** for a **noisy link**. Why ? In a noisy link a frame has a higher probability of damage, which means the resending of multiple frames.
- This resending uses up the **bandwidth** and **slows down** the **transmission**. For noisy links, there is another mechanism that does not resend N frames when just one frame is damaged;
- only the damaged frame is resent. This mechanism is **called Selective Repeat ARQ**. It is **more efficient for noisy links**, but the **processing at the receiver** is more complex

## 11.39

- This Protocol also uses **two** windows: a send window and a receive window.
- There are differences between the windows in this protocol and the ones in Go-Back-N. **First**, the size of the send window is much smaller; it is  $2^{m-1}$ . The reason for this will be discussed later. **Second**, the receive window is the **same size** as the send window.
- **For example**, if  $m = 4$ , the sequence numbers go from 0 to 15, **but the size** of the window is just 8 (it is 15 in the Go-Back-N Protocol). The smaller window size means less efficiency in filling the pipe, **but** the fact that there are fewer duplicate frames.

## 11.40

## Figure 11.18 Send window for Selective Repeat ARQ



11.41

- The **receive window** in Selective Repeat is totally different from the one in Go Back-N. First, the size of the receive window is the **same as** the size of the send window ( $2^{m-1}$ ). The Selective Repeat Protocol **allows** as many frames as

the size of the receive window to arrive **out of order** and be kept until there is a set of in-order frames to be delivered to the network layer.

- Because the sizes of the send window and receive window are the same, all the frames in the send frame can arrive out of order and be stored until they can be delivered. We need, however, **to mention that the receiver never** delivers packets out of order to the network layer. Figure 11.19 shows the receive window in this

11.42

**Figure 11.19** Receive window for Selective Repeat ARQ



**11.43**

**Figure 11.20** Design of Selective Repeat ARQ



**11.44**



**Note**



**In Selective Repeat ARQ, the size of the sender and receiver window must be at most one-half of  $2^m$ .**

11.45

## **Piggybacking**

- The three protocols we discussed in this section are all **unidirectional**: data frames flow in only one direction although control information such as ACK and NAK frames can travel in

the other direction. In real life, data frames are normally flowing in both directions: from node A to node B and from node B to node A. This means that the **control information** also needs to flow in both directions.

- A technique called **piggybacking** is used to improve the efficiency of the **bidirectional** protocols. When a frame is carrying data from A to B, it can also carry control information about arrived (or lost) frames from B; and verse versa.

11.46

### **Figure 11.24** Design of piggybacking in Go-Back-N ARQ

**Piggybacking – control information flow in both directions and improves the efficiency of bidirectional protocols**



11.47

- **Note** that each node now has two windows: one send window and one receive window. Both also need to use a timer.
- An important point about piggybacking is that both sites must use the same algorithm.

- This algorithm is complicated because it needs to combine two arrival events into one.

11.48



**HDLC**



High-level Data Link Control (HDLC) is a bit-oriented protocol for communication over point-to-point and multipoint links. It implements the ARQ mechanisms .

HDLC provides two common transfer modes that can be used in different configurations: normal response mode (**NRM**) and

asynchronous balanced mode (**ABM**).

In normal response mode (NRM), the station configuration is unbalanced. We have one primary station and multiple secondary



stations. A primary station can send commands; a secondary station can only respond. The NRM is used for both point-to-point and multiple-point links, as shown in Figure 11.25.



## 11.50

### ■ Asynchronous Balanced Mode

In asynchronous balanced mode (ABM), the configuration is balanced. The link is point-to-point, and each station can function as a primary and a secondary (acting as peers), as

shown in Figure 11.26. This is the common mode today.



**11.51**