BACHELOR OF COMPUTER APPLICATIONS

SEMESTER 3

DCA2201

COMPUTER NETWORKING

Unit 4

Data Link Layer – Reliable Transmission

Table of Contents

1. INTRODUCTION

In the previous unit, we discussed the basics of datalink layer and framing and error correction in datalink layer. Data link layer encapsulates each network layer packet within a link layer frame before transmission over the link. Various error control measures are required to ensure a reliable transmission of data. In this unit, we will discuss the concept of data link layer protocols. Data link layer protocols are procedures for achieving reliable, efficient communication of whole units of information called frames between two machines connected by a communication channel.

We will start this unit with an explanation about the design issues related to the data link layer. Then we will discuss different elementary data link protocols such as simplex protocol, stop and wait protocol for error free channel and stop and wait protocol for noisy channel.

1.1 Objectives

After studying this unit, you should be able to:

- *Describe design issues of datalink layer*
- *Explain simplex protocol*
- *Describe stop and wait protocol for an error-free channel*
- *Explain stop and wait protocol for a Noisy channel*

2. DATA LINK LAYER DESIGN ISSUES

Let us now discuss the issues of the data link layer. The data link layer has a number of functions, such as: providing a well-defined service interface to the network layer, dealing with transmission errors and regulating the flow of data so that slow receivers are not flooded by fast senders. The data link layer accepts the packets from the network layer and encapsulates them into frames for transmission. Each frame has a header, payload and trailer as shown in figure 4.1.

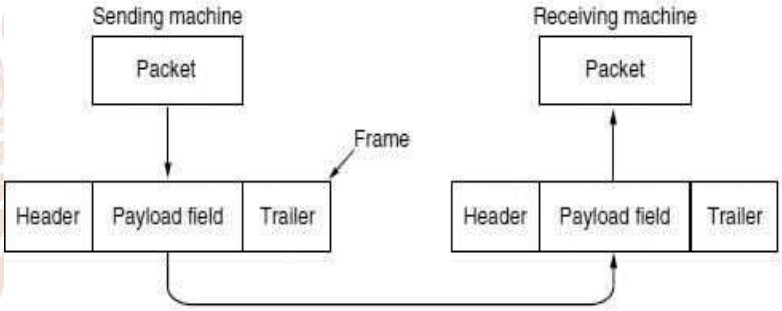

Figure 4.1: Relationship between packets and frames

In the following sections, along with framing, we will discuss error control and flow control in detail.

2.1 Services Provided to The Network Layer

The major service of Data link layer is to transfer data from the network layer on the source machine to the network layer on the destination machine. The data link layer can be designed to offer various services such as: (i) Unacknowledged connectionless service, (ii) Acknowledged connectionless service and (iii) Acknowledged connection-oriented service.

In unacknowledged connectionless service, the source machine sends independent frames to the destination machine without having the destination machine acknowledge them. There is no logical connection between the sender and the receiver. One good example of this service is the ethernet. If a frame is lost due to noise on the line, no attempt is made to find out the loss and recover from it. This class of service is appropriate when the error rate is very low, so recovery is left to higher layers. It is also appropriate for real-time traffic, such as voice, in which late data is worse than bad data.

In acknowledged connectionless service, there are still no logical connections used, but each frame which are sent is individually acknowledged. So here, the sender comes to know whether a frame has arrived correctly or been lost. If it has not arrived within a specified time interval, it can be sent again. This service is useful over unreliable channels, such as wireless systems. IEEE 802.11 (WiFi) is a good example of this class of service.

The most denoted service the data link layer can provide to the network layer is connection-oriented service. With this service, the source and destination machines establish a connection before any data is transferred. Each frame sent over the connection is numbered, and the data link layer guarantees that each frame sent is indeed received. Furthermore, it guarantees that each frame is received exactly once and that all frames are received in the right order. Connection-oriented service thus provides the network layer processes with the equivalent of a reliable bit stream. It is appropriate over long, unreliable links such as a satellite channel or a long-distance telephone circuit.

In connection-oriented service, there are three distinct phases. The first phase is connection establishment with initialization of variables on both sender and receiver side. One or more frames are actually transmitted in the second phase. Connection is released in the third phase.

2.2 Framing

Data link layer break up the data packets received from the network layer to discrete frames, compute a token called checksum (you have already read about checksum in unit 3) for each frame and include the checksum in the frame when it is transmitted. At the receiver side, the checksum is recomputed. If there is a mismatch in the newly computed checksum and the one contained in the frame, the data link layer knows that an error has occurred and takes steps to deal with it. There are four different methods for framing. They are:

- byte count
- flag bytes with byte stuffing
- flag bits with bit stuffing
- Physical layer coding violations.

The first framing method uses a field in the header to specify the number of bytes in the frame. At the receiver side, byte count is analyzed to find out the number of bytes in the frame and hence the end of the frame is also found. Figure 4.2 (a) depicts the byte count method. In this figure, there are four small example frames of sizes 5, 5, 8, and 8 bytes, respectively.

One of the major problems with this algorithm is that the count can be misrepresented by a transmission error. For example, if the byte count of 5 in the second frame of Fig. 4.2 (b) becomes a 7 due to a single bit flip, the destination will get out of synchronization. It will then be unable to locate the correct start of the next frame. Since checksum is incorrect, destination can identify that an error occurred during transmission, but it has no way to detect the starting of the next frame. Sending back acknowledgement for retransmission is also not effective because receiver cannot find out how many bytes to skip over to get to the start of the retransmission. For this reason, the byte count method is rarely used by itself.

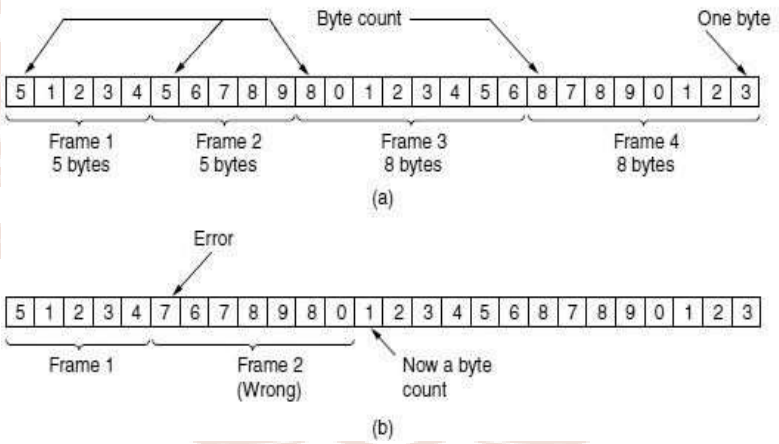

Figure 4.2: A byte stream (a) without errors (b) with one error

The second method is the use of a flag byte, which is a special byte and it is used as both the starting and ending delimiter. Two consecutive flag bytes indicate the end of one frame and the start of the next. Thus, if the receiver ever loses synchronization, it can just search for two flag bytes to find the end of the current frame and the start of the next frame. The trouble in this case is that the flag byte can occur in data also. One way to solve this problem is to have the sender's data link layer insert a special escape byte (ESC) just before each''accidental'' flag byte in the data. Thus, a framing flag byte can be distinguished from one in the data by the absence or presence of an escape byte before it. The data link layer on

the receiving end removes the escape bytes before giving the data to the network layer. This technique is called **byte stuffing**. If an escape byte occurs in the middle of data, it is too stuffed with an escape byte. At the receiver, the first escape byte is removed, leaving the data byte that follows it. The process of byte stuffing is shown in figure 4.3.

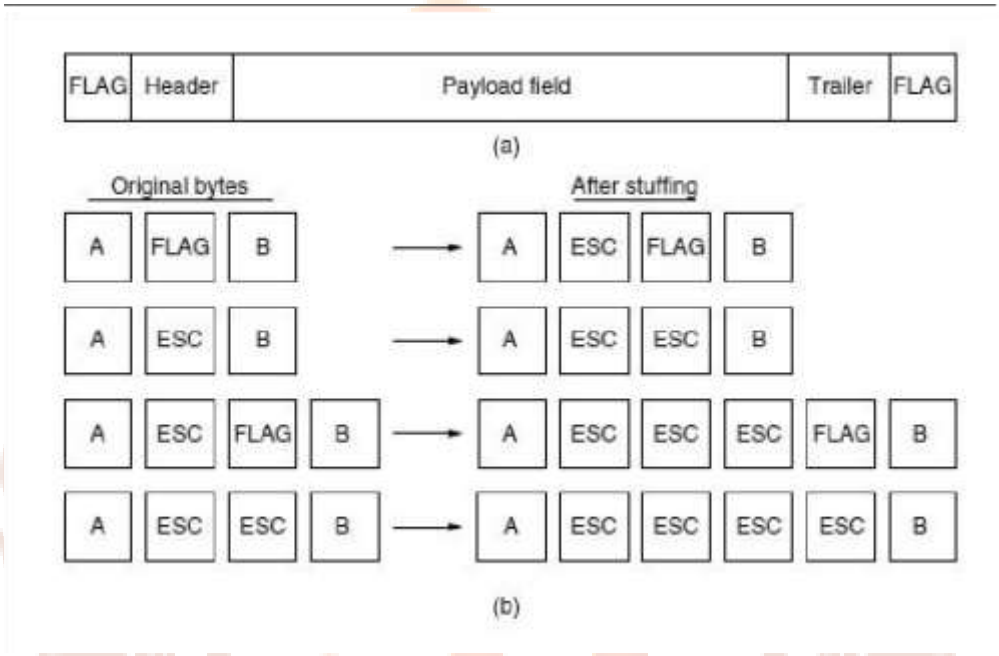

Figure 4.3: (a) A frame delimited by flag bytes.

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

The third method is bit stuffing. In this scheme, framing has been done at the bit level, so frames can contain an arbitrary number of bits made up of units of any size. As we discussed in the previous chapter, bit stuffing was developed for the once very popular HDLC (High level Data Link Control) protocol. In this method, each frame begins and ends with a special bit pattern, 01111110. This pattern is a flag byte. Whenever the sender's data link layer encounters five consecutive 1s in the data, it automatically stuffs a 0 bit into the outgoing bit stream. When the receiver sees five consecutive incoming 1 bit, followed by a 0 bit, it automatically deletes the 0 bit. Figure 4.4 gives an example of bit stuffing.

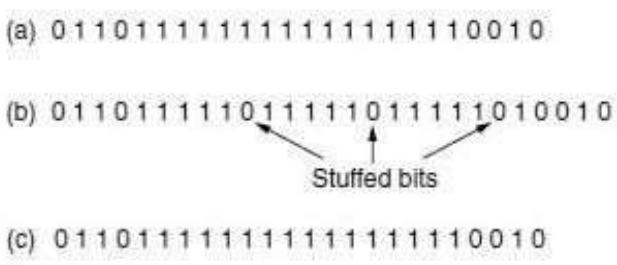

Figure 4.4: bit stuffing (a) The original data (b) the data as they appear on the line. (c) The data stored in receiver's memory after destuffing.

The last method of framing is physical layer coding violations, which is to use a shortcut from the physical layer. We can use some reserved signals to indicate the start and end of frames. In effect, we are using 'coding violations to delimit frames. The beauty of this scheme is that, because they are reserved signals, it is easy to find the start and end of frames and there is no need to stuff the data. Many data link protocols use a combination of these methods for safety.

2.3 Error Control

At the destination, when data link layer receives bit stream from physical layer it is not guaranteed to be error free. Some bits may have different values and the number of bits received may be less than, equal to, or more than the number of bits transmitted. It is the responsibility of the data link layer to detect and correct errors.

The usual way to ensure reliable delivery is to provide the sender with some feedback about what is happening at the other end of the line. Typically, the protocol calls for the receiver to send back special control frames bearing positive or negative acknowledgements about the incoming frames. If the sender receives a positive acknowledgement about a frame, it knows the frame has arrived safely. On the other hand, a negative acknowledgement means that something has gone wrong and the frame must be transmitted again. Another trouble is that the frame may vanish completely due to hardware troubles. In this case the receiver will not react and will not give any feedback. Similarly, if the acknowledgement frame is lost, the sender will not know how to proceed. This problem is handled by introducing timers into the data link layer. When the sender transmits a frame, it generally also starts a timer. The

timer will wait for a period which is equal to the time required for the frame to reach the destination, be processed there and have the acknowledgement propagate back to the sender. In normal case, if the frame is correctly received, the acknowledgement will get back before the timer runs out, in which case the timer will be cancelled. However, if either the frame or the acknowledgement is lost, the timer will go off after a specific period of time known as round trip time (which is equivalent to the time required for the frame to reach destination plus the time required for the acknowledgement to reach the sender.), so that sender will be alert about the problem and it will transmit the frame again.

However, when frames may be transmitted multiple times there is a danger that the receiver will accept the same frame two or more times and pass it to the network layer more than once. To prevent this from happening, it is generally necessary to assign sequence numbers to outgoing frames, so that the receiver can distinguish retransmissions from originals. Hence, this is one of the major duty of the datalink layer to manage the timers and sequence numbers so as to ensure that each frame is ultimately passed to the network layer at the destination exactly once.

2.4 Flow Control

The measures of data flow control in a network should be managed by data link layer, when there is a fast sender and slow receiver and if sender wants to transmit frames faster than the receiver which can accept them. This situation can occur when the sender is running on a fast, powerful computer and the receiver is running on a slow, low-end machine. There are two different flow control approaches, they are: feedback-based flow control and rate-based flow control. In feedback-based flow control, the receiver sends back information to the sender giving it permission to send more data. In rate-based flow control, the protocol has a built-in mechanism that limits the rate at which senders may transmit data, without using feedback from the receiver.

Self-Assessment Questions - 1

1. Which layer provides services to the network layer?

   (a) Datalink layer (b) transport layer (c) physical layer (d) session layer

2. In _____ service, there are still no logical connections used, but each frame sent is individually acknowledged.
3. Ethernet is a good example for service.
4. ___is an example for Acknowledged connectionless service.
5. The data link layer on the receiving end removes the escape bytes before giving the data to the network layer. This technique is called_____
6. In bit stuffing, each frame begins and ends with a special bit pattern called _____.
7. Measures should be taken by data link layer, when there is a fast sender and slow receiver.

3. ELEMENTARY DATA LINK PROTOCOLS

Assume that machine 'A' wants to send a long stream of data to machine B using a reliable, connection-oriented service. B also wants to send data to A simultaneously. A data link layer frame consists of an embedded packet, some control information (in the header) and a checksum (in the trailer). The frame is then transmitted to the data link layer on the other machine. We also assume that there exists suitable procedure to send (*to_physical_layer*) and receive (*from_physical_layer*) frame.

Initially the receiver just waits for the procedure call, *wait_for_event* (&*event*). This procedure only returns when something has happened (e.g., a frame has arrived). Upon return, the variable event tells what happened. The set of possible events differ based on protocols. When a frame arrives at the receiver, the checksum will be recomputed at the receiver side. If the checksum is incorrect, the datalink layer is so informed (so *event=chksum_err*). If the inbound frame arrived undamaged, the data link layer is also informed (*event = frame arrival*) so that it can acquire the frame for inspection using from physical layer procedure. When the receiver side datalink layer acquires an undamaged frame, it checks the control information in the header, and if everything is correct then it passes the packet portion to the network layer. Figure 4.5 shows the code about some declarations (in C) common to many of the data link protocols discussed in the coming sections.

Five data structures are defined there. They are: *Boolean*, *seq nr, packet, frame kind*, and *frame*. A *Boolean* is an enumerated data type and can take on values such as true and false. A *seq nr* is a small integer used to number the frames. These sequence numbers run from 0 up to *MAX SEQ,* which is defined in each protocol needing it. A *packet* is the unit of information exchanged between the network layer and the data link layer.

```
#define MAX PKT 1024 /* determines packet size in bytes */
typedef enum {false, true} boolean; /* boolean type */
typedef unsigned int seq nr; /* sequence or ack numbers */
typedef struct {unsigned char data[MAX PKT];} packet; /* packet definition */
typedef enum {data, ack, nak} frame kind; /* frame kind definition */
typedef struct { /* frames are transported in this layer */
frame kind kind; /* what kind of frame is it? */
seq nr seq; /* sequence number */
seq nr ack; /* acknowledgement number */
packet info; /* the network layer packet */
} frame;

/* Wait for an event to happen; return its type in event. */
void wait for event(event type *event);
/* Fetch a packet from the network layer for transmission on the channel. */
void from network layer(packet *p);
/* Deliver information from an inbound frame to the network layer. */
void to network layer(packet *p);
/* Go get an inbound frame from the physical layer and copy it to r. */
void from physical layer(frame *r);
/* Pass the frame to the physical layer for transmission. */
void to physical layer(frame *s);
/* Start the clock running and enable the timeout event. */
void start timer(seq nr k);
/* Stop the clock and disable the timeout event. */
void stop timer(seq nr k);
/* Start an auxiliary timer and enable the ack timeout event. */
void start ack timer(void);
/* Stop the auxiliary timer and disable the ack timeout event. */
void stop ack timer(void);
/* Allow the network layer to cause a network layer ready event. */
void enable network layer(void);
/* Forbid the network layer from causing a network layer ready event. */
void disable network layer(void);
/* Macro inc is expanded in-line: increment k circularly. */

#define inc(k) if (k < MAX SEQ) k = k + 1; else k = 0
```

Figure 4.5: Common definitions used in datalink protocols

A *frame* is composed of four fields. They are: *kind, seq, ack,* and *info*. The first three fields contain control information and the last may contain actual data to be transferred. These control fields are collectively called the frame header. The *kind* field tells whether there are any data in the frame, because some of the protocols distinguish frames containing only control information from those containing data as well. The *seq* and *ack* fields are used for sequence numbers and acknowledgements, respectively. The info

field of a data frame contains a single packet.

Number of procedures are also listed in figure 4.5. The procedure wait_for_event wait for something to happen. The procedures *to_network_ layer* and *from_network_layer* is used by the data link layer to pass packets to the network layer and accept packets from the network layer, respectively. *From_physical_layer* and *to_physical_layer* pass frames between the data link layer and the physical layer. The procedures *start_timer* and *stop_timer* turn the timer on and off, respectively. The procedures *start_ack_timer* and *stop_ack_timer* control an auxiliary timer used to generate acknowledgements under certain conditions.

3.1 Simplex Protocol

Simplex protocol is one of the data link protocols. As the name specifies, this protocol is as simple as it can be because it does not worry about the possibility of anything going wrong. Data is transmitted in one direction only. Both the transmitting and receiving network layers are always ready to transmit or receive data. Processing time can be ignored. Infinite buffer space is available. And best of all, the communication channel between the data link layers never damages or loses frames. This protocol consists of two procedures, sender () and receiver (). The sender runs in the data link layer of the source machine, and the receiver runs in the data link layer of the destination machine. No sequence numbers or acknowledgements are used.

Here, sender is in an infinite while loop just pumping data out onto the line as fast as it can. The body of the loop consists of three actions, they are: fetch a packet from network layer, construct an outbound frame using the variable s, and send the frame. The receiver is equally simple. Initially, it waits for something to happen, the only possibility being the arrival of an undamaged frame. Once the frame arrives, the procedure *wait_for_event returns*, with event set to *frame_arrival*. The call to *from_physical_layer* removes the newly arrived frame from the hardware buffer and puts it in the variable r, where the receiver code can get to it. Finally, the data portion is passed on to the network layer, and the data link layer settles back to wait for the next frame.

Simplex protocol is an unrealistic protocol because it doesn't handle flow control or error correction. Figure 4.6 shows the code and is the representation of simplex protocol.

```
typedef enum {frame_arrival} event_type;
#include "protocol.h"

void sender1(void)
{
 frame s;                          /* buffer for an outbound frame */
 packet buffer;                    /* buffer for an outbound packet */

 while (true) {
     from_network_layer(&buffer);  /* go get something to send */
     s.info = buffer;              /* copy it into s for transmission */
     to_physical_layer(&s);        /* send it on its way */
 }                                 /* Tomorrow, and tomorrow, and tomorrow,
                                      Creeps in this petty pace from day to day
                                      To the last syllable of recorded time.
                                         – Macbeth, V, v */
}

void receiver1(void)
{
 frame r;
 event_type event;                 /* filled in by wait, but not used here */

 while (true) {
     wait_for_event(&event);       /* only possibility is frame_arrival */
     from_physical_layer(&r);      /* go get the inbound frame */
     to_network_layer(&r.info);    /* pass the data to the network layer */
 }
}
```

Figure 4.6: Simplex Protocol

3.2 Stop And Wait Protocol For An Error-Free Channel

Another type of data link protocol is the stop and wait protocol. This protocol implements flow control mechanisms and prevents the sender from overloading the receiver. Flow control is the mechanism to prevent the sender from sending frames faster so that receiver is unable to process them. This protocol also assumes that the communication channel is error free. Protocols in which the sender sends one frame and then waits for an acknowledgement before proceeding are called stop-and-wait. In this protocol, the receiver sends feedback to the sender. After having passed a packet to its network layer, the receiver sends a little dummy frame back to the sender which, in effect, gives the sender permission to transmit the next frame. After sending a frame, the sender will wait for the little dummy

frame to arrive. Here, actual data traffic is from sender to receiver only, but frames travel in both directions. In this protocol, first sender sends a frame, then the receiver sends a frame (dummy frame for acknowledgment), then the sender sends another frame and so on. Figure 4.7 shows the code about the procedure for stop and wait protocol for an error-free channel.

```
typedef enum {frame_arrival} event_type;
#include "protocol.h"

void sender2(void)
{
 frame s;                          /* buffer for an outbound frame */
 packet buffer;                    /* buffer for an outbound packet */
 event_type event;                 /* frame_arrival is the only possibility */

 while (true) {
     from_network_layer(&buffer);  /* go get something to send */
     s.info = buffer;              /* copy it into s for transmission */
     to_physical_layer(&s);        /* bye-bye little frame */
     wait_for_event(&event);       /* do not proceed until given the go ahead */
 }
}

void receiver2(void)
{
 frame r, s;                       /* buffers for frames */
 event_type event;                 /* frame_arrival is the only possibility */
 while (true) {
     wait_for_event(&event);       /* only possibility is frame_arrival */
     from_physical_layer(&r);      /* go get the inbound frame */
     to_network_layer(&r.info);    /* pass the data to the network layer */
     to_physical_layer(&s);        /* send a dummy frame to awaken sender */
 }
}
```

Figure 4.7: Stop and Wait Protocol for an error-free channel

Similar to simplex protocol, the sender starts out by fetching a packet from the network layer, create a frame using it and sending it to the destination. In this protocol, the sender must wait until an acknowledgement frame arrives before looping back and fetching the next packet from the network layer. In the receiver side, after delivering packet to the network layer, receiver sends an acknowledgement frame back to the sender before entering the wait loop again.

3.3 Stop And Wait Protocol For A Noisy Channel

This is also another type of data link protocol. This protocol deals with the normal situation of a communication channel that makes errors. Frames can be either damaged or lost

completely in the communication channel. If the frame is damaged in transit, the receiver hardware will detect this while computing the checksum. When compared to the protocol for error free channel, here we are using an additional timer and the sender could send a frame, but the receiver would only send an acknowledgement frame if the data were correctly received. If a damaged frame arrived at the receiver, it would be discarded. After a while the sender would time out and send the frame again. This process would be repeated until the frame finally arrived intact.

Another trouble in this scenario is that, consider the situation where a sender A sends a packet 1 to the receiver B, it is correctly received at B and B sends an acknowledgement frame back to A. But the acknowledgment frame gets lost completely and the timer on A eventually times out. Since sender A has not received the acknowledgment, it assumes that the data frame was lost or damaged and sends the frame containing packet 1 again. This duplicate frame also arrives at B and is passed to the network layer there. Here, duplication of the frame occurs.

So, some way is required to identify the duplicate frame. The obvious way to achieve this is to have the sender put a sequence number in the header of each frame it sends. Then the receiver can check the sequence number of each arriving frame to see if it is a new frame or a duplicate to be discarded. A one-bit sequence number (0 or 1) is sufficient. At each instant of time, the receiver expects a particular sequence number next. When a frame containing the correct sequence number arrives, it is accepted and passed to the network layer, then acknowledged. Then the expected sequence number is incremented modulo 2 (i.e., 0 becomes 1 and 1 becomes 0). Any arriving frame containing the wrong sequence number is rejected as a duplicate.

```
#define MAX_SEQ 1                                   /* must be 1 for protocol 3 */
typedef enum {frame_arrival, cksum_err, timeout} event_type;
#include "protocol.h"

void sender3(void)
{
  seq_nr next_frame_to_send;                        /* seq number of next outgoing frame */
  frame s;                                          /* scratch variable */
  packet buffer;                                    /* buffer for an outbound packet */
  event_type event;

  next_frame_to_send = 0;                           /* initialize outbound sequence numbers */
  from_network_layer(&buffer);                      /* fetch first packet */
  while (true) {
      s.info = buffer;                              /* construct a frame for transmission */
      s.seq = next_frame_to_send;                   /* insert sequence number in frame */
      to_physical_layer(&s);                        /* send it on its way */
      start_timer(s.seq);                           /* if answer takes too long, time out */
      wait_for_event(&event);                       /* frame_arrival, cksum_err, timeout */
      if (event == frame_arrival) {
          from_physical_layer(&s);                  /* get the acknowledgement */
          if (s.ack == next_frame_to_send) {
              stop_timer(s.ack);                    /* turn the timer off */
              from_network_layer(&buffer);          /* get the next one to send */
              inc(next_frame_to_send);              /* invert next_frame_to_send */
          }
      }
  }
}

void receiver3(void)
{
  seq_nr frame_expected;
  frame r, s;
  event_type event;

  frame_expected = 0;
  while (true) {
      wait_for_event(&event);                       /* possibilities: frame_arrival, cksum_err */
      if (event == frame_arrival) {                 /* a valid frame has arrived */
          from_physical_layer(&r);                  /* go get the newly arrived frame */
          if (r.seq == frame_expected) {            /* this is what we have been waiting for */
              to_network_layer(&r.info);            /* pass the data to the network layer */
              inc(frame_expected);                  /* next time expect the other sequence nr */
          }
          s.ack = 1 - frame_expected;               /* tell which frame is being acked */
          to_physical_layer(&s);                    /* send acknowledgement */
      }
  }
}
```

Figure 4.8: Stop and Wait protocol for a Noisy Channel

Protocols in which the sender waits for a positive acknowledgement before advancing to the next data item are often called *ARQ (Automatic Repeat reQuest)* or *PAR (Positive Acknowledgement with Retransmission).* The procedure of this protocol is shown through code in figure 4.8. The sender remembers the sequence number of the *next frame to send* in next_frame_to_send; the receiver remembers the sequence number of the next frame expected in *frame_expected.* Each protocol has a short initialization phase before entering the infinite loop.

After transmitting a frame, the sender starts the timer. If it was already running, it would be reset to allow another full timer interval. The interval should be chosen in such a way that it should allow enough time for the frame to get to the receiver, process at the receiver, and for the acknowledgement frame to propagate back to the sender. Only when that interval has elapsed is it safe to assume that either the transmitted frame or its acknowledgement has been lost, and to send a duplicate.

The sender waits for some time for an event after transmitting the frame and starting the timer. Any of the three possible events can happen: an acknowledgment frame arrives correctly; a damaged acknowledgement frame arrives or the timer expires. If a valid acknowledgement comes in, the sender fetches the next packet from its network layer and puts it in the buffer, overwriting the previous packet. It also advances the sequence number. Instead, if a damaged frame arrives or the timer expires, the sender will send a duplicate frame without changing the buffer or the sequence number.

When a valid frame arrives at the receiver, its sequence number is checked to see if it is a duplicate. If not, it is accepted, passed to the network layer, and an acknowledgement is generated. Duplicate and damaged frames are not accepted at the receiver, but it results the last correctly received frame to be acknowledged to inform the sender to advance to the next frame or retransmit a damaged frame.

Self-Assessment Questions - 2

8. A _____ is an enumerated type and can take on the values true and false.
9. A _____ is the unit of information exchanged between the network layer and the data link layer.
10. A_____ is composed of four fields: kind, seq, ack, and info.
11. In _____ protocol, infinite buffer space is available and the communication channel between the data link layers never damages or loses frames.
12. is an unrealistic protocol because it doesn't handle flow control or error correction.
13. Protocols in which the sender sends one frame and then waits for an acknowledgement before proceeding is called _____.
14. Protocols in which the sender waits for a positive acknowledgement before advancing to the next data item are often called _____.

4. SUMMARY

Let us recapitulate the important concepts discussed in this unit:

- The data link layer has a number of functions, such as: providing a well-defined service interface to the network layer, dealing with transmission errors and flow control.
- The data link layer can be designed to offer various services such as: Unacknowledged connectionless service, acknowledged connectionless service and Acknowledged connection-oriented service.
- There are four different methods for framing. They are: byte count, flag bytes with byte stuffing, flag bits with bit stuffing and physical layer coding violations.
- Elementary datalink protocols are: simplex protocol, stop and wait protocol for an error free channel and stop and wait protocol for a noisy channel.
- In Simplex protocol, Data are transmitted in one direction only. Both the transmitting and receiving network layers are always ready. Processing time can be ignored. Infinite buffer space is available.
- Protocols in which the sender sends one frame and then waits for an acknowledgement before proceeding are called stop-and-wait.

5. TERMINAL QUESTIONS

1. What are the services provided to the network layer by data link layer? Explain.
2. Explain the four basic framing methods.
3. Describe data link layer error control and flow control.
4. Explain Simplex protocol.
5. Differentiate between stop and wait protocols for error free channels and Noisy channels.

6. ANSWERS

Self-Assessment Questions

1. Data link layer
2. Acknowledged connectionless service
3. Unacknowledged connectionless service
4. Wi-Fi (802.11)
5. Byte stuffing
6. Flag byte
7. Flow control
8. Boolean
9. Packet
10. Frame
11. Simplex protocol
12. Simplex protocol
13. Stop and Wait protocol
14. ARQ (Automatic Repeat request)

Terminal Questions

1. Major Service provided by the data link layer to the network layer is transferring data from the network layer on the source machine to the network layer on the destination machine. The data link layer can be designed to offer various services such as: Unacknowledged connectionless service, acknowledged connectionless service and Acknowledged connection-oriented service. (Refer section 2.1 for more details).
2. There are four basic framing methods, they are: *byte count, flag bytes with byte stuffing, flag bits with bit stuffing and physical layer coding violations.* The first framing method uses a field in the header to specify the number of bytes in the frame. At the receiver side, byte count is analyzed to find out the number of bytes in the frame and hence the end of the frame (Refer section 2.2 for more details)
3. At the destination, when data link layer receives bit stream from physical layer it is not guaranteed to be error free. Some bits may have different values and the number of bits

received may be less than, equal to, or more than the number of bits transmitted. It is the responsibility of data link layer to detect and correct errors. Flow control measures should be taken by data link layer, when there is a fast sender and slow receiver and sender wants to transmit frames faster than the receiver can accept them. (Refer section 2.3 and 2.4 for more details).

4. As the name specified, this protocol is as simple as it can be because it does not worry about the possibility of anything going wrong. Data is transmitted in one direction only. Both the transmitting and receiving network layers are always ready. Processing time can be ignored. Infinite buffer space is available. And best of all, communication channel between the data link layers never damages or loses frames. (Refer section 3.1 for more details).
5. In the Stop and Wait Protocol for an Error-Free Channel protocol, the receiver sends feedback to the sender. After having passed a packet to its network layer, the receiver sends a little dummy frame back to the sender which, in effect, gives the sender permission to transmit the next frame. The Stop and Wait Protocol for noisy channel protocol deals with the normal situation of a communication channel that makes errors. Frames can be either damaged or lost completely. If the frame is damaged in transit, the receiver hardware will detect this when computing the checksum. When compared to the protocol for error free channel, here we are using an additional timer and the sender could send a frame, but the receiver would only send an acknowledgement frame if the data were correctly received. (Refer section 3.2 and 3.3 for more details).

References:

- Andrew S. Tanenbaum, David J. Wetherall, "*Computer Networks*," Fifth edition.
- Larry L. Peterson, Bruce S. Davie, "*Computer Networks – a Systems Approach,"* Fifth edition.
- James F. Kurose, Keith W.Ross, "*Computer Networking – A top-down approach*," Sixth edition.
- Behrouz A. Forouzan, Sophia Chung Fegan, "*Data Communication and Networking*," Fourth edition.

- William Stallings, "*Computer Networking with Internet Protocols and Technology*," Third edition.