Study Material By Manikrao Dhore
Sub: Computer Networks

Section I: Logical Link Control

Course Outcome:

CO2: Estimate reliability issues based on error control, flow control and pipelining by using bandwidth, latency, throughput and efficiency

Unit-II Logical Link Control
[CO2 → PO1, PO2, PO3, PO5, PO12, PSO01 –CO Strength -3,2,2,2,3,3]

Logical Link Control: Design Issues: Services to Network Layer, **Framing**, Error Control: Parity Bits, Hamming Codes and CRC. Flow Control Protocols: Unrestricted Simplex, Stop and Wait, Sliding Window Protocol, WAN Connectivity: PPP and HDLC. PPPoE, PPPoA. Is DOCSIS used in 2022? Do we use DSL line in 2022? Do we use coaxial cable in 2022? Is PPP still used? [4 Hrs]

Functions of Data Link Layer

- Services to Network layer
- Framing: Break packets into frames according to the specifications of physical layer standard i.e. encapsulation and de-capsulation of data
- Error Control and Recovery for physical Channel
- Flow control: Provides reliable transfer of data across media.
- Physical addressing, network topology, error notification
- Link Management
- Collision Avoidance and collision content resolution

Design Issues:

I. Services Provided To The Network Layer
II. Framing
III. Error Control
IV. Flow Control

I. Services Provided To The Network Layer

Transfer of data from n/w layer of Source Machine to the N/w layer of DM

Process the data in bits.

1. Unacknowledged connectionless:

- Independent frames, receiving not in serial order
- No connection establishment before hand and released afterwards
- No acknowledgement
- No attempt to recover error
- Good when error rate is low
- Good for real time applications

2. Acknowledged connectionless

- No connection establishment before hand and released afterwards
- Acknowledgement
- Lost packets are sent again
- Good for wireless systems

3. Acknowledged connection oriented

- Establish connection before hand and releases afterwards
- acknowledgement
- guarantees the delivery of frames
- serial order
- Three Phases : Establish-Transfer-release

II. Framing

- To break bit streams into discrete frames
- Computer check sum for each frame at source
- Recomputed check sum for each frame at destination
- Detect errors and deal with them

- Discard bad frames and send back error report
- Insert time gap between frames
- Mark start and end of frame

1. Character count:

- Specify no. of characters in the frame in header.
- Change in header due to error will create a problem

2. Byte stuffing or Character Stuffing:

- Use of flag for start and end delimiter Flag -------Flag.
  Ex. DLE STX ---------data----------------DLE ETX
- If same sequence of flag occurs in the frame, it will create a problem

3. Bit stuffing

- Each frame begins with 01111110 and after each five consecutive 1's it inserts 0.

III. Error Control

- How to make sure that all frames are eventually delivered to the destination and received in proper order
- Destination sends + ack on receiving frame
- Destination sends –ve ack in case of wrong or lost frames
- If ack lost, sender will hang forever
- Uses timer: if frame is received correctly, ack will get back before timer runs out
- If frame is lost timer will go off
- Use of sequence numbers to know lost frames

	Bit Stuffing Examples
Ex1.	A bit string, 0101111101111110010, needs to be transmitted at the data link layer. What bit string is transmitted after bit stuffing? Assume the bit-stuffing.
Sol	01011111 **0** 011111 **0** 10010
Ex2	A bit string, 1101111111101111010, needs to be transmitted at the data link layer. What bit string is transmitted after bit stuffing?
Sol	11011111 **0** 11101111010
	Byte Stuffing on Field

Byte Oriented Protocols

●View frame as collection of bytes (not bits)

●Special byte acts as the sentinel

●Examples:

–BISYNC (Binary Synchronous Communication) developed by IBM

–DDCMP (Digital Data Communication Message Protocol)

SYN	SYN	SOH	Header	STX	Message	ETX	CRC

–PPP (Point-to-Point Protocol)

Ex. DLE STX A B C D DLE ETX

Ex. DLE STX A B DLE C D DLE ETX

Sol: DLE STX A B **DLE** DLE C D DLE ETX

Suppose the last two bytes are DLE ETX, what sequence of bytes precede CRC?

Ans: **DLE** DLE **DLE** ETX DLE ETX

IV. Flow Control

- Sender is fast and receiver is slow
- Requires a mechanism by which sender will not send the frames faster than receiver can handle

----

Error Correcting Codes

1. Parity bit (Even or odd no of 1's)

1011010 -> 10110100 - Even Parity

1011010 -> 10110101 - odd Parity

PC To PC Transfer (RS232-C/D)

2. Hamming Codes [Used for 1-bit error detection and correction]

1000 1001 Exclusive-OR / XOR Operation

1011 0001

0011 1000 – Hamming distance is three

n=m+r where m- message/data bits + r – redundant bits/check bits
and n-bit codeword for even parity
Hamming Code(7,4) means 7-bits code word and 4 bits message
Hamming Code works as follows for (n=m+r = 7=4+3)
Data bits are at positions 3, 5, 6, 7.
Check bits are at positions 1, 2, 4.
Check bit at position 1 is a parity of bits at positions 3, 5, 7 that is b3+b5+b7
check bit at position 2 is a parity of bits at positions 3, 6, 7 that is b3+b6+b7
check bit at position 4 is a parity of bits at positions 5, 6, 7. That is b3+b5+b7

Examples on Hamming Code:

Ex1 : If Data Word is = 1111 calculate the code word?

Solution:

1 2 3 4 5 6 7

Step-I: Initially: Insert the check bits as 0 at positions 1,2 and 4→ 0010111

Step-II: Calculate check bit at position 1.

Check bit at position 1 is a even parity of bits at positions 3, 5, 7;

Bits at positions 3,5,7 are 111, therefore check bit will be 1 to make it even parity.

Step-II: Calculate check bit at position 2.

Check bit at position 2 is a even parity of bits at positions 3, 6, 7;

Bits at positions 3,6,7 are 111, therefore check bit will be 1 to make it even parity.

Step-III: Calculate check bit at position 4.

check bit at position 4 is a even parity of bits at positions 5, 6, 7.

Bits at positions 5,6,7 are 111, therefore check bit will be 1 to make it even parity.

Therefore code word n = 1111111

Ex 2: If Data Word is = 1011 calculate the code word?

Solution:

1 2 3 4 5 6 7

Step-I: Initially: Insert the check bits as 0 at positions 1,2 and 4→ 0010011

Step-II: Calculate check bit at position 1.

Check bit at position 1 is a even parity of bits at positions 3, 5, 7;

Bits at positions 3,5,7 are 101, therefore check bit will be 0 to make it even parity.

Step-II: Calculate check bit at position 2.

Check bit at position 2 is a even parity of bits at positions 3, 6, 7;

Bits at positions 3,6,7 are 111, therefore check bit will be 1 to make it even parity.

Step-III: Calculate check bit at position 4.

check bit at position 4 is a even parity of bits at positions 5, 6, 7.

Bits at positions 5,6,7 are 011, therefore check bit will be 0 to make it even parity.

Therefore code word n = 0110011

Ex 3: Test if 0110111 code word is correct, assuming they were created using an even parity Hamming Code .If one is incorrect, indicate what the correct code word should have been. Also, indicate what the original data was.

Solution:

1 2 3 4 5 6 7

Re-Check each parity bit using

Check bit at position 1 is an even parity of bits at positions 1, 3, 5, 7 that is b1+b3+b5+b7

check bit at position 2 is an even parity of bits at positions 2, 3, 6, 7 that is b2+b3+b6+b7

check bit at position 4 is an even parity of bits at positions 4, 5, 6, 7 that is b4+b5+b6+b7

Step-I: Re-check check bit at position 1

0110111 : b1+b3+b5+b7 = 0111, odd parity : incorrect

Step-II: Re-check check bit at position 2

0110111 : b2+b3+b6+b7 =1111, even parity: correct

Step-II: Re-check check bit at position 4

0110111 : b4+b5+b6+b7 = 0111, odd parity : incorrect

Parity bits 1 and 4 are incorrect

1 + 4 = 5, so the error occurred in bit 5

Therefore code word n = 0110011

Hamming Code(8,4) [Source: http://users.cis.fiu.edu/~downeyt/cop3402/hamming.html]

1. Mark all bit positions that are powers of two as parity bits. (positions 1, 2, 4, 8, 16, 32, 64, etc.)
2. All other bit positions are for the data to be encoded. (positions 3, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 17, etc.)
3. Each parity bit calculates the parity for some of the bits in the code word. The position of the parity bit determines the sequence of bits that it alternately checks and skips.
   Position 1: check 1 bit, skip 1 bit, check 1 bit, skip 1 bit, etc. (1,3,5,7,9,11,13,15,...)
   Position 2: check 2 bits, skip 2 bits, check 2 bits, skip 2 bits, etc. (2,3,6,7,10,11,14,15,...)
   Position 4: check 4 bits, skip 4 bits, check 4 bits, skip 4 bits, etc. (4,5,6,7,12,13,14,15,20,21,22,23,...)
   Position 8: check 8 bits, skip 8 bits, check 8 bits, skip 8 bits, etc. (8-15,24-31,40-47,...)
   Position 16: check 16 bits, skip 16 bits, check 16 bits, skip 16 bits, etc. (16-31,48-63,80-95,...)
   Position 32: check 32 bits, skip 32 bits, check 32 bits, skip 32 bits, etc. (32-63,96-127,160-191,...)
   etc.
4. Set a parity bit to 1 if the total number of ones in the positions it checks is odd. Set a parity bit to 0 if the total number of ones in the positions it checks is even.

Here is an example:

A byte of data: 10011010
Create the data word, leaving spaces for the parity bits: _ _ 1 _ 0 0 1 _ 1 0 1 0
Calculate the parity for each parity bit (a ? represents the bit position being set):

- Position 1 checks bits 1,3,5,7,9,11:
  ? _ 1 _ 0 0 1 _ 1 0 1 0. Even parity so set position 1 to a 0: 0 _ 1 _ 0 0 1 _ 1 0 1 0
- Position 2 checks bits 2,3,6,7,10,11:
  0 ? 1 _ 0 0 1 _ 1 0 1 0. Odd parity so set position 2 to a 1: 0 1 1 _ 0 0 1 _ 1 0 1 0

- Position 4 checks bits 4,5,6,7,12:
  0 1 1 ? 0 0 1 _ 1 0 1 0. Odd parity so set position 4 to a 1: 0 1 1 1 0 0 1 _ 1 0 1 0
- Position 8 checks bits 8,9,10,11,12:
  0 1 1 1 0 0 1 ? 1 0 1 0. Even parity so set position 8 to a 0: 0 1 1 1 0 0 1 0 1 0 1 0
- Code word: 011100101010.

Finding and fixing a bad bit

The above example created a code word of 011100101010. Suppose the word that was received was 011100101110 instead. Then the receiver could calculate which bit was wrong and correct it. The method is to verify each check bit. Write down all the incorrect parity bits. Doing so, you will discover that parity bits 2 and 8 are incorrect. It is not an accident that 2 + 8 = 10, and that bit position 10 is the location of the bad bit. In general, check each parity bit, and add the positions that are wrong, this will give you the location of the bad bit.

Test if **011100101110** code word is correct, assuming they were created using an even parity Hamming Code .If one is incorrect, indicate what the correct code word should have been. Also, indicate what the original data was.

Step 1 011100101110

For position 1 check bits(1,3,5,7,9,11) even parity 1 is set 0. Position 1 is correct

Step 2 011100101110

For position 2 check bits (2,3,6,7,10,11) odd incorrect for even set it to 1

Step 3 011100101110

For position 4 check bits are (4,5,6,7,12,13,14,15,20,21,22,23)

Even correct.

Step 4 011100101110

For position 8 check bits are (8-15,24-31,40-47)

Odd ..incorrect

you will discover that parity bits 2 & 8 are incorrect .

2+8 =10, and that bit position is the location of the bad bit.

Original code word - **011100101010**

Error Correcting Codes: Cyclic Redundancy Check (CRC)

M = k bit messages, **F** = n bits frame check sequence (FCS)

T = (k + n) bits frame where n<k, **P** = n+1 is the predermined divisor

We want mod (T,P) = 0

Rewrite T = M 2^n + F

F = M 2^n / P

Ex: M = 1010001101 (10 bits)

P = 110101 (6 bits)

P= 6 hence n=5 bits F = 5 bits to be calculated

M 2^n = M 2^5 =101000110100000

2^5 M / P = 101000110100000 / 110101

Method –I : Modulo-2 Arithmetic

110101	1	0	1	0	0	0	1	1	0	1	0	0	0	0	0	Q
	1	1	0	1	0	1										1
	x	1	1	1	0	1										
		1	1	1	0	1	1									
		1	1	0	1	0	1									1
		x	0	1	1	1	0									
			0	1	1	1	0	1	nd							
				0	0	0	0	0								0
				1	1	1	0	1	0							
				1	1	0	1	0	1							1
				x	0	1	1	1	1	1						
					0	0	0	0	0	0						0
						1	1	1	1	1	0					
						1	1	0	1	0	1					1
						x	0	1	0	1	1	0				
							0	0	0	0	0	0				0
							x	1	0	1	1	0	0			
								1	1	0	1	0	1			1

								x	1	1	0	0	1	0		
									1	1	0	1	0	1		1
	F= Remainder = R								x	x	0	1	1	1	0	0

Therefore F = 01110

T = M 2^n + F

101000110100000

+ 01110

101000110101110

At the receiver , it receives T =101000110101110

At receiving end, it divides T by same P =110101

110101	1	0	1	0	0	0	1	1	0	1	0	1	1	1	0	Q
	1	1	0	1	0	1										1
	x	1	1	1	0	1										
		1	1	1	0	1	1									
		1	1	0	1	0	1									1
		x	0	1	1	1	0									
			0	1	1	1	0	1	nd							
				0	0	0	0	0								0
				1	1	1	0	1	0							
				1	1	0	1	0	1							1
				x	0	1	1	1	1	1						
					0	0	0	0	0	0						0
						1	1	1	1	1	0					
						1	1	0	1	0	1					1
						x	0	1	0	1	1	1				
							0	0	0	0	0	0				0
							x	1	0	1	1	1	1			
								1	1	0	1	0	1			1
								x	1	1	0	1	0	1		
									1	1	0	1	0	1		1
									0	0	0	0	0	0		
	F= Remainder = R								x	x	0	0	0	0	0	0

If remainder is 0 then frame T is received correctly

Method –II : Polynomial Method

Message = M(x) = (k+n) -bits

Divisor = G(x) = P = n+1 bits

F= g(x) = x^c = n-bits

$G(x) \overline{)M(x) * x^c} (Q(x)$

T(x) = (k+n) bits

Ex. M(x) = $x^6 + x^4 + x^3 +1$

G(x) = $x^3 + 1$

g(x) = x^3

Therefore M(x) * x^c

= $x^9 + x^7 + x^6 + x^3$

$x^3 + 1$) $x^9 + x^7 + x^6 + x^3$ ($x^6 + x^4 + x + 1$

$x^9 + \quad + x^6$

$x^7 + \quad + x^3$

$x^7 + \quad + x^4$

$x^4 + x^3$

$x^4 + x$

$x^3 + x$

$x^3 + 1$

Remainder R=F $x + 1$

$T(x) = x^9 + x^7 + x^6 + x^3 + x + 1$

At receiver

$\quad\quad$ ------------------------------

$x^3 + 1\ )\ x^9 + x^7 + x^6 + x^3 + x + 1(x^6 + x^4 + x + 1$

$\quad\quad x^9 + \quad + x^6$

$\quad\quad\quad x^7 + \quad + x^3$

$\quad\quad\quad x^7 \quad + \quad x^4$

$\quad\quad\quad\quad x^4 + x^3 + x$

$\quad\quad\quad\quad x^4 + x$

$\quad\quad\quad\quad\quad x^3 + 1$ and finally 0

Definitions to Simulate protocols

void wait_for_event(event_type *event)

void from_nw_layer(packet *p) and void to_nw_layer(packet *p)

void from_phy_layer(frame *r) and void to_phy_layer(frame *r)

void start_timer(seq_nr, k) and void stop_timer(seq_nr, k)
void start_ack_timer and void stop_ack_timer
void enable_nw_layer and void disable_nw_layer

I. Unrestricted Simplex Protocol

- Very simple protocol
- Data transmission in one direction only (either sender or receiver)
- Tx and Rx always ready
- Infinite buffer space available
- Communication Channel is error free

```
Typedef enum{frame_arrival} event_type;
void sender(void)
{
    frame s
    packet buffer;

    while(true)
    {
       from_nw_layer(&buffer);
       s.info = buffer;
       to_phy_layer(&s);
    }
}

void receiver(void)
{
    frame r
    event_type event

    while(true)
    {
       wait_for_event(&event)
       from_phy_layer(&r);
       to_nw_layer(&r.info);
    }
}
```

II. Simplex Stop-and-Wait protocol

- Data transmission in one direction only (either sender or receiver)
- finite buffer space available

- Communication Channel is error free
- Prevent the fast sender, sender has to *wait for acknowledgement* before proceeding to send the next packet.

```
typedef enum{frame_arrival} event_type;

void sender(void)
{
    frame s
    packet buffer;
    event_type event

    while(true)
    {
       from_nw_layer(&buffer);
       s.info = buffer;
       to_phy_layer(&s);
      wait_for_event(&event)
    }
}

void receiver(void)
{
    frame r,s    s is ack
    event_type event

    while(true)
    {
       wait_for_event(&event)
       from_phy_layer(&r);
       to_nw_layer(&r.info);
```

```
            to_phy_layer(&s);
        }
    }
```

Simplex for noisy channel

- Data transmission in one direction only (either sender or receiver)
- finite buffer space available
- Communication Channel is **not error free**
- Sender sends a frame
- Receiver sends ack if frame is received correctly otherwise no ack in case of damged frame
- Use of timer. If timer runs out, resend the frame

Worst case:

If ack get lost, sender timer will time out and sender will send a same frame again and there will be a duplication of data at receiver end.

Sol : Use the sequence number : 0 and 1

Use the fig in notebook for explanation.

Frame 0

Ack 1

Frame 1

Ack 0

Frame 0 --- lost

Time out at sender

Frame 0

Ack1

Frame 1

Ack0 --- lost

Frame 1 --- receiver will discard the duplicate frame 1

Sliding window (1-bit)

- Data transmission in both direction ie **full duplex communication**

- finite buffer space available
- Communication Channel is **not error free**
- Use of timer
- Sender sends a frame
- Receiver sends ack along with its if frame

(seq no, ack, frame no)

Receiver sends ack as a seq no of frame received.

Normal Scenario: Timers are synchronized

B has waited till the first frame from sender A to receive to synchronize the ack

"*" asterisk indicates frame delivered at network layer.

A sends (0,1,A0)	B gets(0,1,A0)* B sends (0,0,B0)
A gets(0,0,B0)* A sends (1,0,A1)	B gets(1,0,A1)* B sends (1,1,B1)
A gets(1,1,B1)* A sends (0,1,A2)	B gets(0,1,A2)* B sends (0,0,B2)
A gets(0,0,B2)* A sends (1,0,A3)	B gets(1,0,A3)* B sends (1,1,B3)

Abnormal Scenario: Timers are premature timeouts

Both A and B started sending their frames at the same time. A's time out interval is too little. Retransmission of duplicate frames has happened.

A sends (0,1,A0)	*B send (0,1,B0)* B gets(0,1,A0)* *B sends (0,0,B0)*
A gets(0,1,B0)* *A sends (0,0,A0)*	B gets(0,0,A0) *B sends (1,0,B1)*
A gets(0,0,B0) *A sends (1,0,A1)*	B gets(1,0,A1)* *B sends (1,1,B1)*
A gets(1,0,B1)* *A sends (1,1,A1)*	B gets(1,1,A1) B sends (1,1,B2)

Why Sliding Window Protocol?

Let a = Propagation Time / Transmission Time

If a>1 that is propagation time is greater than transmission time, then solution is to allow multiple frames to be in transit at a time.

Example.

BW = 50 Kbps

RTT = 500 ms

Frame Size = 1000 bits

Therefore to transfer 1000 bits it requires only 20 ms

Transmission will start at t=0 and at t=20 ms it will be over,

But to send next packet it requires ack which will be available after 500 ms.

It means that there would be only 4% utilization of line or sender is blocked for 96% of time.

It indicates that long transit time, high BW and short frame affects the efficiency.

Hence it would be better to have window size equal to 25 frames.
Therefore sender will get acknowledgement for frame 0 after sending 25 frames.
This technique is known as pipelining.
Sequence number = k-bits = 2^k
Window size = 2^k-1

If k=3 then seq numbers are : 0,1,2,3,4,5,6,7 ie 8
Window size = 7
Pipelining Scenarios: Go back n

0	1	2	3	4	5	6	7	8	2	3	4	5	6	7	8

0	1	Error	D	D	D	D	D	D	2	3	4	5	6	7	8

Pipelining Scenarios: Window size is large 0r Selective repeat

0	1	2	3	4	5	2	6	7	8	9	10	11	12	13	14

0	1	Error	3	4	5	2	6	7	8	9	10	11	12	13	14

Sliding Window
Use Diagram from Tananbaum Book

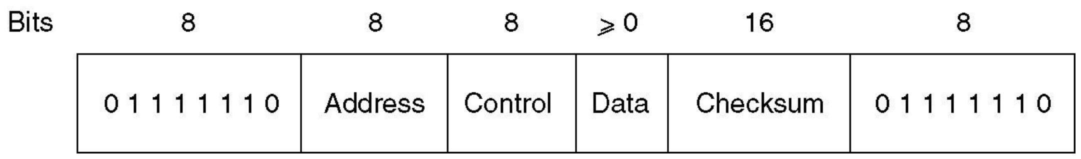
HDLC (High Level Data Link Control)

DLL LLC Protocol

Classical bit-oriented protocol

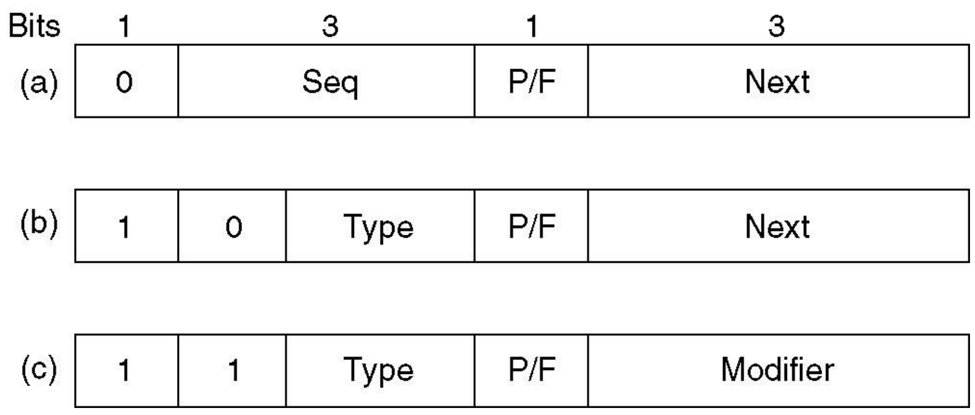
Used for point to point and point to multipoint

Uses bit stuffing



Control Frames are

1.Information frame
2.Supervisory Frame
3.Unnumbered Frame



Fields in the frame are

Flags – 8-bit flag for boundary

Address – Only for multicast : 1111 1111

Control –

Information Frame : For User Data

3-bit sequence number

P/F – Poll/Final

P- To invite data

All frames consists P and last frame consists F

Next is the frame number(sequence number) of next frame

Supervisory Frame : For flow control

Type : 0,1,2,3

Type 0 : is an ack frame

Type 1: -ve ack

Type 2: receiver not ready due to limited buffer

Type 3: selective repeat

Unnumbered Frame : Error Control

Is used if frame is lost

Commands

DISC – Disconnect

SNRM(Set Normal Response mode) : For asymmetric communication. For master slave, for dumb terminal

SABM(set Asynchronous balance mode) : For Synchronous communication

FRMR(Frame Reject) – Correct checksum but semantic problem

Uses on Field

HDLC SUBSET	USES
NRM (Normal Response Mode)	Multipoint networks that typically use SDLC
LAP (Link Access Procedure)	Early X.25 implementations
LAPB (Link Access Procedure, Balanced)	Current X.25 implementations
LAPD (Link Access Procedure for the	ISDN D channel and frame

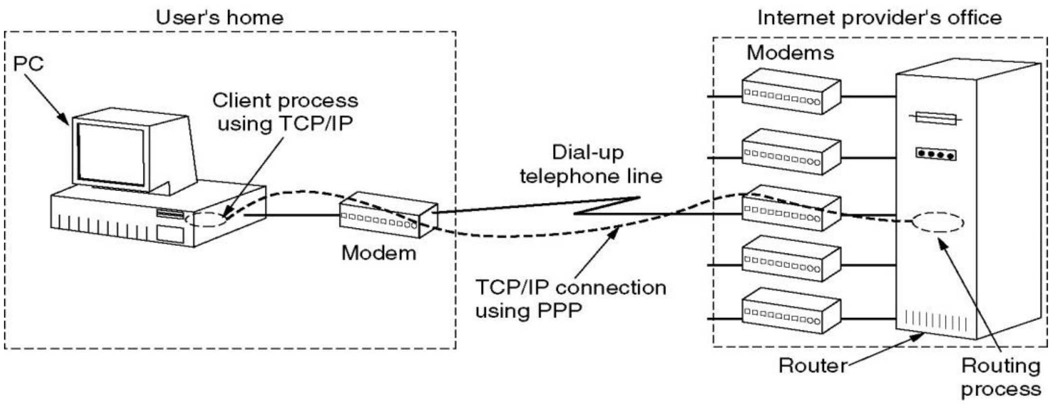
Integrated Services Digital Network D channel)	relay
LAPM (Link Access Procedure for Modems)	Error-correcting modems (specified as part of V.42)

DLL in the Internet

PC->TCP/IP Client Process_>modem->Telephone Line->modem->routers->ISP

Example: Point to point backbone in LAN

Dial up Internet connection at home


User's home
PC
Client process using TCP/IP
Modem
Dial-up telephone line
TCP/IP connection using PPP
Internet provider's office
Modems
Router
Routing process

PPP(Point-To Point Protocol)

Byte Oriented Protocol

Only for Point-to-Point

Bytes	1	1	1	1 or 2	Variable	2 or 4	1
	Flag 01111110	Address 11111111	Control 00000011	Protocol	Payload	Checksum	Flag 01111110

Fields are:

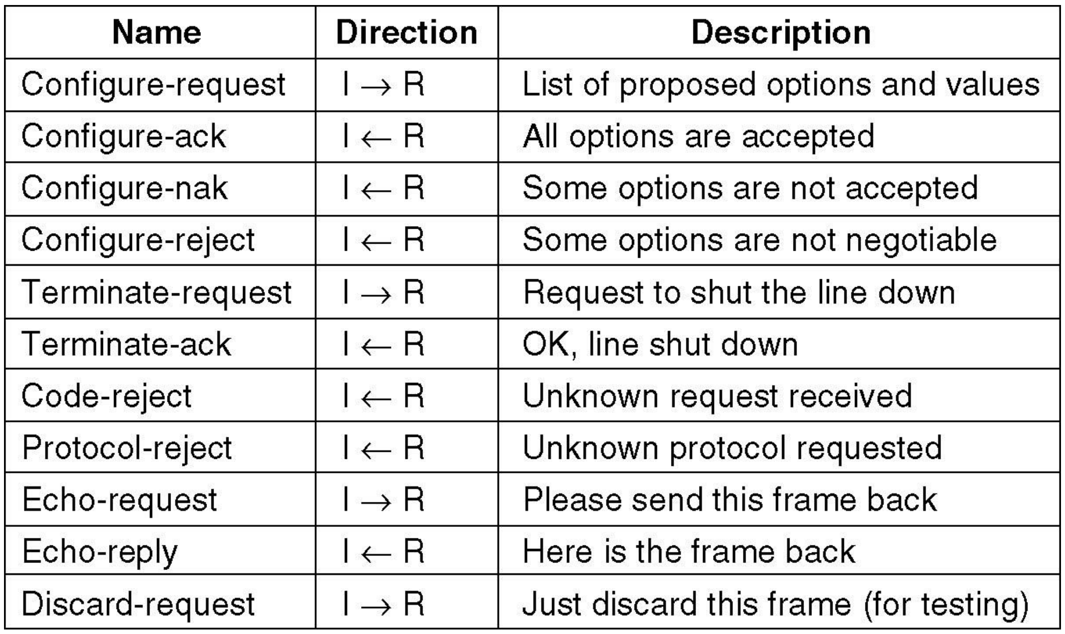
Flag : To mention the boundaries

Address : for all stations

Control Field : default value is 00000011. It indicates unnumbered frame. It is not reliable protocol using sequence numbers.

Protocol : LCP, NCP, IP, IPX

Uses **LCP** (link Control Protocol) : Bringing line up, testing them, negotiating options and bringing them down



Name	Direction	Description
Configure-request	$I \rightarrow R$	List of proposed options and values
Configure-ack	$I \leftarrow R$	All options are accepted
Configure-nak	$I \leftarrow R$	Some options are not accepted
Configure-reject	$I \leftarrow R$	Some options are not negotiable
Terminate-request	$I \rightarrow R$	Request to shut the line down
Terminate-ack	$I \leftarrow R$	OK, line shut down
Code-reject	$I \leftarrow R$	Unknown request received
Protocol-reject	$I \leftarrow R$	Unknown protocol requested
Echo-request	$I \rightarrow R$	Please send this frame back
Echo-reply	$I \leftarrow R$	Here is the frame back
Discard-request	$I \rightarrow R$	Just discard this frame (for testing)

NCP (Network Control Protocol) : to negotiate network layer options.

Payload

Example:

PC calls ISP via router

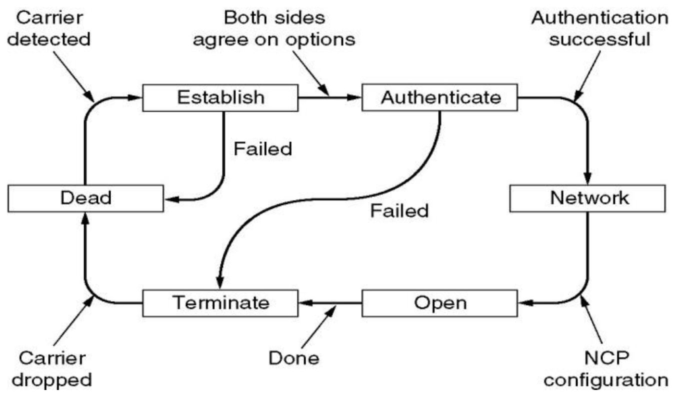
Router modem answers

PC sends LCP packets with options :Cfg – request, ack(All options accepted), nak(some options not accepted), reject(Some options are not negotiable)

PC sends NCP packet for getting IP through DHCP

State Diagram

Dead->Establish(or reject)->Authenticate(or terminate)->NW->open->Terminate_>dead


Carrier detected
Both sides agree on options
Authentication successful
Establish
Authenticate
Failed
Dead
Network
Failed
Terminate
Open
Carrier dropped
Done
NCP configuration