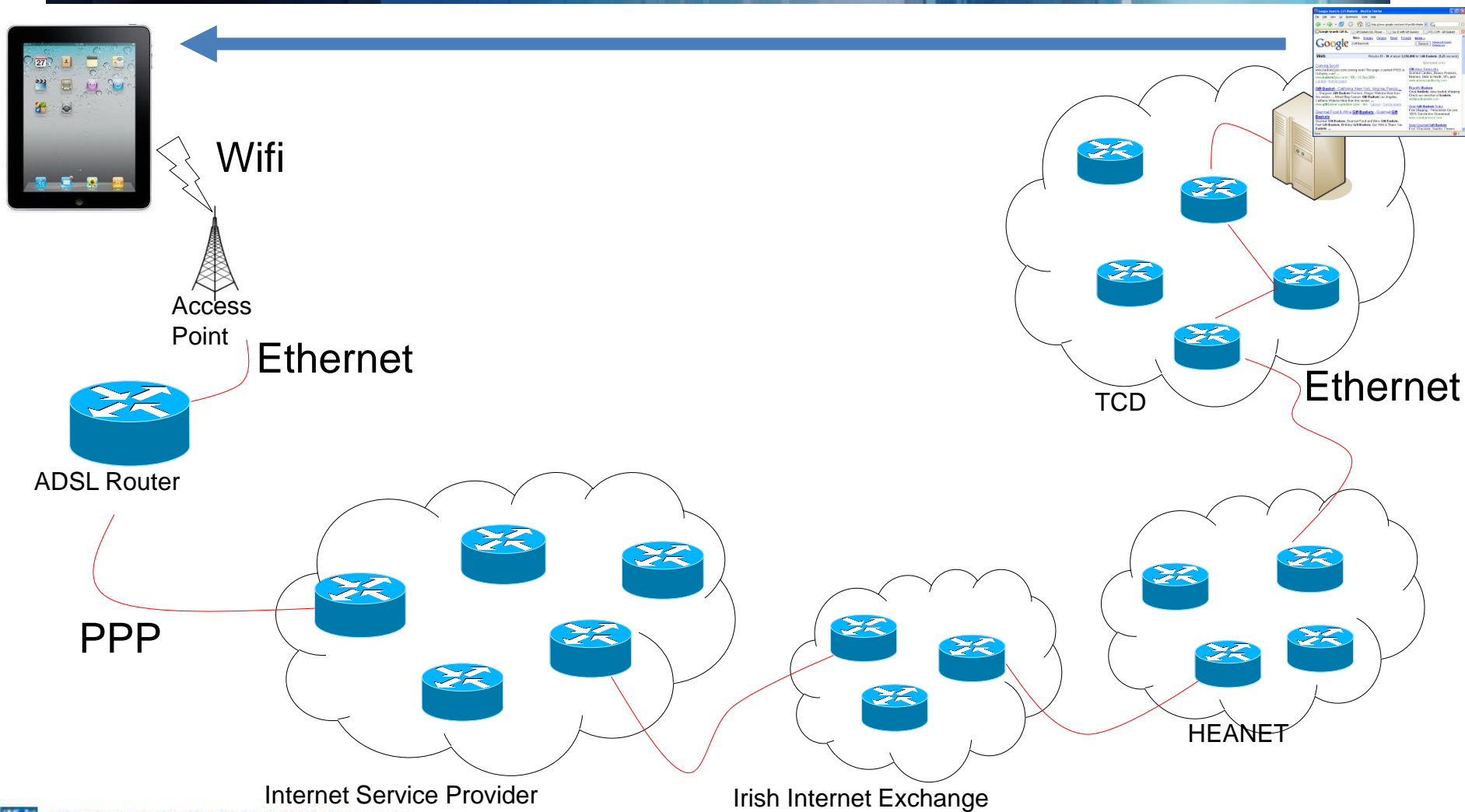
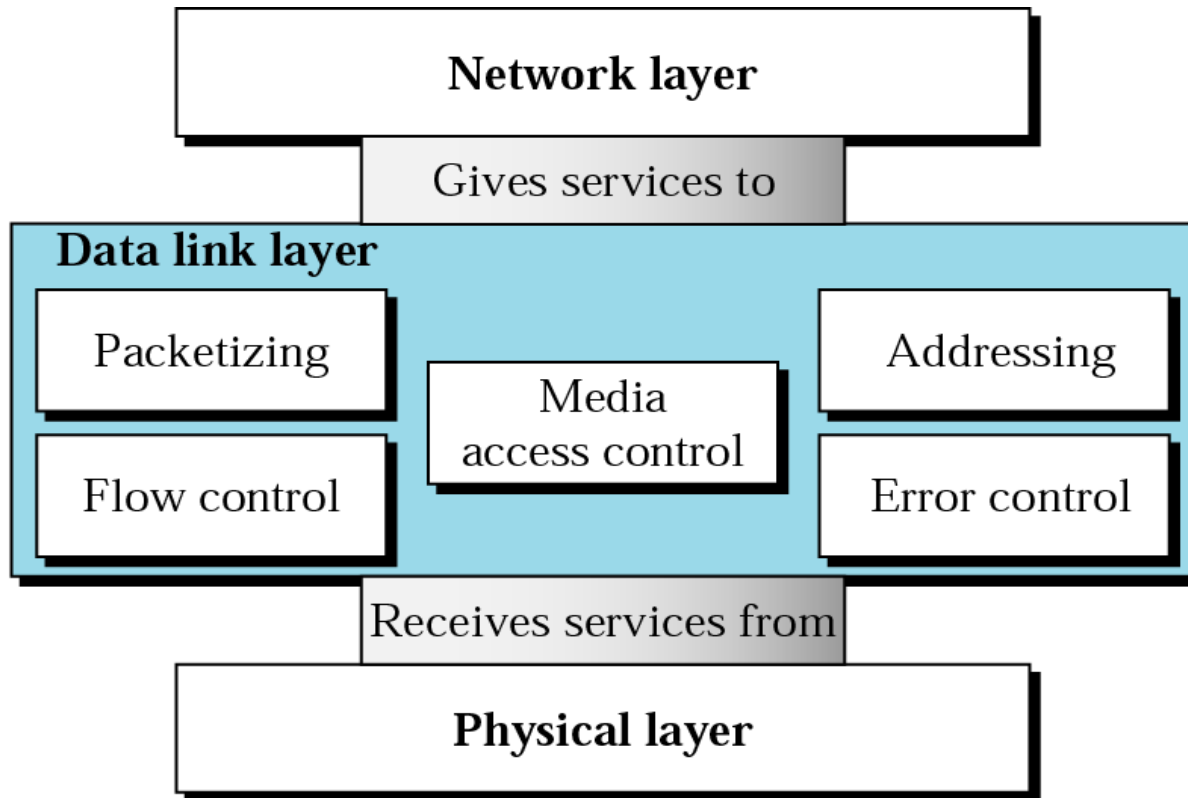


Goal



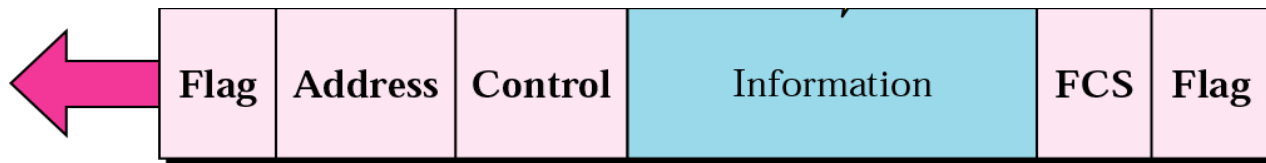
Duties of the Data Link Layer



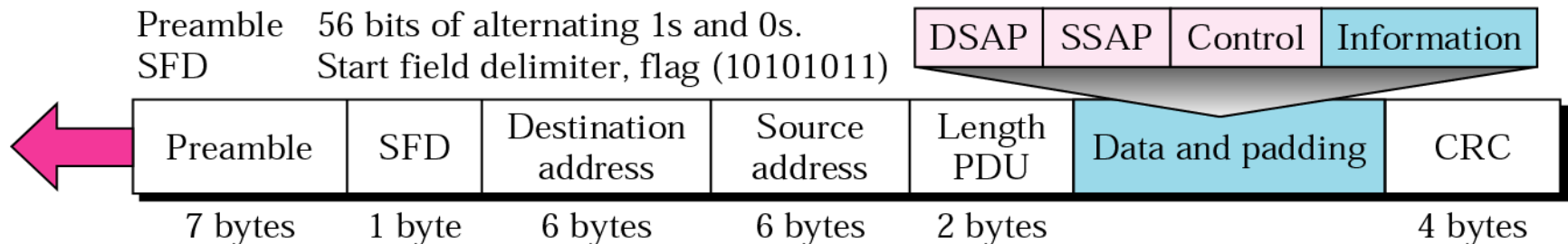
The data link layer is responsible for transmitting frames from one node to the next.

Example: HDLC & Ethernet

HDLC



Ethernet



Ethernet 802.3 Checksum

3.2.8 Frame Check Sequence (FCS) field

A cyclic redundancy check (CRC) is used by the transmit and receive algorithms to generate a CRC value for the FCS field. The frame check sequence (FCS) field contains a 4-octet (32-bit) cyclic redundancy check (CRC) value. This value is computed as a function of the contents of the source address, destination address, length, LLC data and pad (that is, all fields except the preamble, SFD, FCS, and extension). The encoding is defined by the following generating polynomial.

$$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

Mathematically, the CRC value corresponding to a given frame is defined by the following procedure:

- a) The first 32 bits of the frame are complemented.
- b) The n bits of the frame are then considered to be the coefficients of a polynomial $M(x)$ of degree $n-1$. (The first bit of the Destination Address field corresponds to the $x^{(n-1)}$ term and the last bit of the data field corresponds to the x^0 term.)
- c) $M(x)$ is multiplied by x^{32} and divided by $G(x)$, producing a remainder $R(x)$ of degree ≤ 31 .
- d) The coefficients of $R(x)$ are considered to be a 32-bit sequence.
- e) The bit sequence is complemented and the result is the CRC.

The 32 bits of the CRC value are placed in the frame check sequence field so that the x^{31} term is the left-most bit of the first octet, and the x^0 term is the right most bit of the last octet. (The bits of the CRC are thus transmitted in the order $x^{31}, x^{30}, \dots, x^1, x^0$.) See reference [B37].

Ethernet 802.3 Checksum

3.2.8 Frame Check Sequence (FCS) field

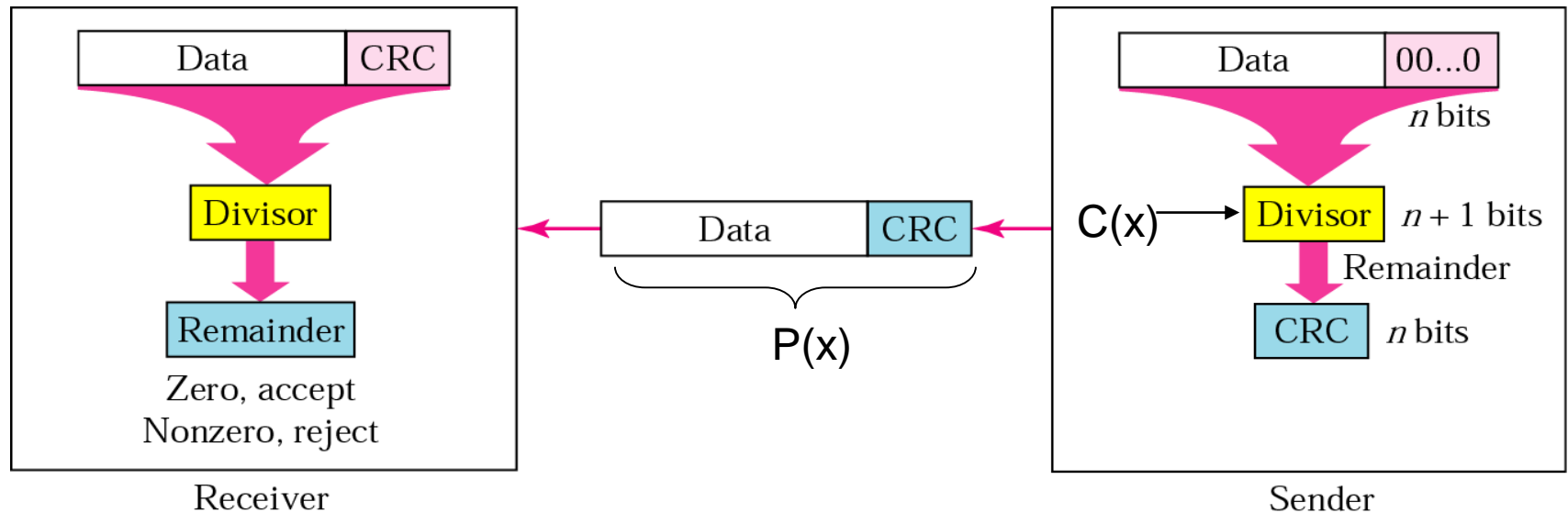
A cyclic redundancy check (CRC) is used by the transmit and receive algorithms to generate a CRC value for the FCS field. The frame check sequence (FCS) field contains a 4-octet (32-bit) cyclic redundancy check (CRC) value. This value is computed as a function of the contents of the source address, destination address, length, LLC data and pad (that is, all fields except the preamble, SFD, FCS, and extension). The encoding is defined by the following generating polynomial.

$$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

Review: Error Detection

- Types of Errors
 - Single-Bit & Burst Errors
- Detection of Errors
 - Parity Check / 2D Parity Check
 - CRC (Sequence of data)
 - Checksum (Chunks of data)
- Correction of Errors
 - Error Correction by Retransmission
 - Forward Error Correction – Hamming Code

Cyclic Redundancy Check (CRC)

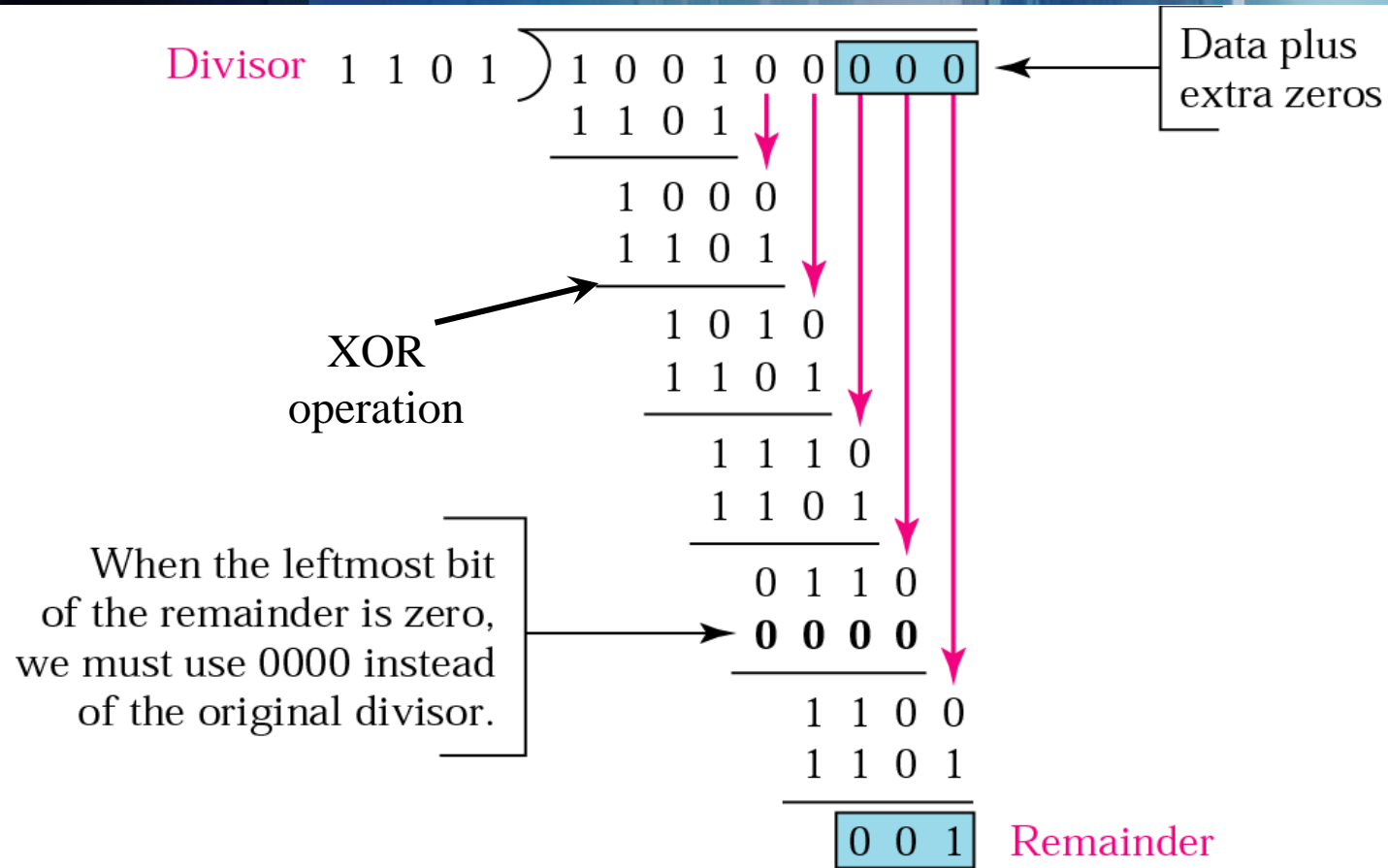


- $P(x)$ divided by $C(x) = 0$
- $(P(x) + \text{remainder})$ divided by $C(x)$ should be $\neq 0$

Standard Polynomials

Name	Polynomial	Application
CRC-8	$x^8 + x^2 + x + 1$	ATM header
CRC-10	$x^{10} + x^9 + x^5 + x^4 + x^2 + 1$	ATM AAL
CRC-16	$x^{16} + x^{12} + x^5 + 1$	HDLC
CRC-32	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$	LANs

CRC: Sender

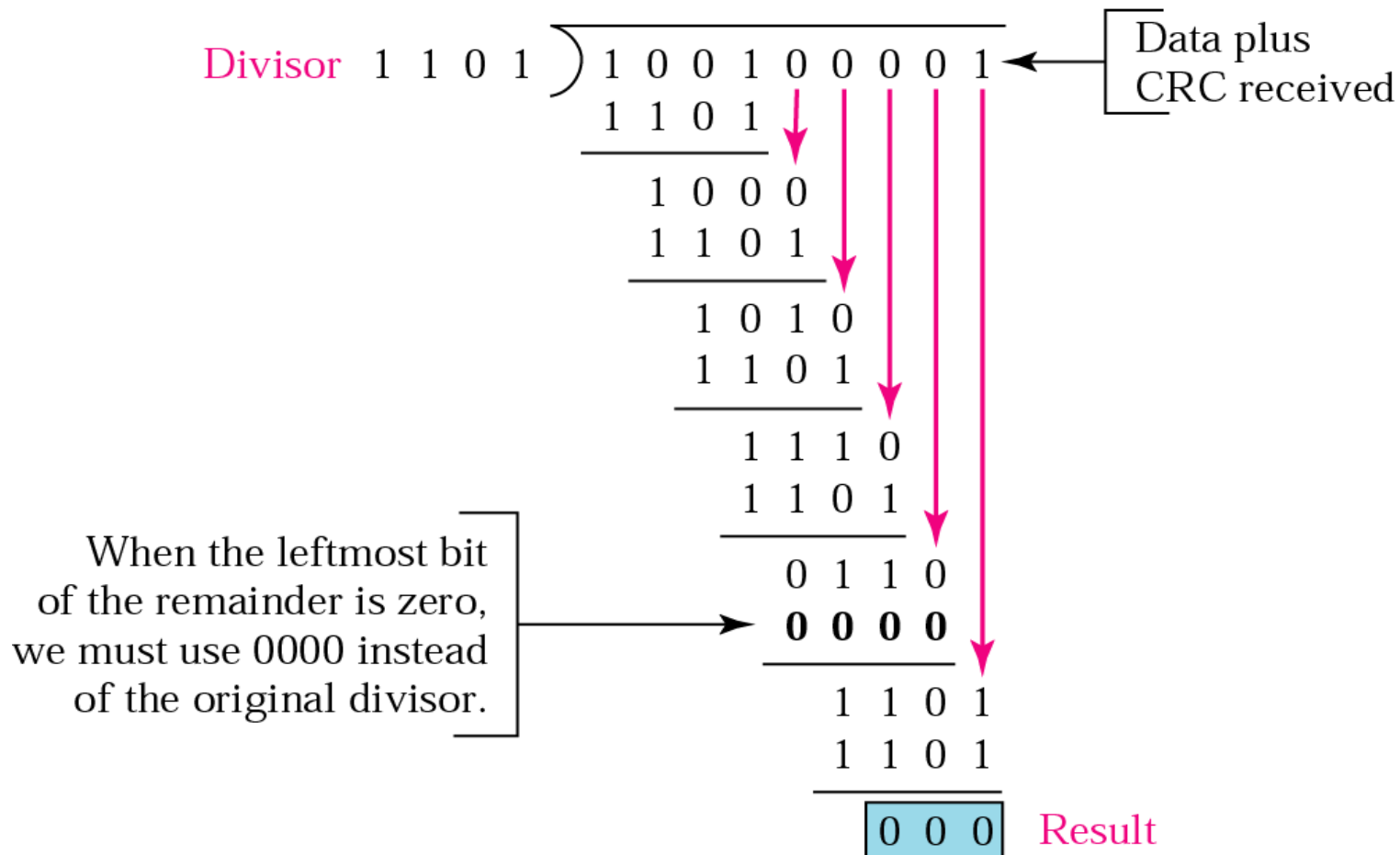


Data transmitted to receiver: 1 0 0 1 0 0 0 0 1

Data CRC

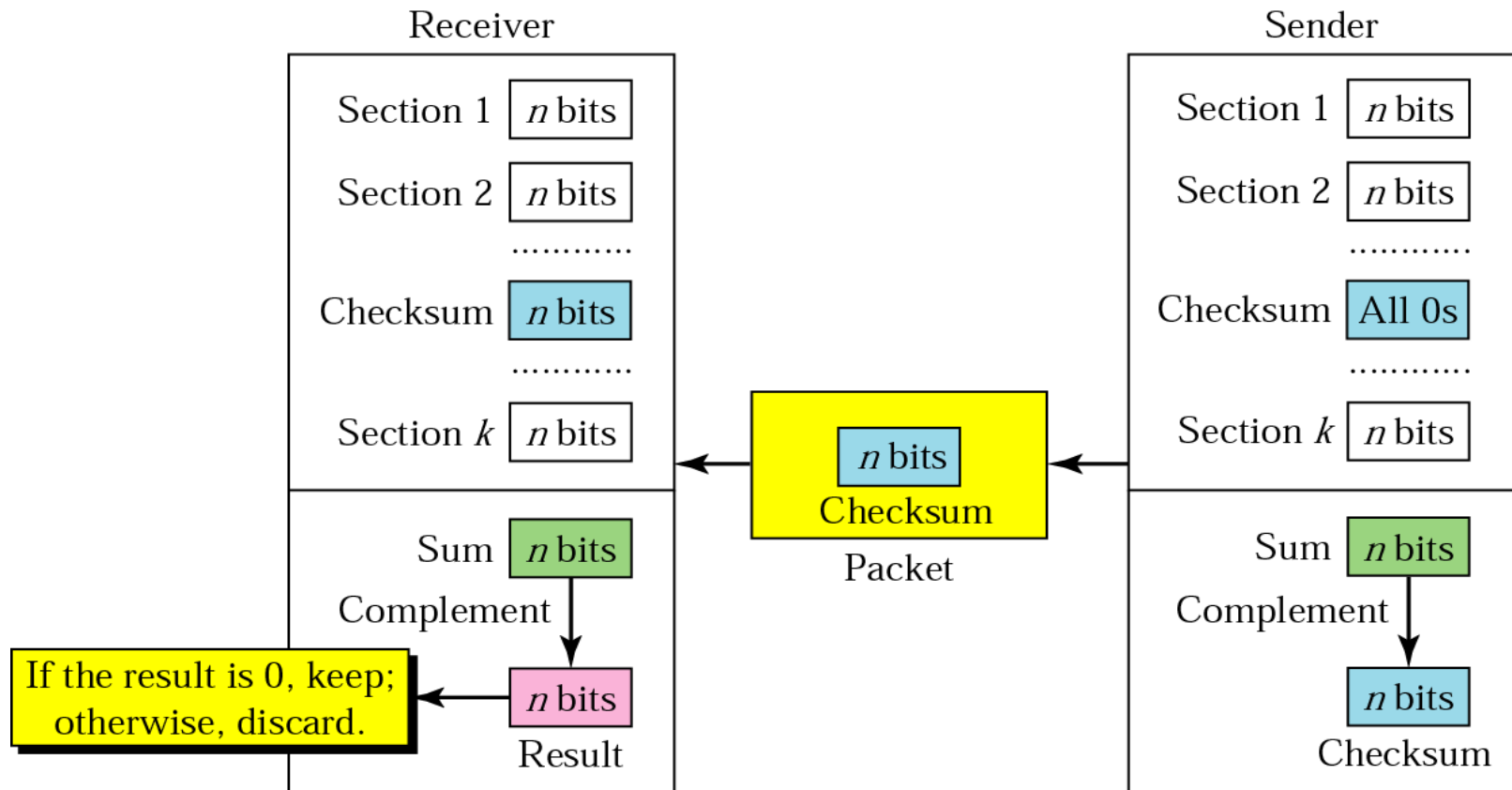
* Figure is courtesy of B. Forouzan

CRC: Receiver



* Figure is courtesy of B. Forouzan

Checksum



* Figure is courtesy of B. Forouzan

Checksum II

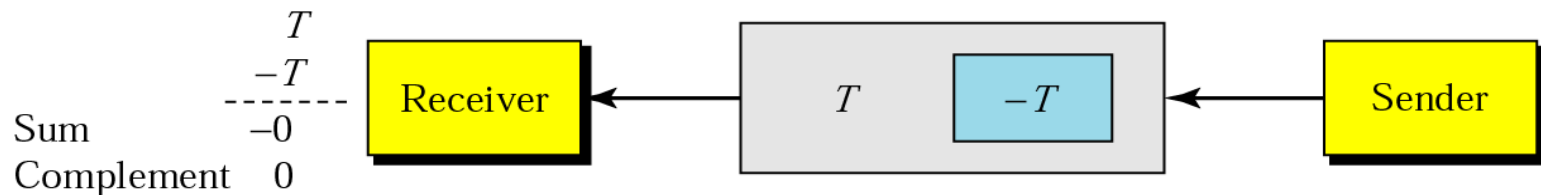
Sender:

The unit is divided into k sections, each of n bits.

All sections are added using one's complement to get the sum.

The sum is complemented and becomes the checksum.

The checksum is sent with the data.



Receiver:

The unit is divided into k sections, each of n bits.

All sections are added using one's complement to get the sum.

The sum is complemented.

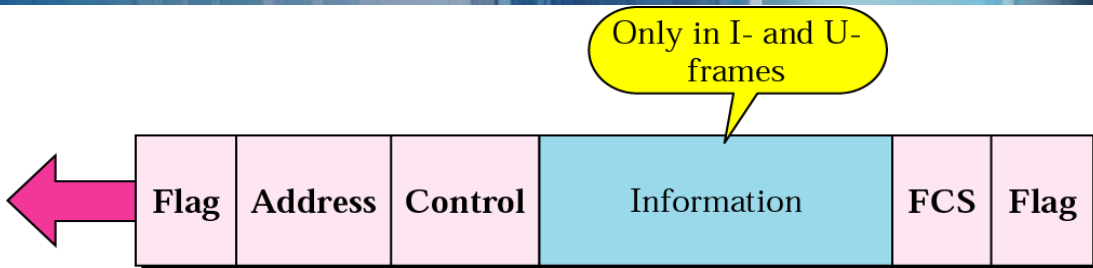
If the result is zero, the data are accepted: otherwise, rejected.

* Figure is courtesy of B. Forouzan

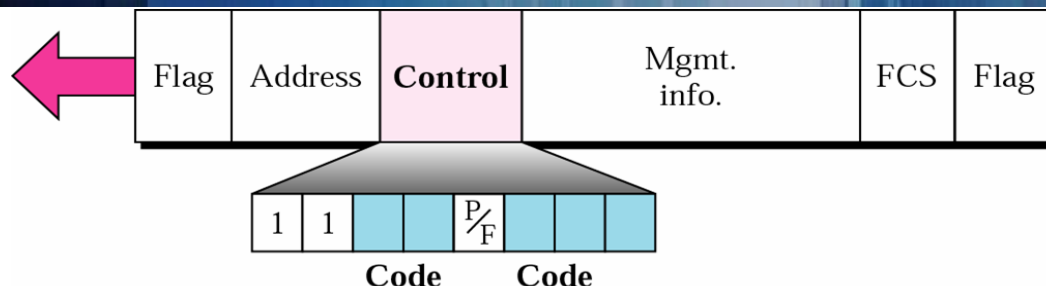
Review: HDLC

- Three Frame Types
 - I-Frame: Information Transfer Format
 - S-Frame: Supervisory Format
 - U-Frame: Unnumbered Format
- Implements Flow Control & Error Control mechanisms
 - Stop-And-Wait
 - Go-Back-N
 - Selective Repeat
- Bit-Stuffing - to avoid confusion of data and flag

HDLC Frame

- Flag= 01111110
 - Specifies beginning and end of frame
 - Address
 - Specifies secondary station as either sender or receiver
 - Control
 - Specifies type of frame and seq.&ack. number
 - Frame Check Sequence (FCS)
 - Either 16- or 32-bit CRC
- 

U-Frame Control Field

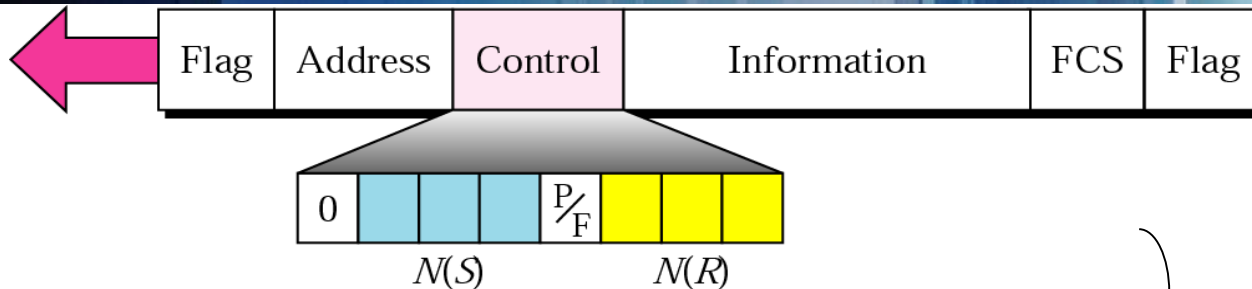


Code	Command/Response	Meaning
00 001	SNRM	Set normal response mode
11 100	SABM	Set asynchronous balanced mode
00 100	UP	Unnumbered poll
00 000	UI	Unnumbered information
00 110	UA	Unnumbered acknowledgment
00 010	DISC	Disconnect
10 000	SIM	Set initialization mode
11 001	RSET	Reset
11 101	XID	Exchange ID
10 001	FRMR	Frame reject

Managing Connection

* Figure is courtesy of B. Forouzan

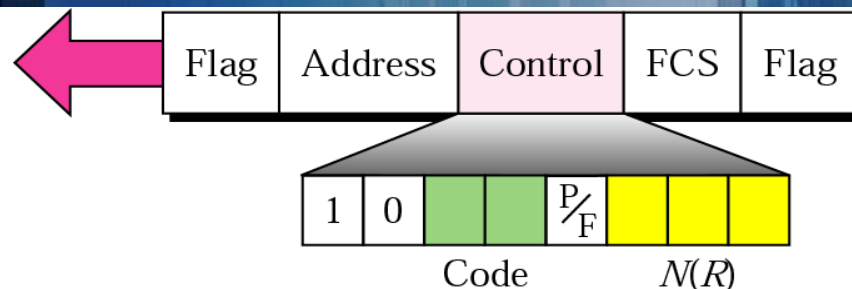
I-Frame



- $N(S)$
 - Sequence **N**umber of **S**ender
- $N(R)$
 - Sequence **N**umber of **R**eceiver
- P/F
 - Poll/Final bit
 - Primary Station: Request for information
 - Secondary Station: response or final frame

Information
Transfer

S-Frame Control Field

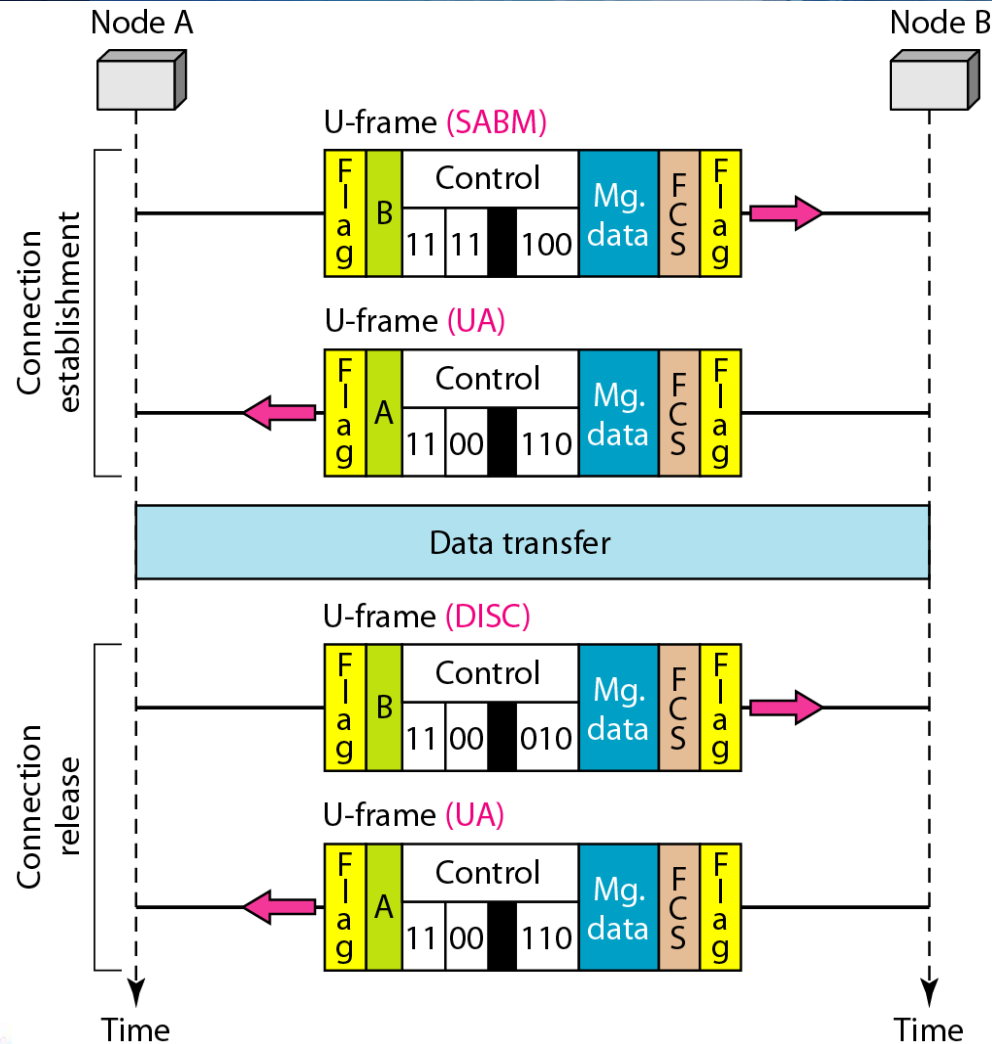


- Code 00 = Receive Ready (RR)
 - Acknowledge frames & waiting for more
- Code 10 = Receive Not Ready (RNR)
 - Acknowledge frames & busy right now
- Code 01 = Reject (REJ)
 - Go-Back-N NAK
- Code 11 = Selective Reject (SREJ)
 - Selective Repeat NAK

Flow&Error
Control

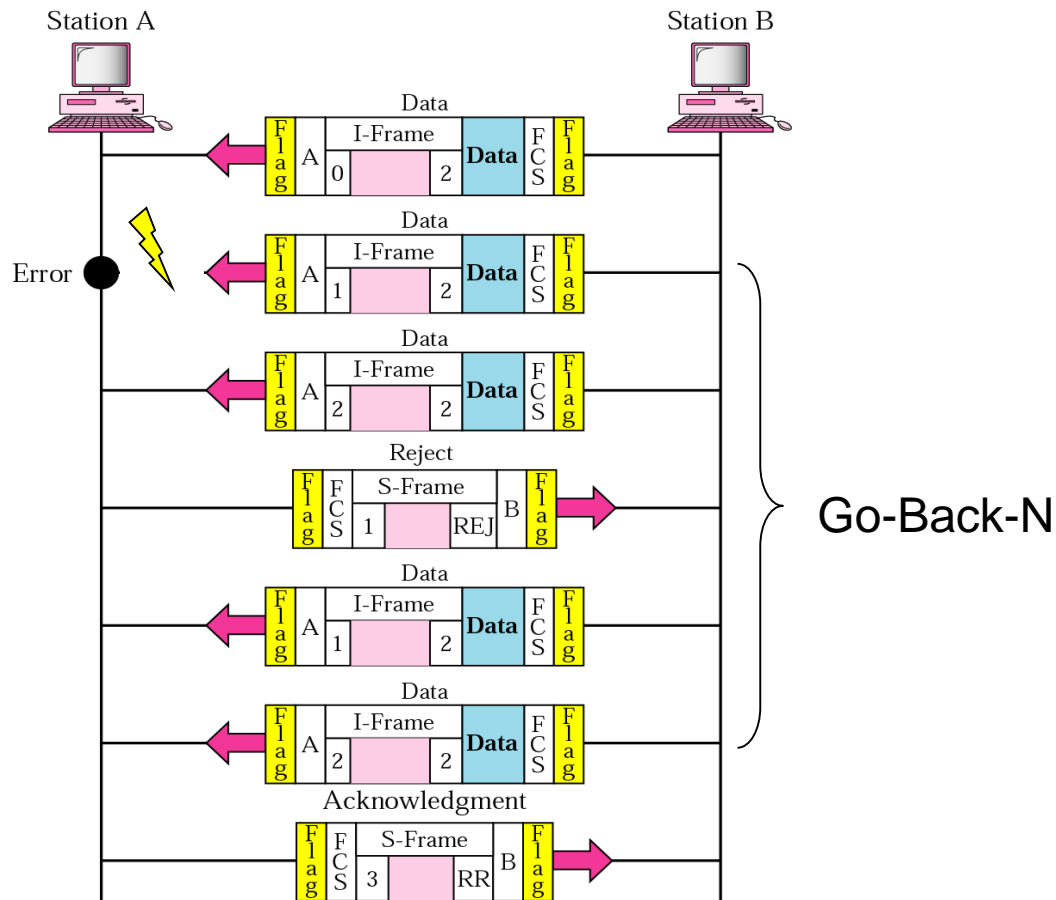
* Figure is courtesy of B. Forouzan

Connection & Disconnection



* Figure is courtesy of B. Forouzan

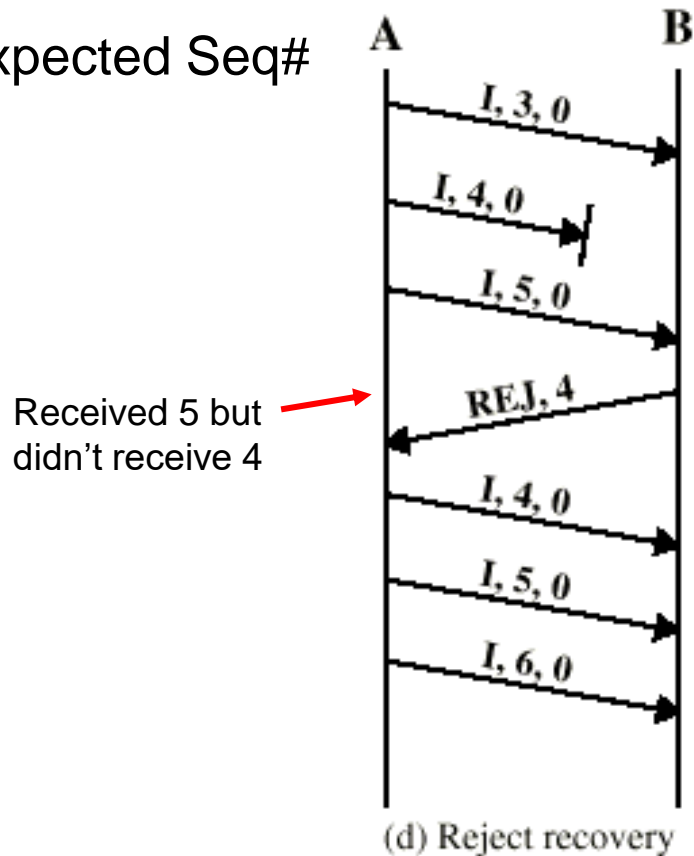
Piggybacking with Error



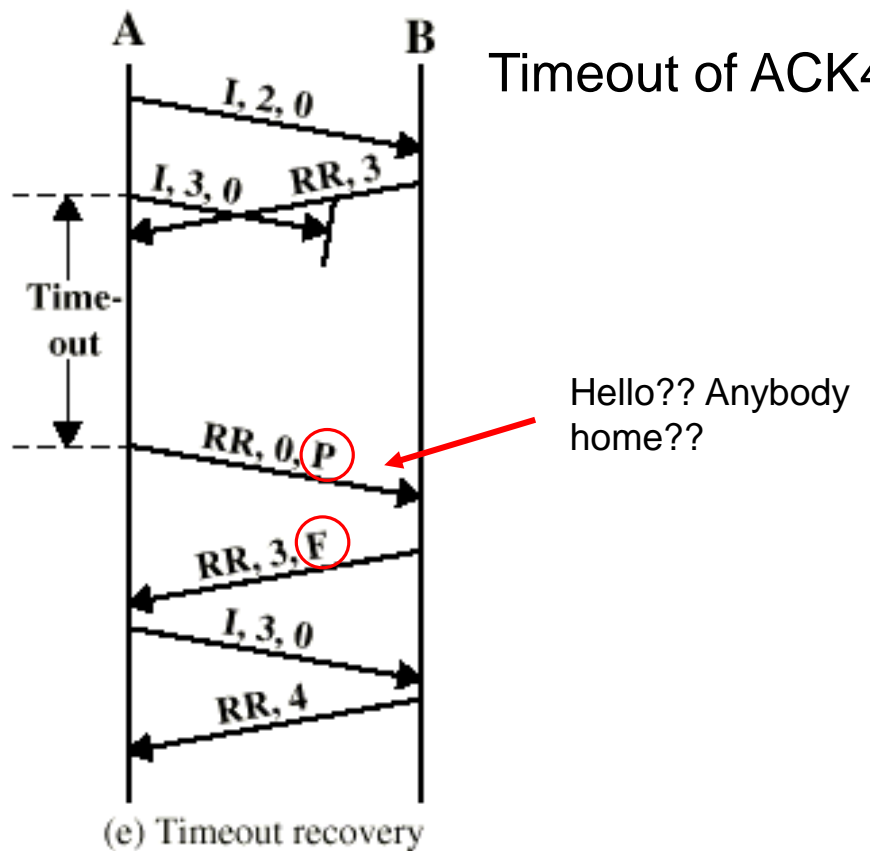
* Figure is courtesy of B. Forouzan

Examples of Operation

Unexpected Seq#



Timeout of ACK4

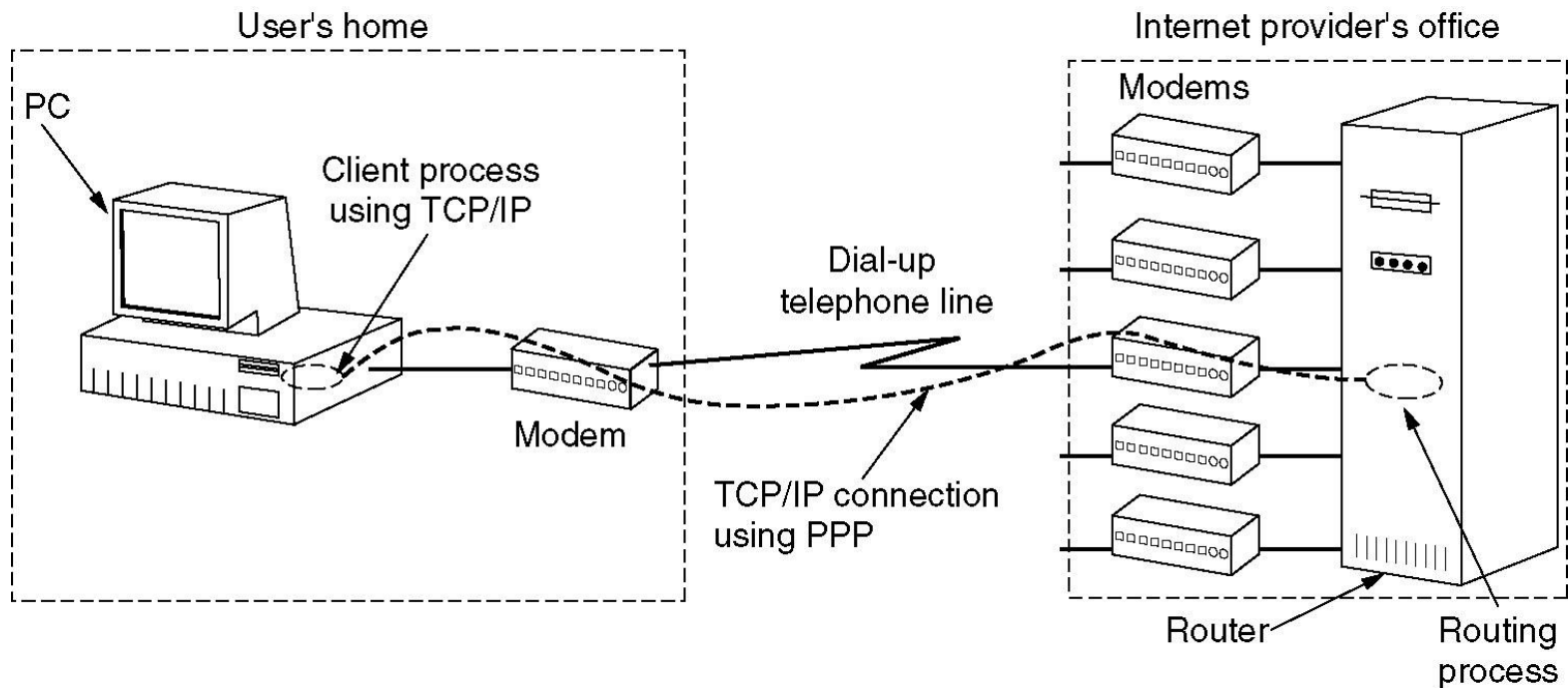


Point-to-Point Protocol (PPP)

Point-to-Point Protocol (PPP)

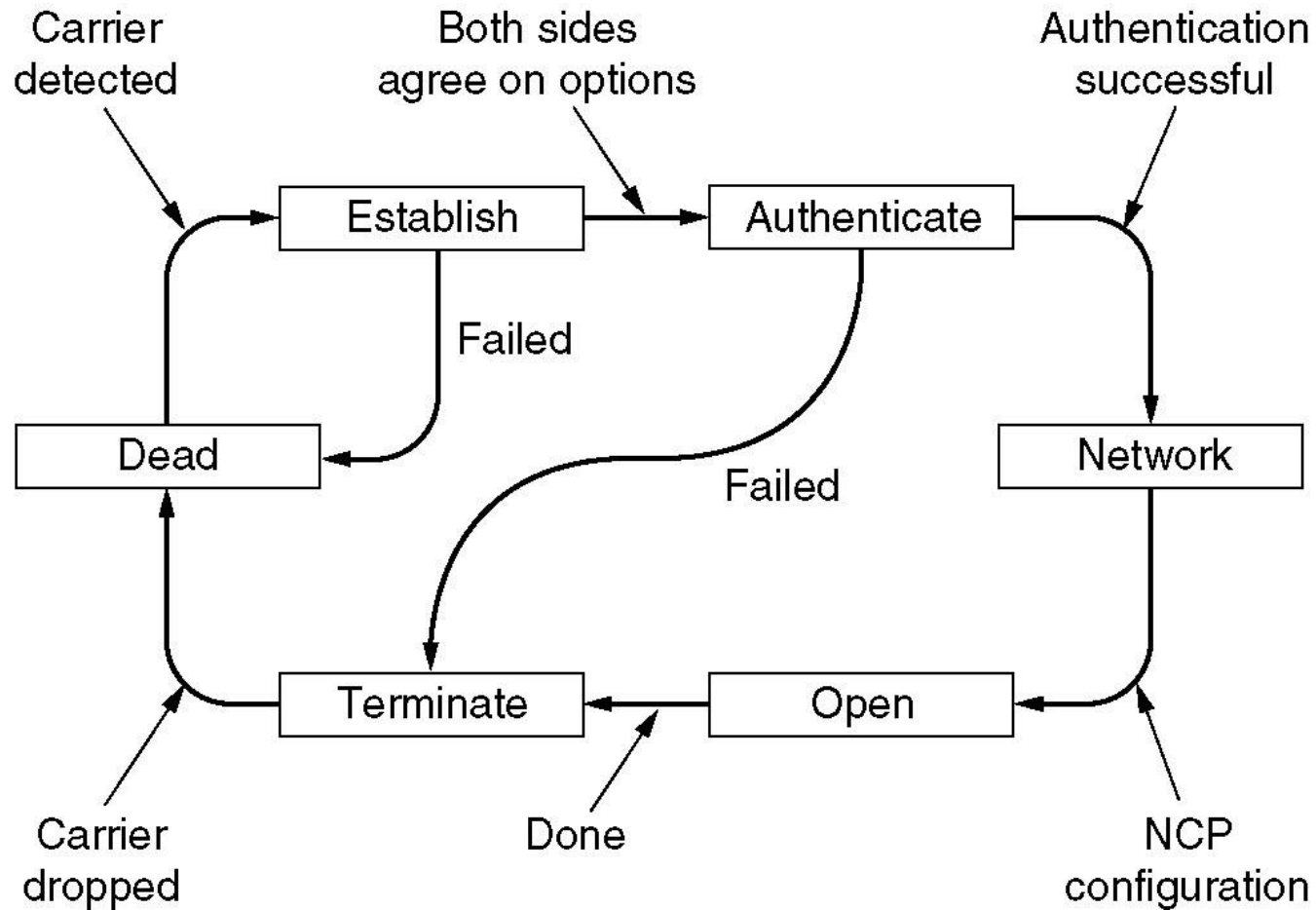
- Used for any kind of serial point to point connection e.g. dial-up, serial x-wire
- Based on HDLC
- Provides
 - Format negotiation
 - Authentication
 - Connection establishment/termination

PPP between Home & ISP



* Figure is courtesy of A. Tanenbaum

PPP – Life Cycle

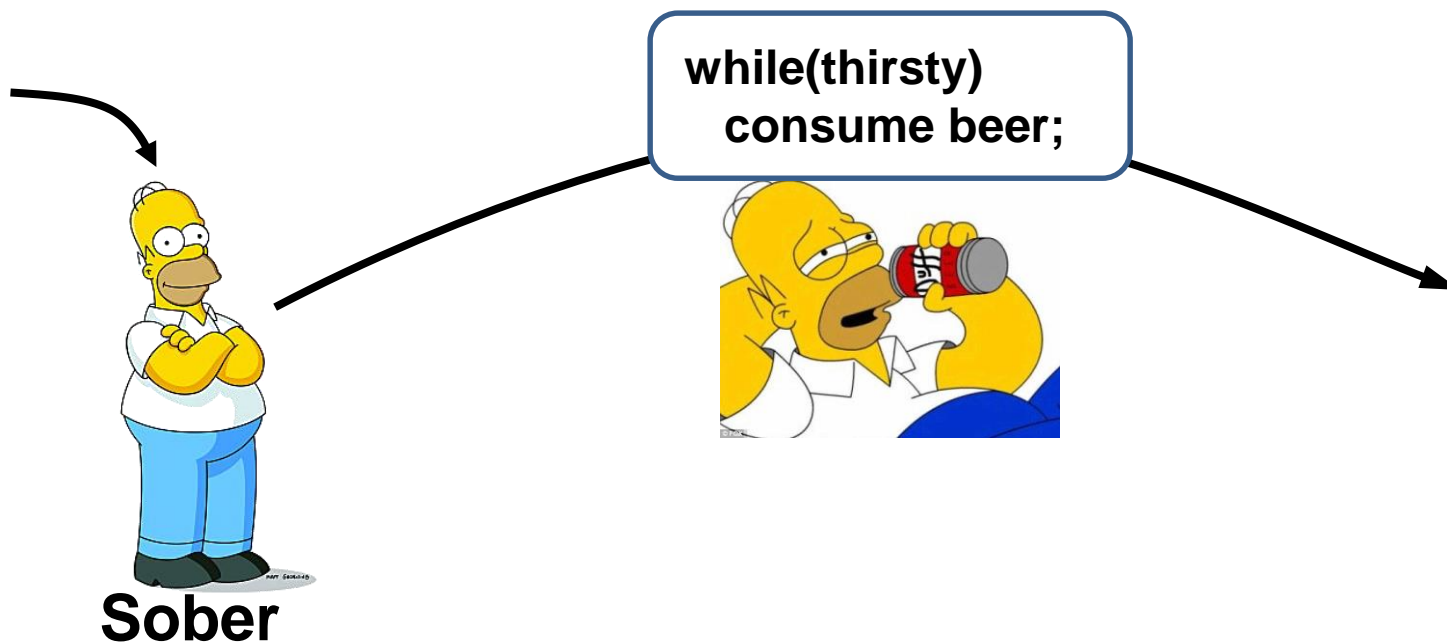


* Figure is courtesy of A. Tanenbaum

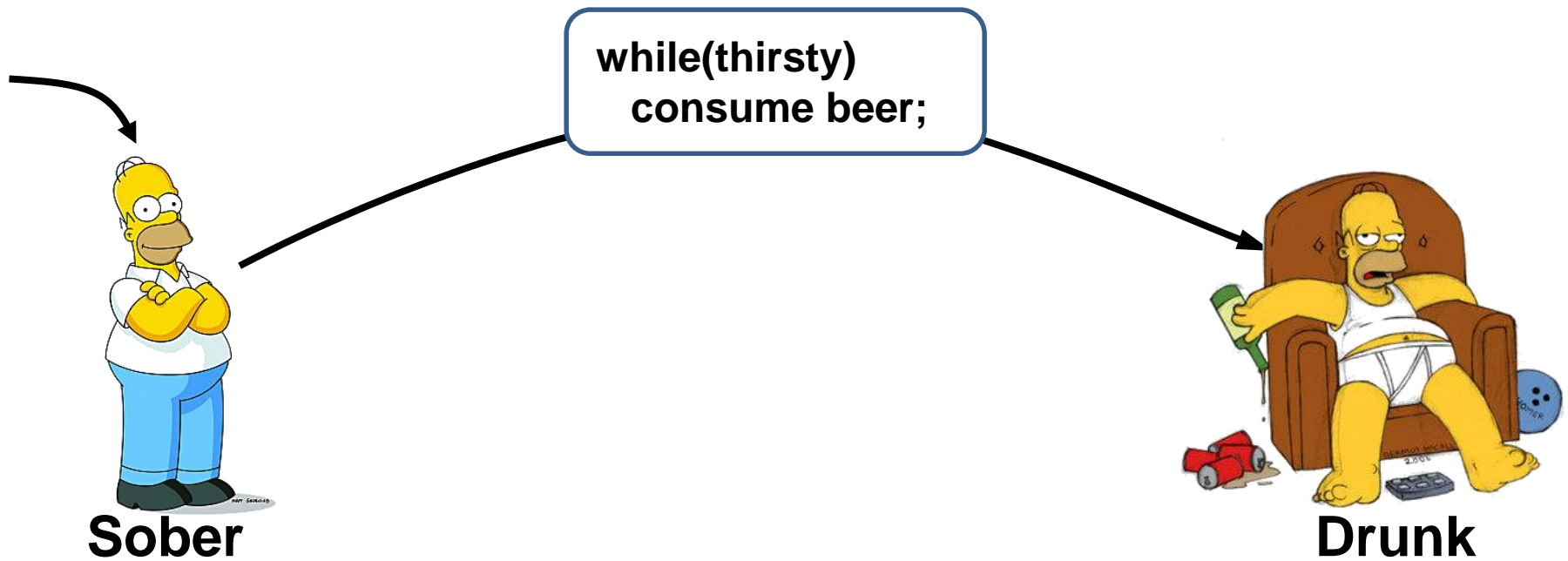
Simpson State Diagram



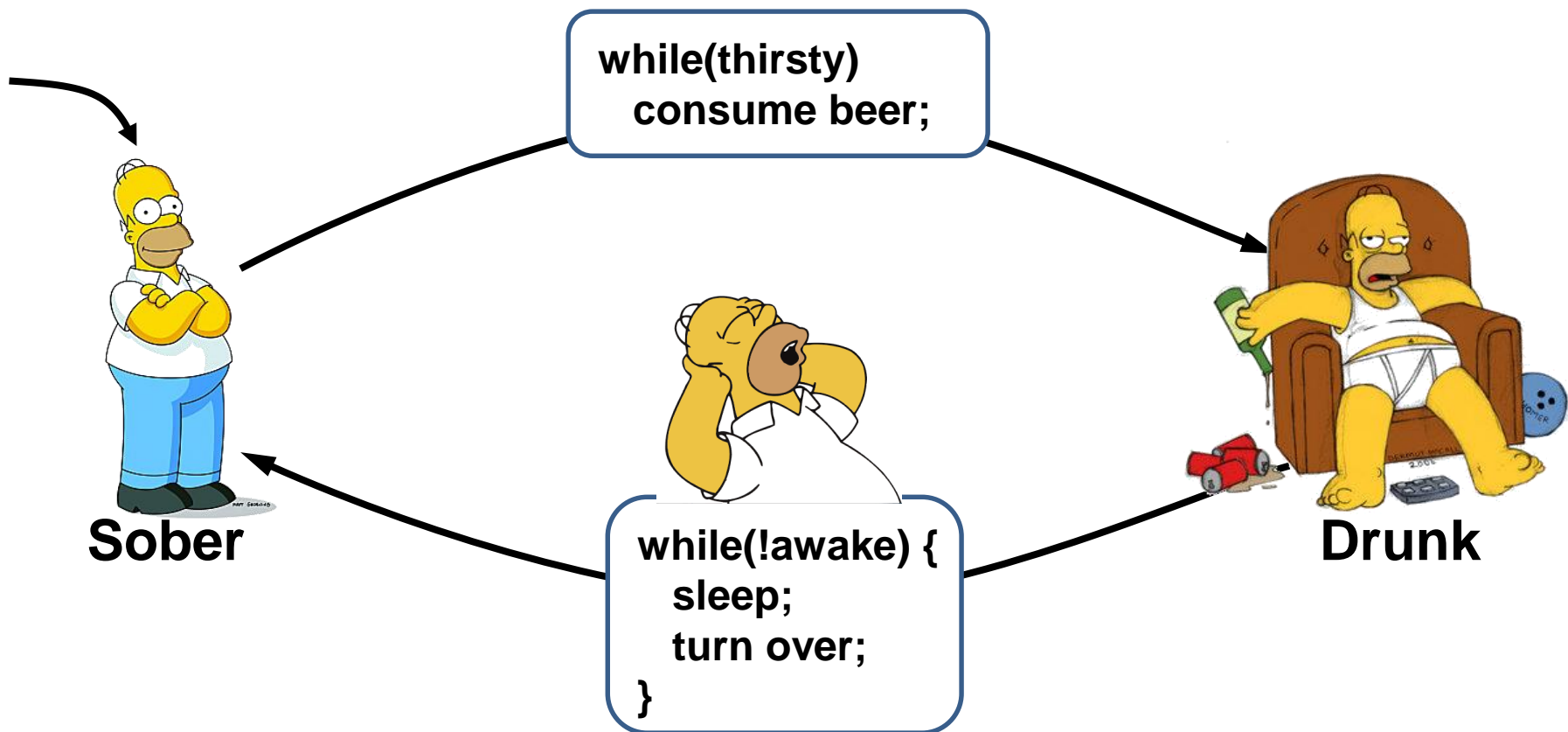
Simpson State Diagram



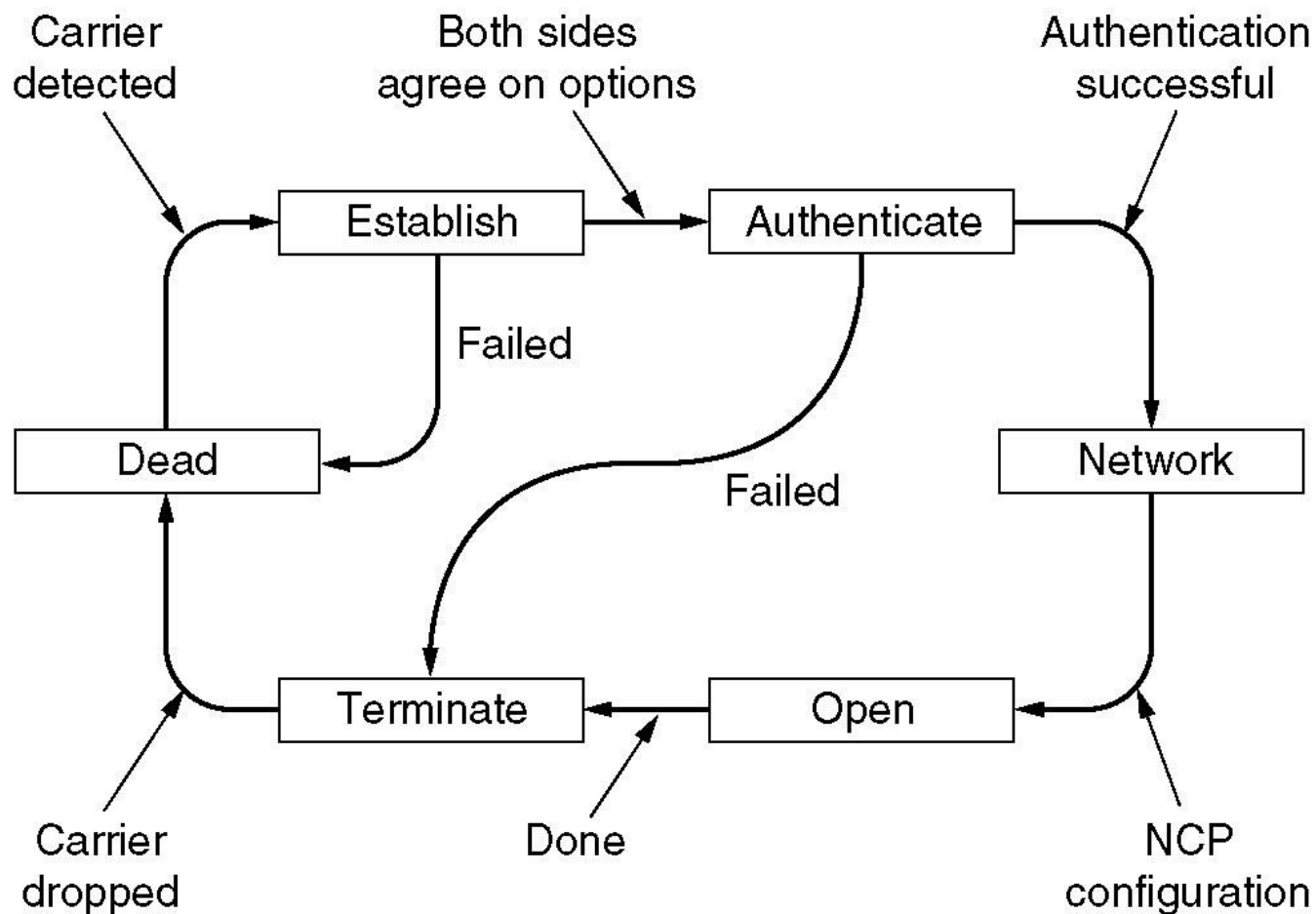
Simpson State Diagram



Simpson State Diagram

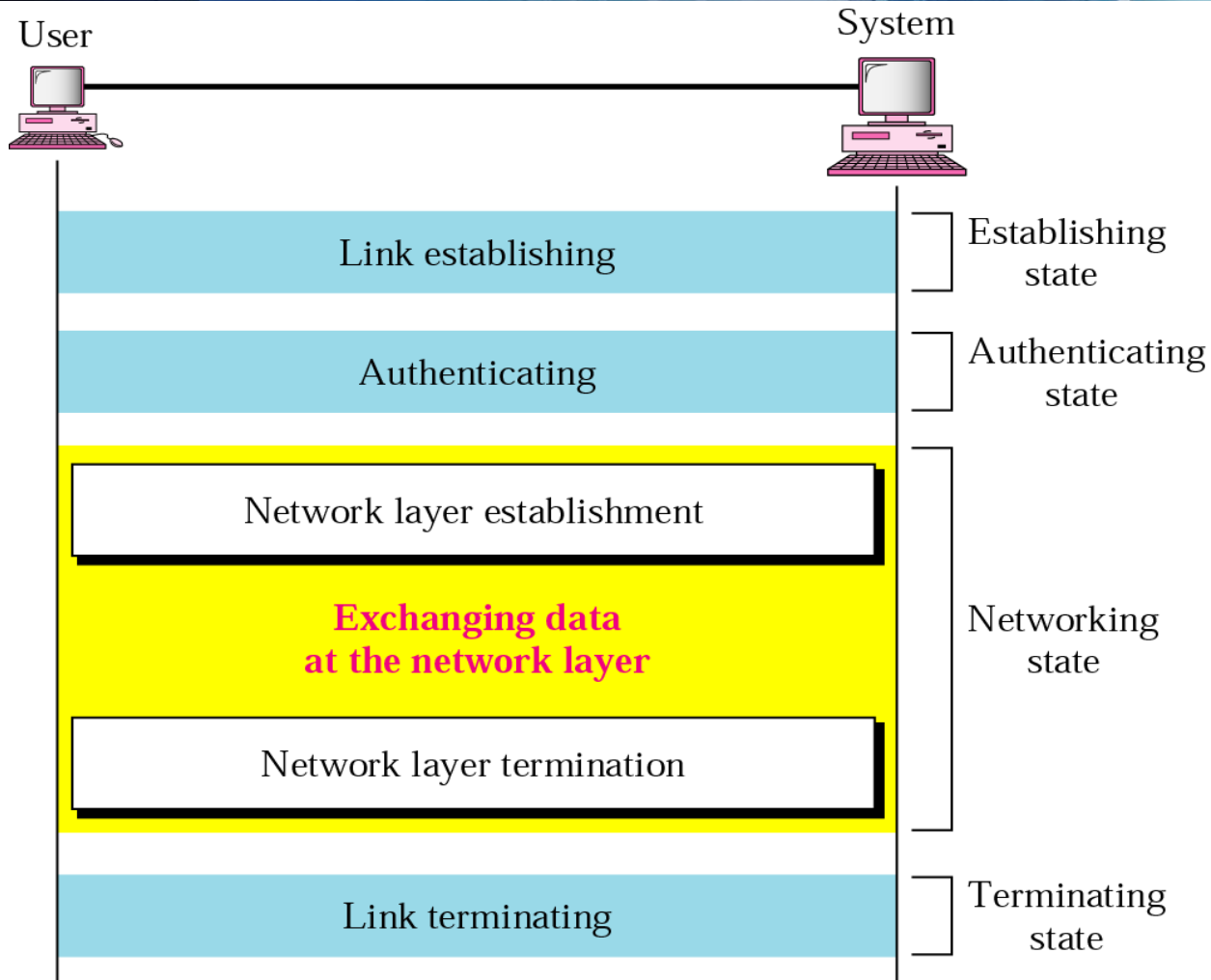


PPP – Life Cycle



* Figure is courtesy of A. Tanenbaum

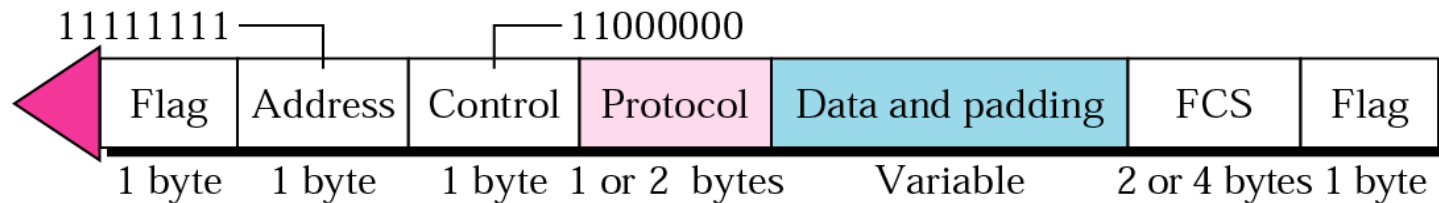
Tasks of PPP



* Figure is courtesy of B. Forouzan

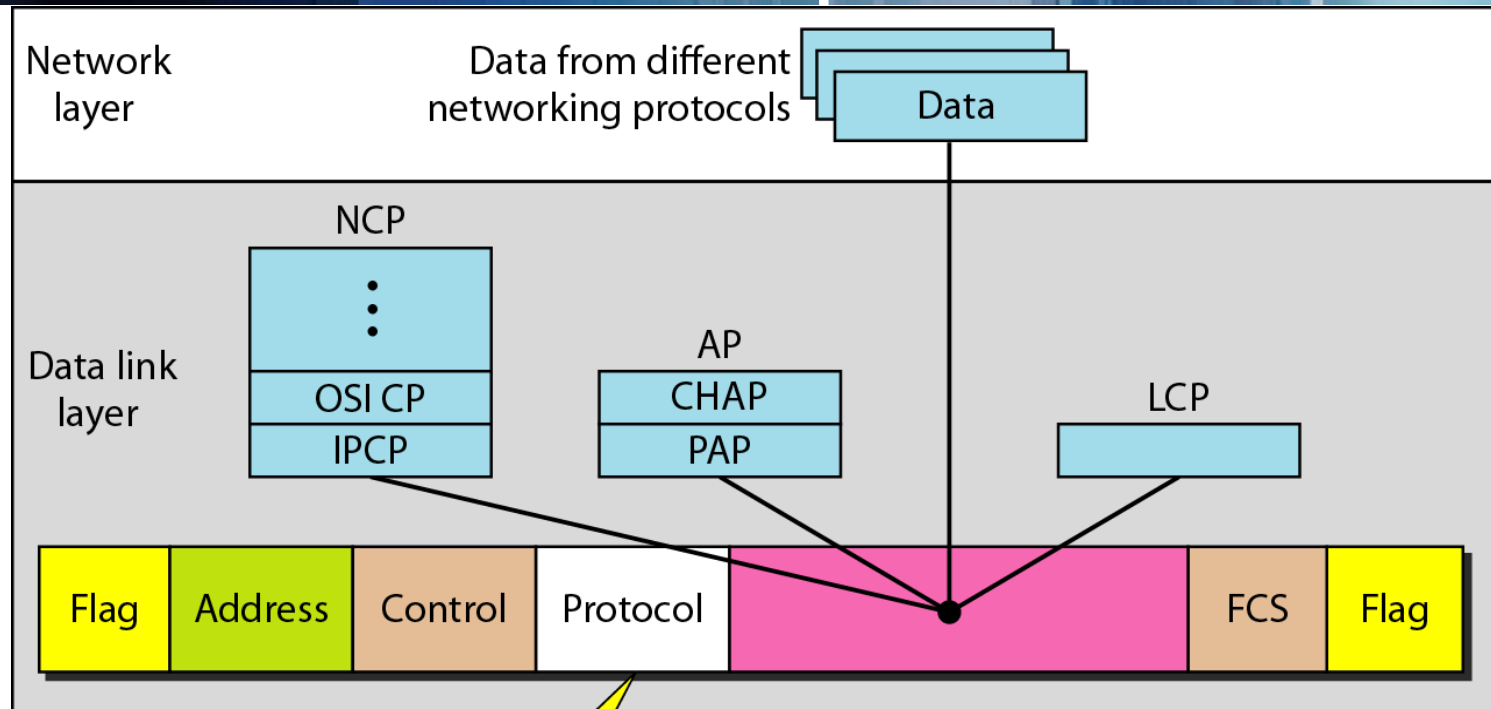
PPP Frame

■ Modified HDLC frame:



- Byte-oriented Protocol
 - Flag Byte: 01111110
 - Escape Byte: 01111101
- FCS: 16- or 32-bit CRC
 - $x^{16} + x^{12} + x^5 + 1$
 - 1 0001 0000 0010 0001 → 16 bits remainder ← 16-degree polynomial
 - $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$

PPP Components

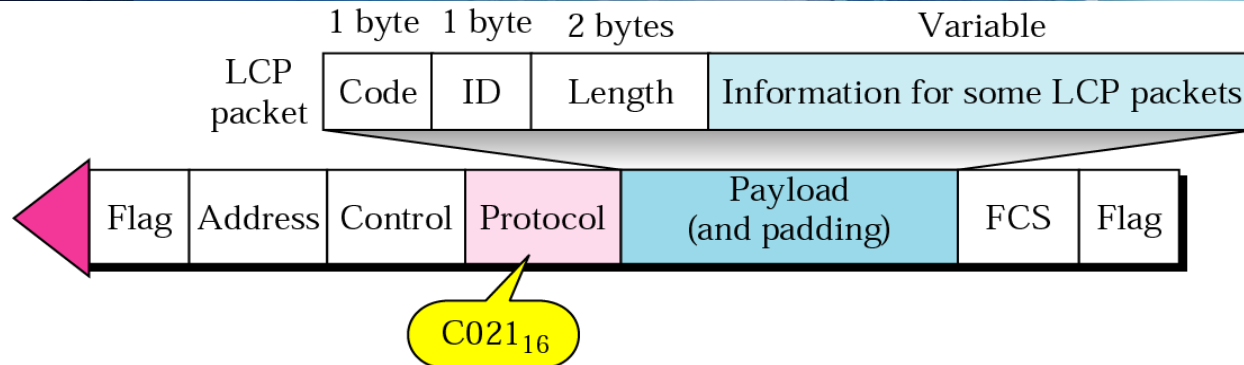


LCP: 0xC021
AP: 0xC023 and 0xC223
NCP: 0x8021 and
Data: 0x0021 and

LCP: Link Control Protocol
AP: Authentication Protocol
NCP: Network Control Protocol

* Figure is courtesy of B. Forouzan

LCP Packet



Code	Packet Type	Description
0x01	Configure-request	Contains the list of proposed options and their values
0x02	Configure-ack	Accepts all options proposed
0x03	Configure-nak	Announces that some options are not acceptable
0x04	Configure-reject	Announces that some options are not recognized
0x05	Terminate-request	Requests to shut down the line
0x06	Terminate-ack	Accepts the shut down request

* Figure is courtesy of B. Forouzan

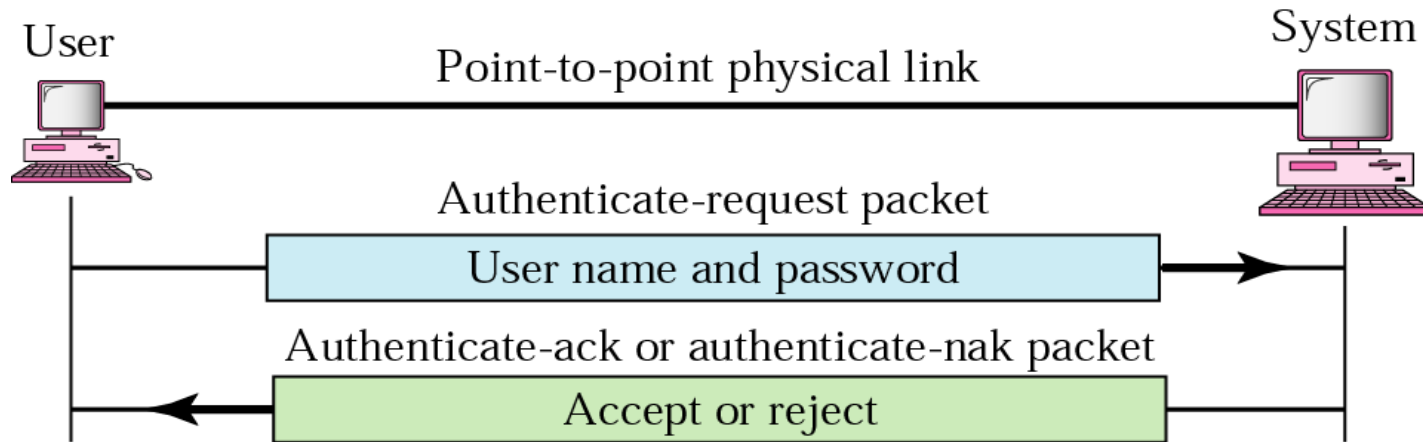
Common Options

Option	Default
Maximum receive unit	1500
Authentication protocol	None
Protocol field compression	Off
Address and control field compression	Off

LCP Debug Codes

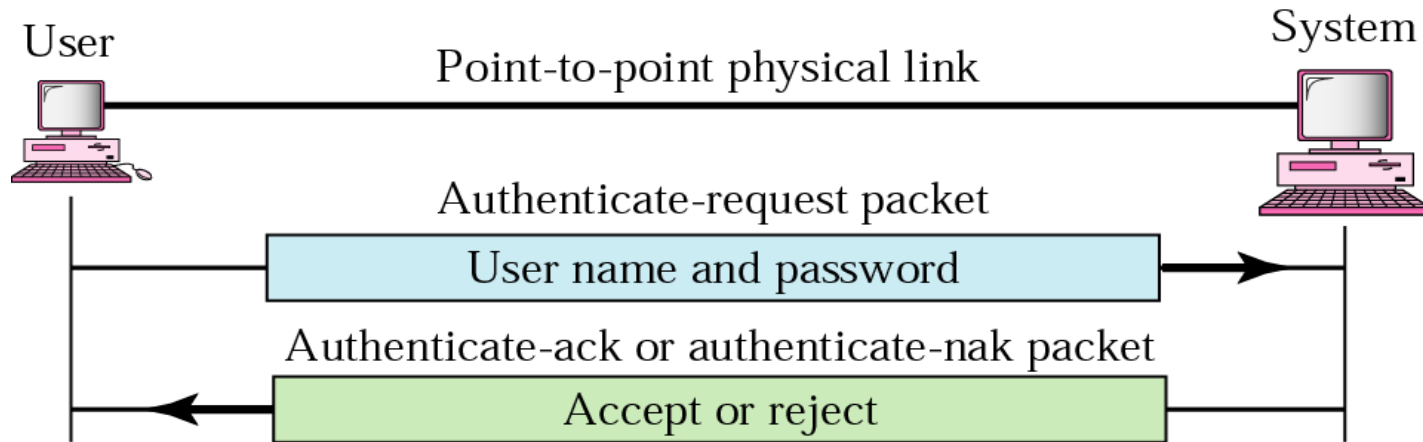
Code	Packet Type	Description
0x07	Code-reject	Announces an unknown code
0x08	Protocol-reject	Announces an unknown protocol
0x09	Echo-request	A type of hello message to check if the other end is alive
0x0A	Echo-reply	The response to the echo-request message
0x0B	Discard-request	A request to discard the packet

Password Authentication Protocol (PAP)



- 2-Way Handshake
- Password transmitted in clear text

Password Authentication Protocol (PAP)



- 2-Way Handshake
- Password transmitted in clear text

...what were they thinking???

Never transfer passwords in clear text

Communication Scenario



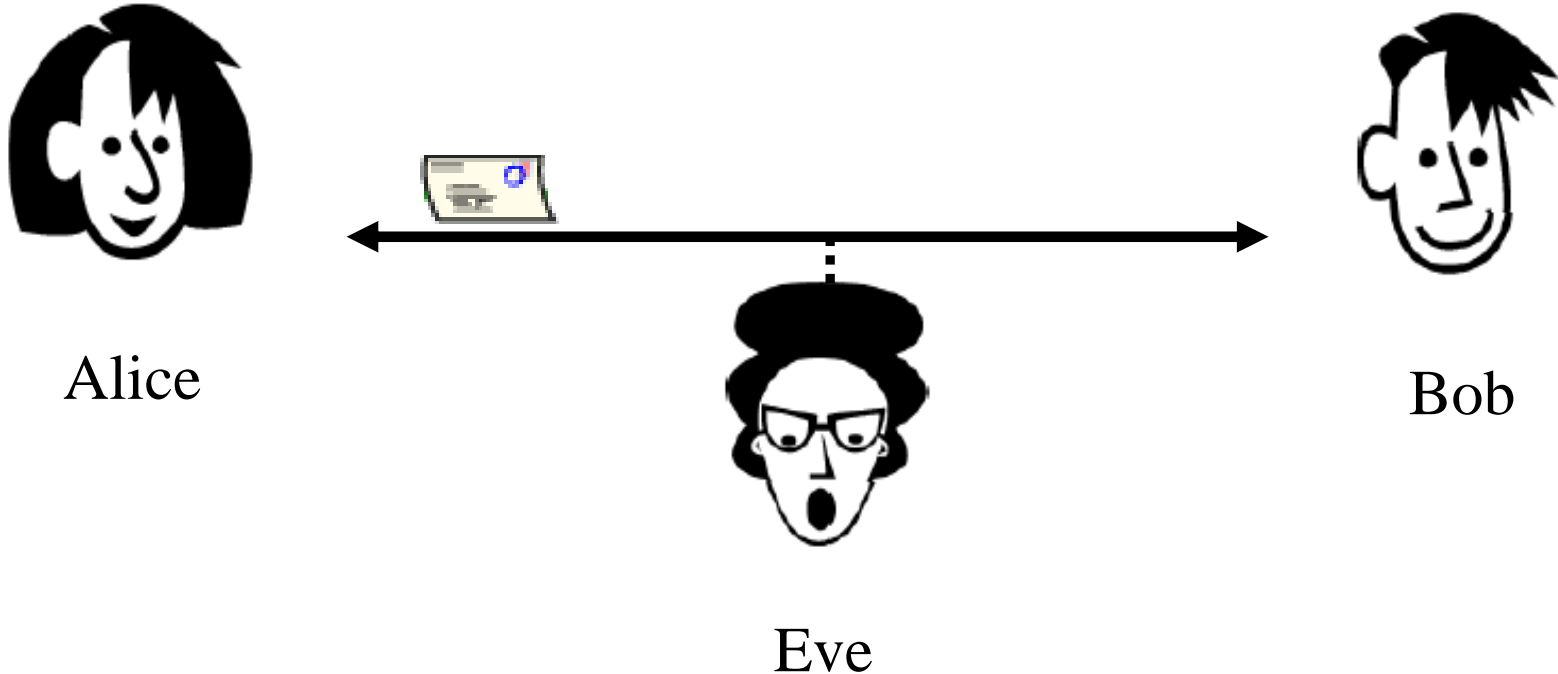
Alice



Bob

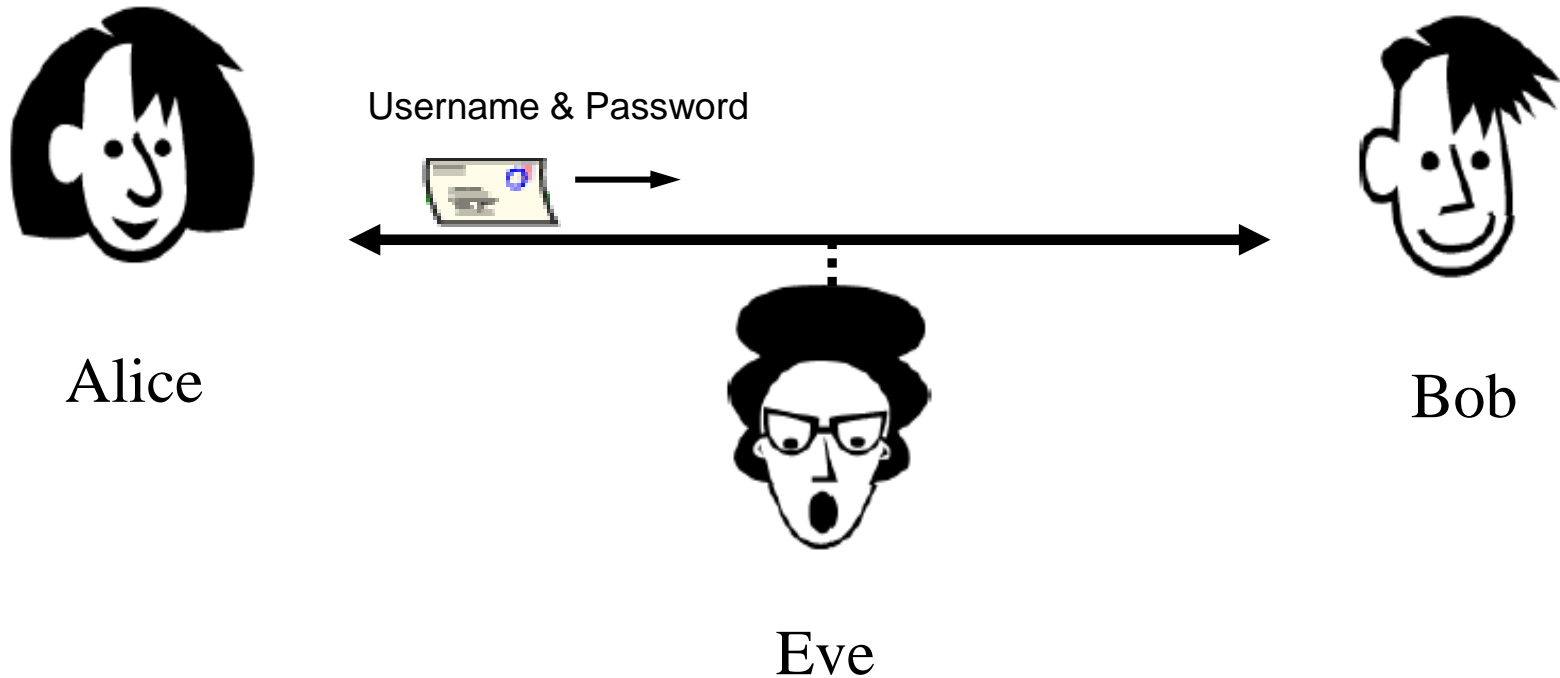
- Alice and Bob want to communicate

Threat: Eavesdropping



- Alice and Bob want to communicate
- Eve is eavesdropping (intercept, delete, add messages)

PAP Transfer



- Eve captures Alice's username & password

Playback Attack



Alice

Username & Password



Bob

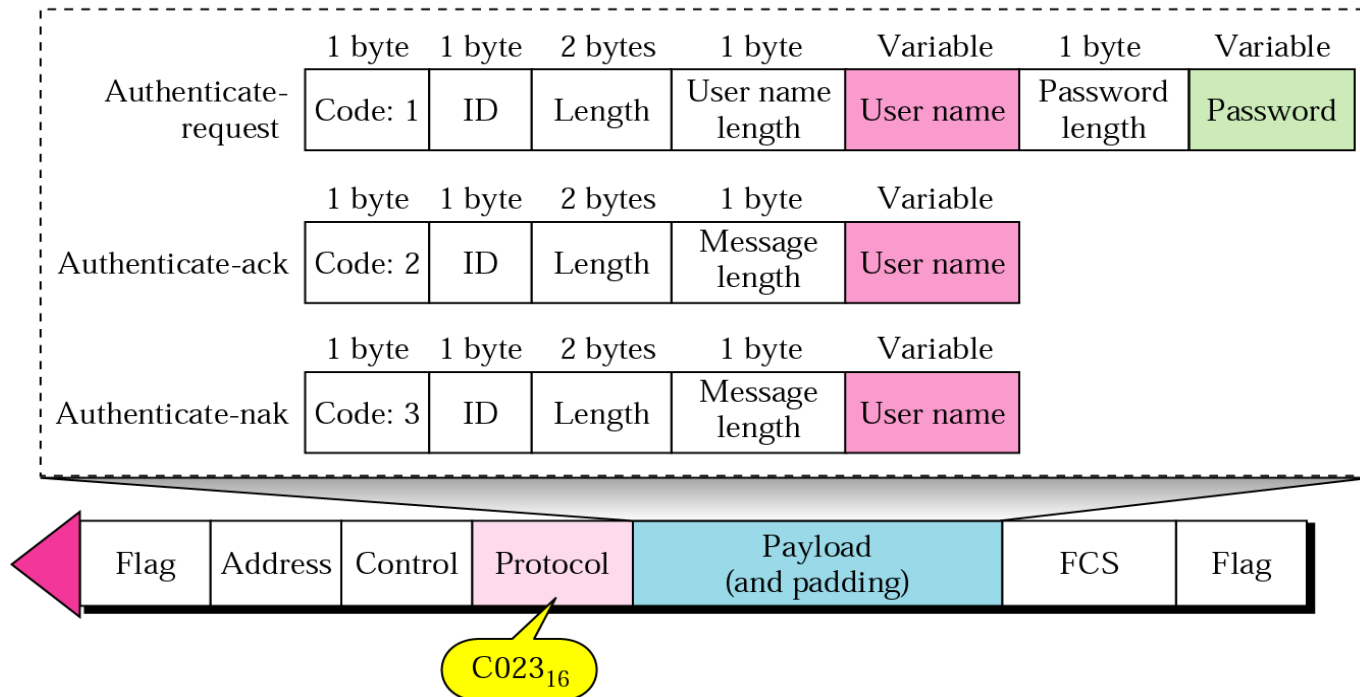


Eve

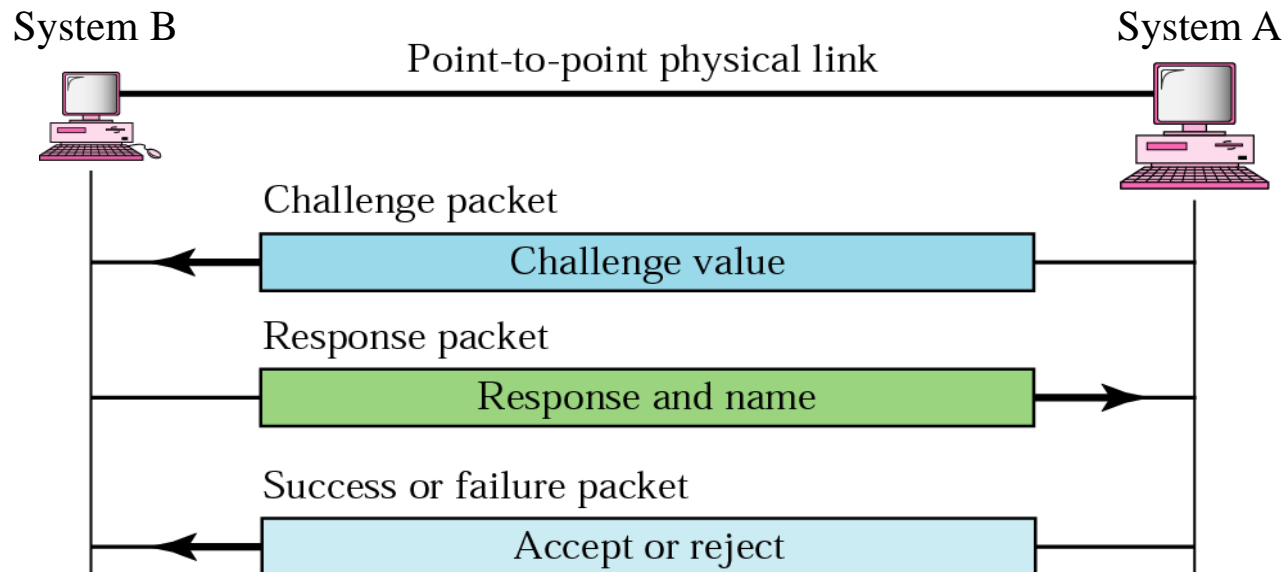
- Eve pretends to be Alice by using sending out the capture message

PAP Packets

PAP packets



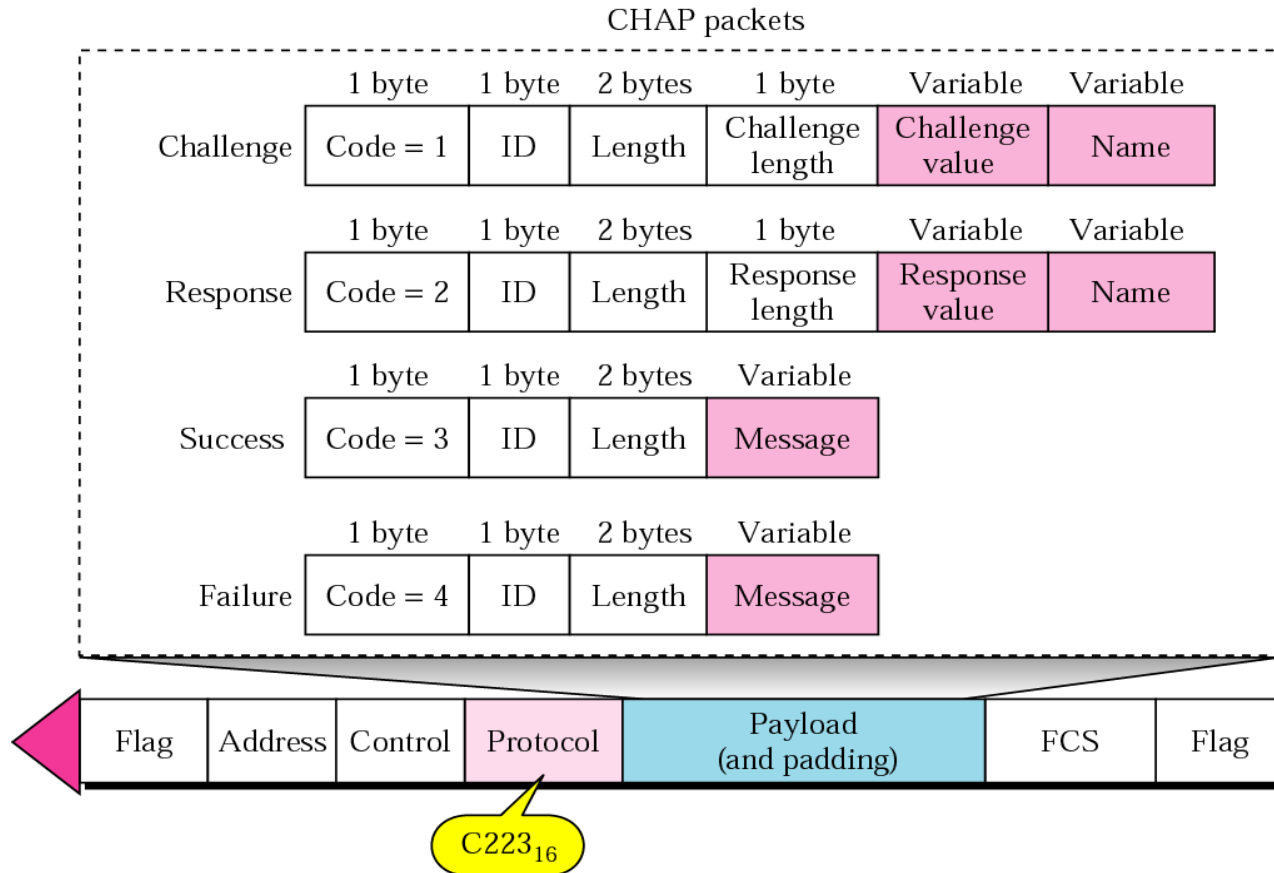
CHAP



- **3-Way Handshake**

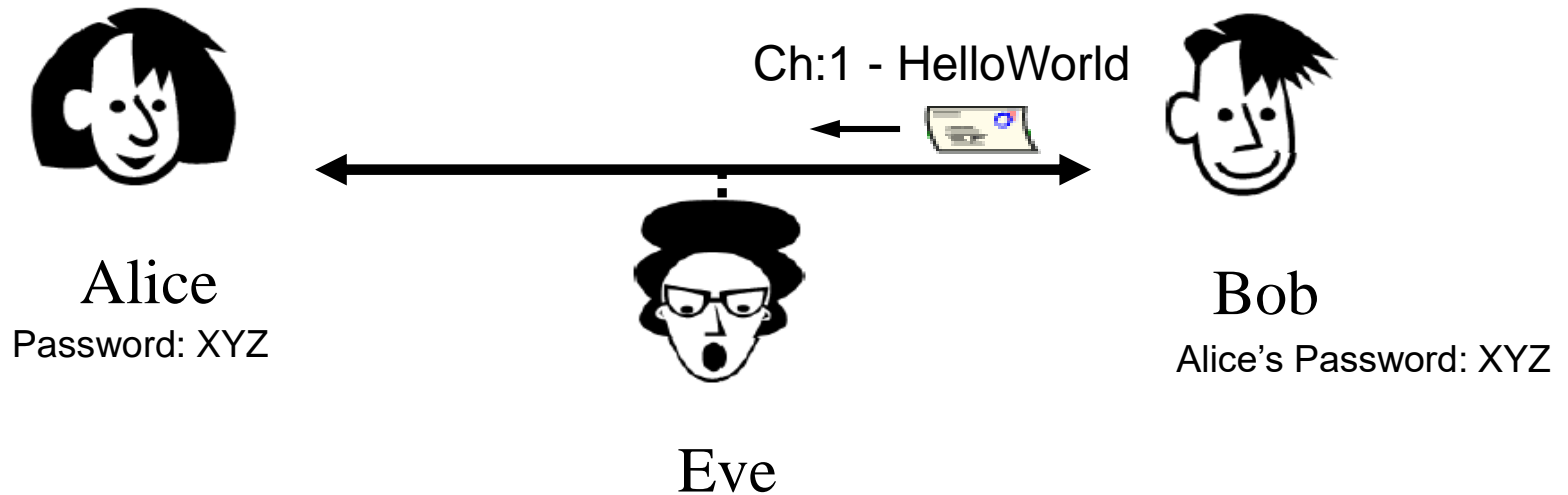
1. A creates challenge \Rightarrow challenge value
2. B processes challenge value with password \Rightarrow response
3. A compares response with own calculation \Rightarrow accepts or rejects response

CHAP Packets

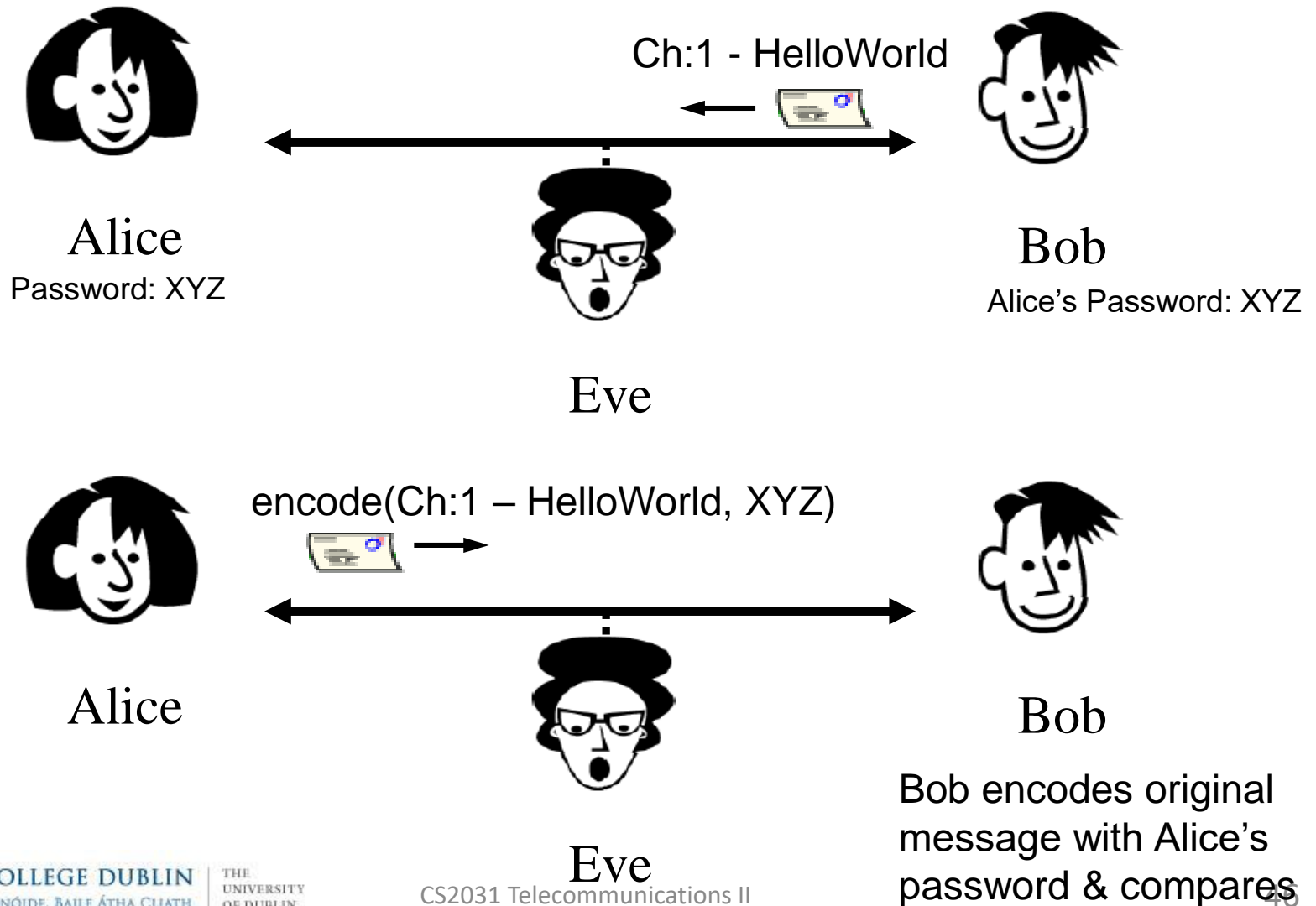


* Figure is courtesy of B. Forouzan

Chap Exchange



Chap Exchange



Chap Replay



Alice

encode(Ch:1 – HelloWorld, XYZ)



Eve

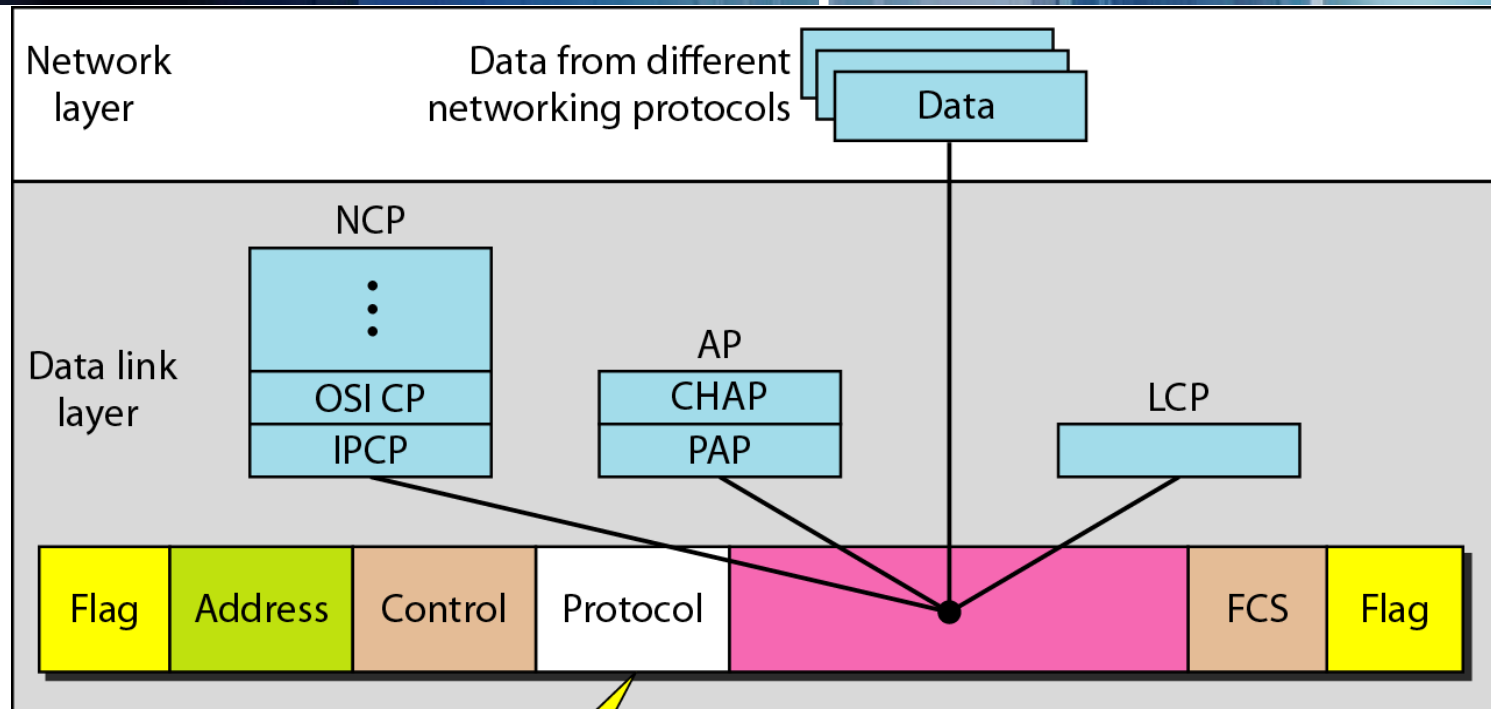


Bob

I've seen this already

- Eve can capture both messages
 - ❑ but will not see the password
 - ❑ or be able to use replay attack

PPP Components



LCP: 0xC021
AP: 0xC023 and 0xC223
NCP: 0x8021 and
Data: 0x0021 and

LCP: Link Control Protocol
AP: Authentication Protocol
NCP: Network Control Protocol

PPP – Encapsulation

flag	addr	ctrl	protocol	data	CRC	flag
7E	FF	03				7E
1	1	1	2	≤ 1500	2	1

0021

IP datagram

C021

link control data

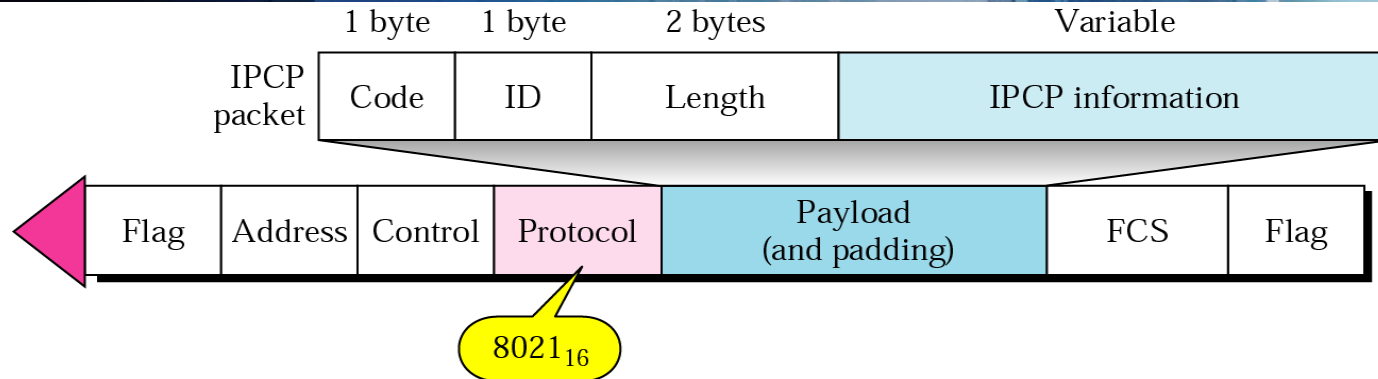
8021

network control data

Network Control Protocol (NCP)

- PPP negotiates Network Layer information
- Consists of specific protocols for network layer protocols
 - IP Control Protocol (IPCP)
 - IPX Control Protocol (IPXCP)
- May include IP Header compression

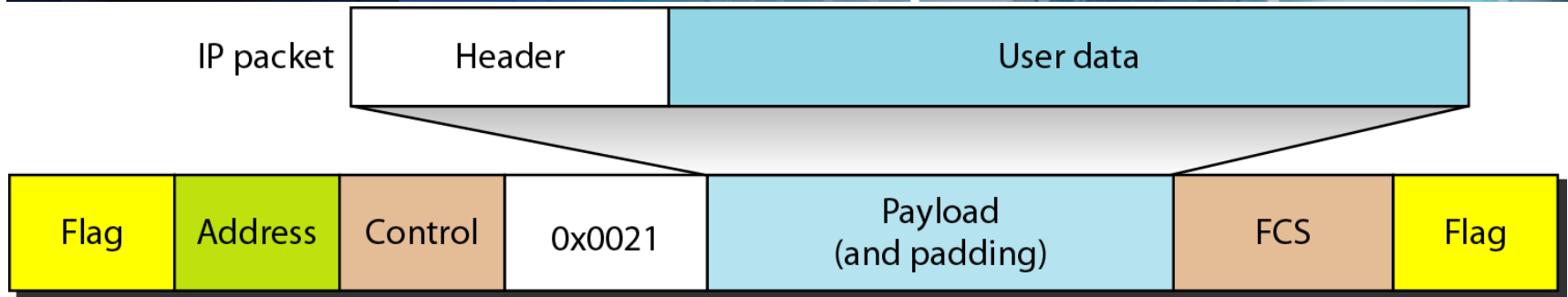
IPCP Packet in PPP Packet



Code	IPCP Packet
01	Configure Request
02	Configure ACK
03	Configure NAK
04	Configure Reject
05	Terminate-request
06	Terminate-ack
07	Code-reject

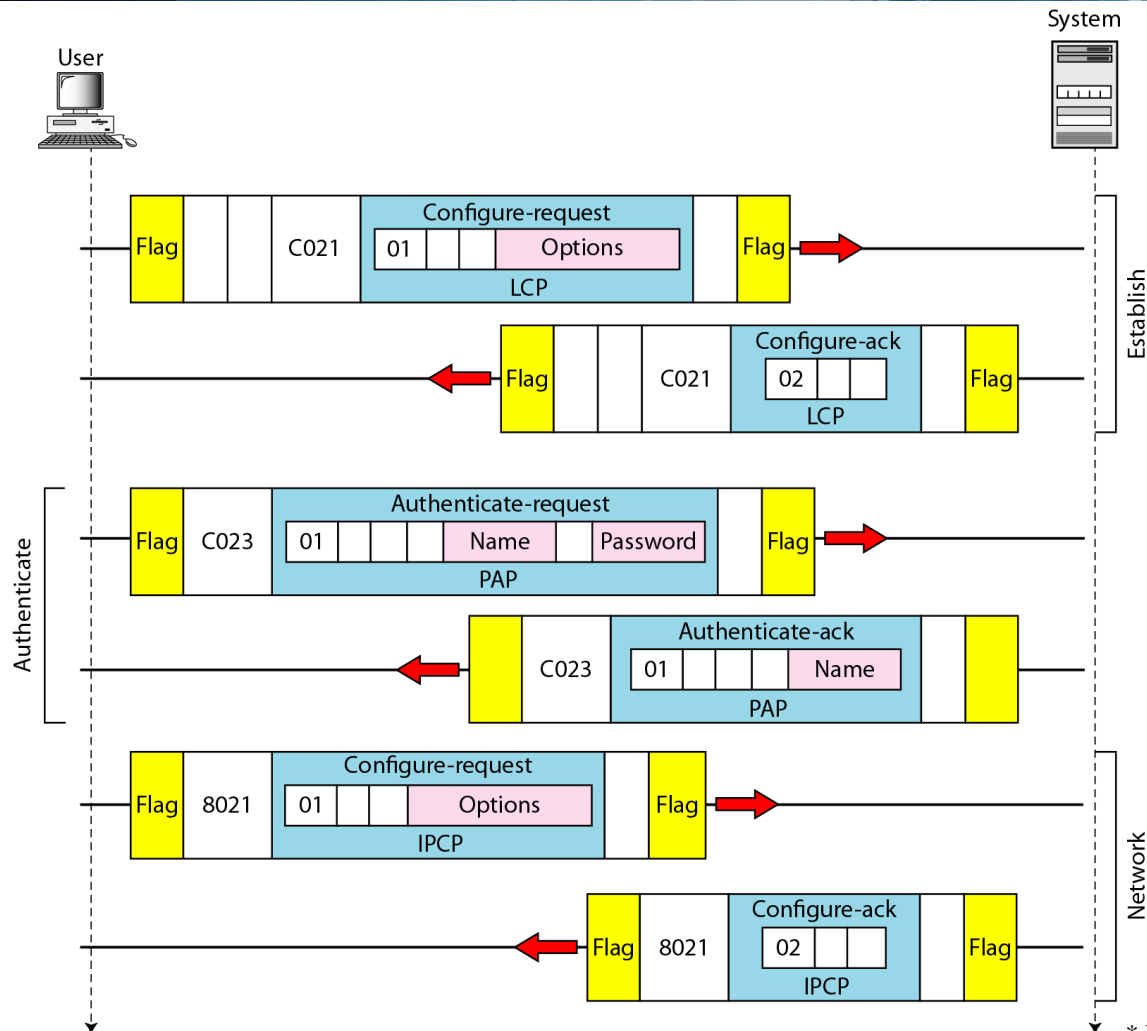
* Figure is courtesy of B. Forouzan

IP Packet Encapsulation



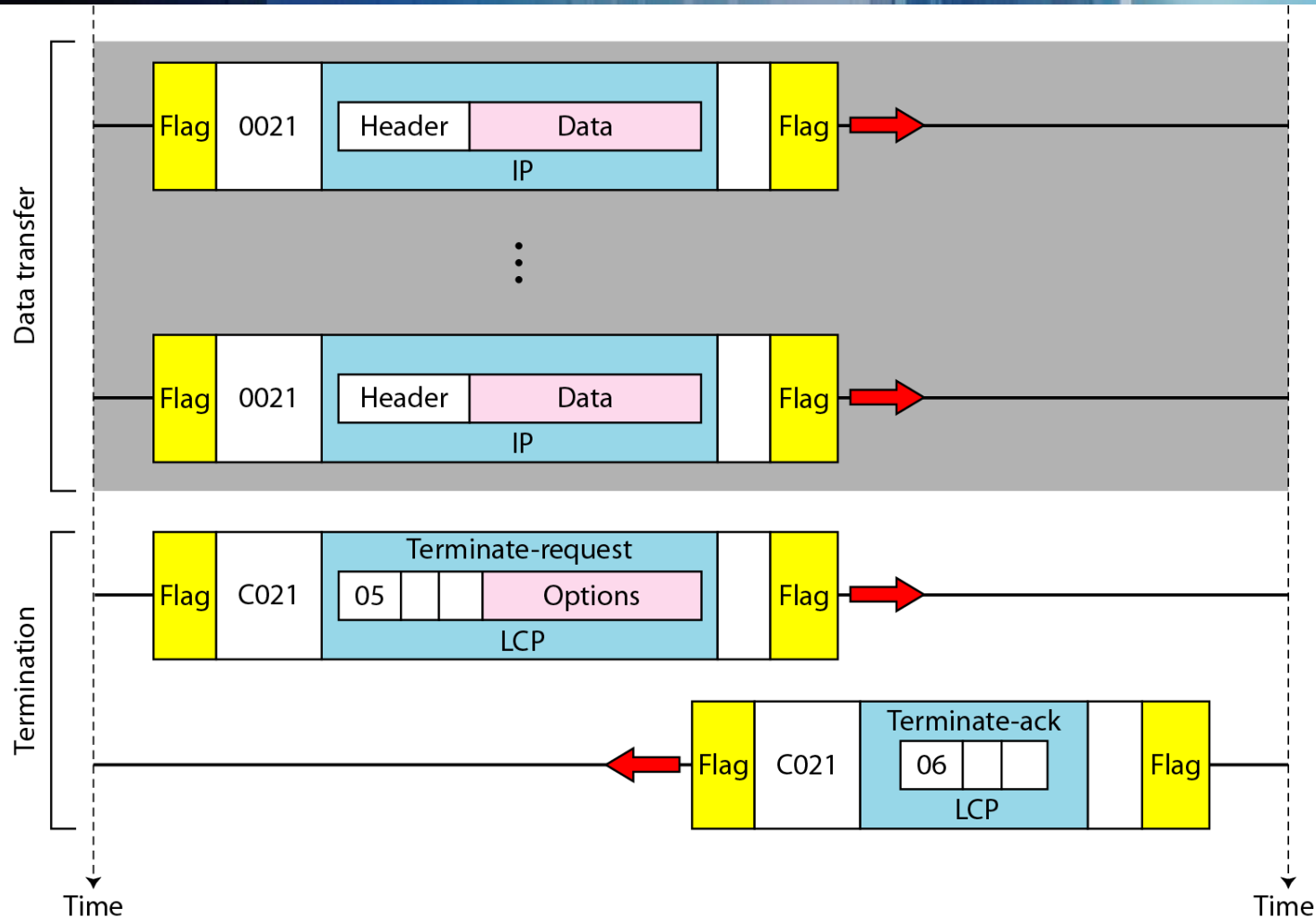
- PPP frame carries IP Packet as payload

Connection Setup & Authentication



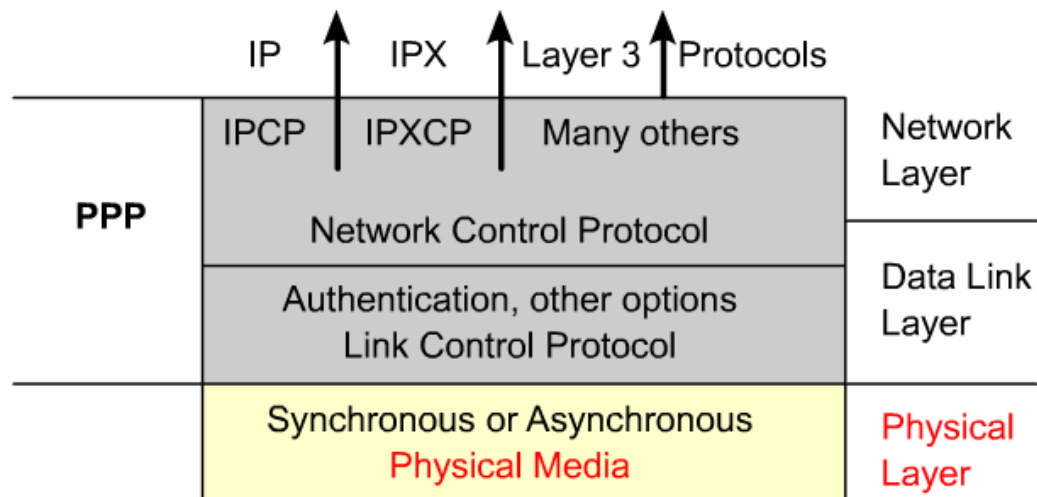
* Figure is courtesy of B. Forouzan

Data Transfer & Termination



* Figure is courtesy of B. Forouzan

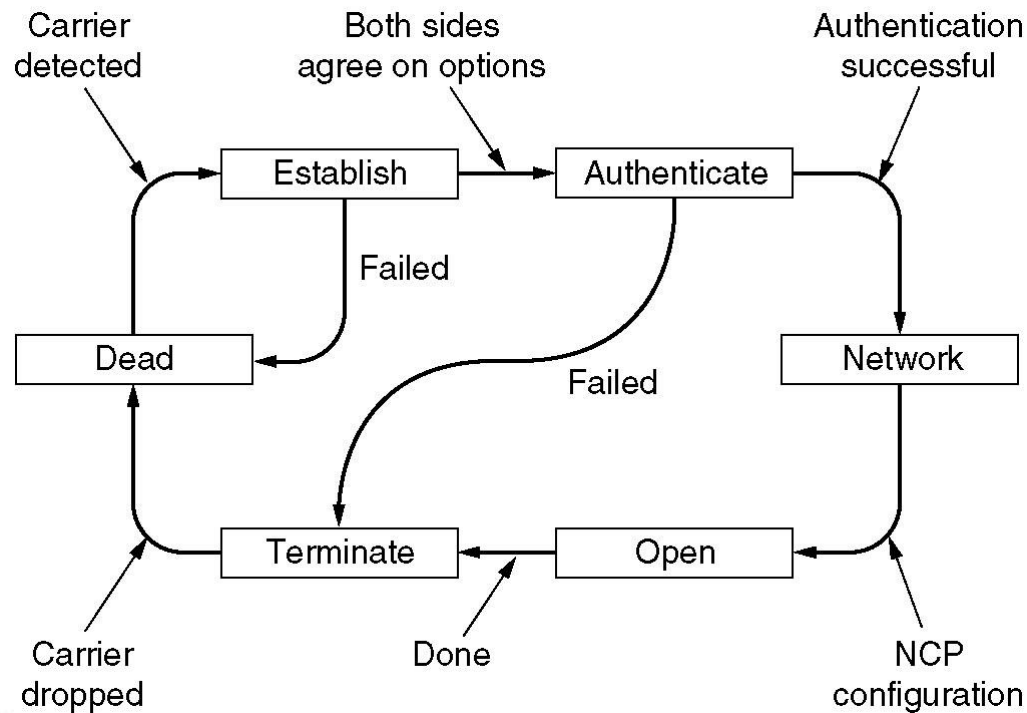
PPP in OSI Stack



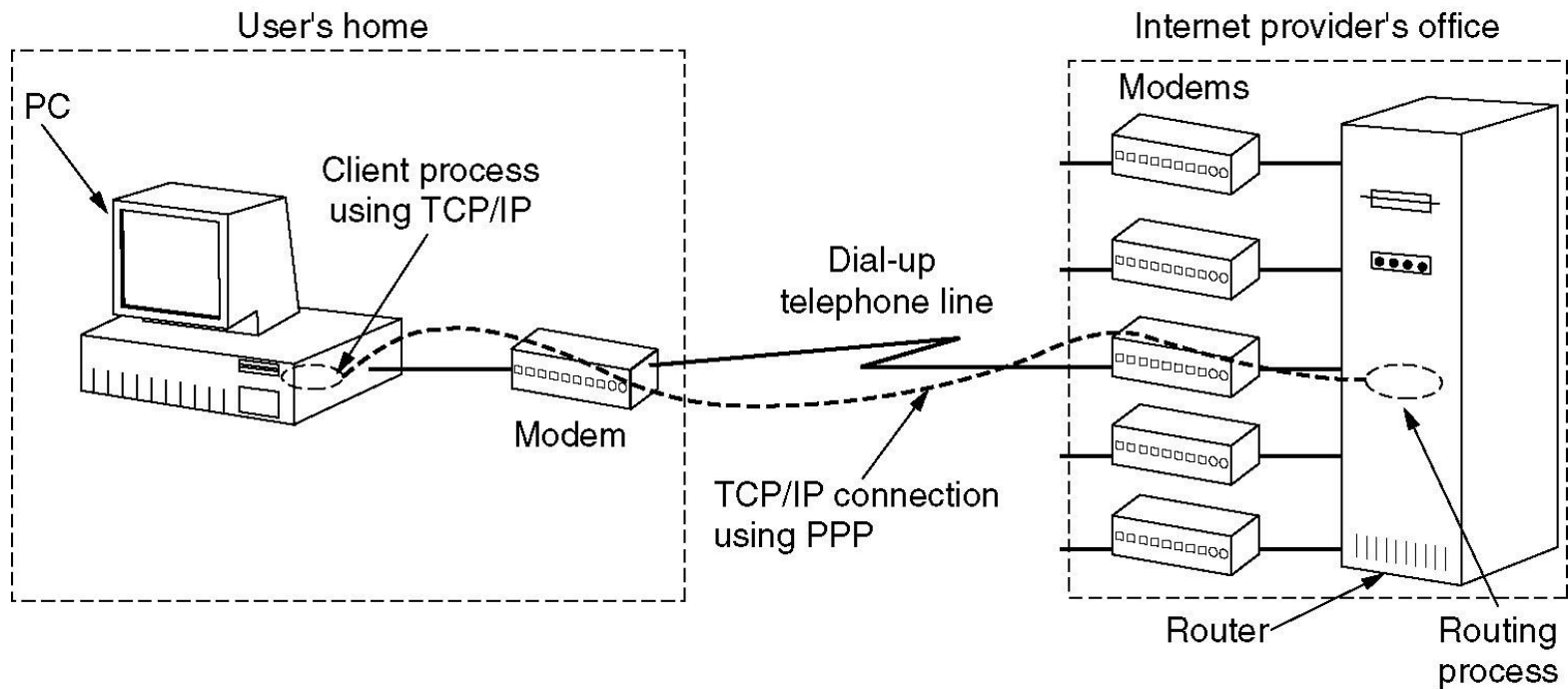
- PPP doesn't implement:
 - error correction/recovery (but error detection)
 - flow control
 - ordered delivery
 - multipoint links
- ⇒ all relegated to higher layers!

PPP

- Connection Establishment & Termination
- Format Negotiation
- Authentication

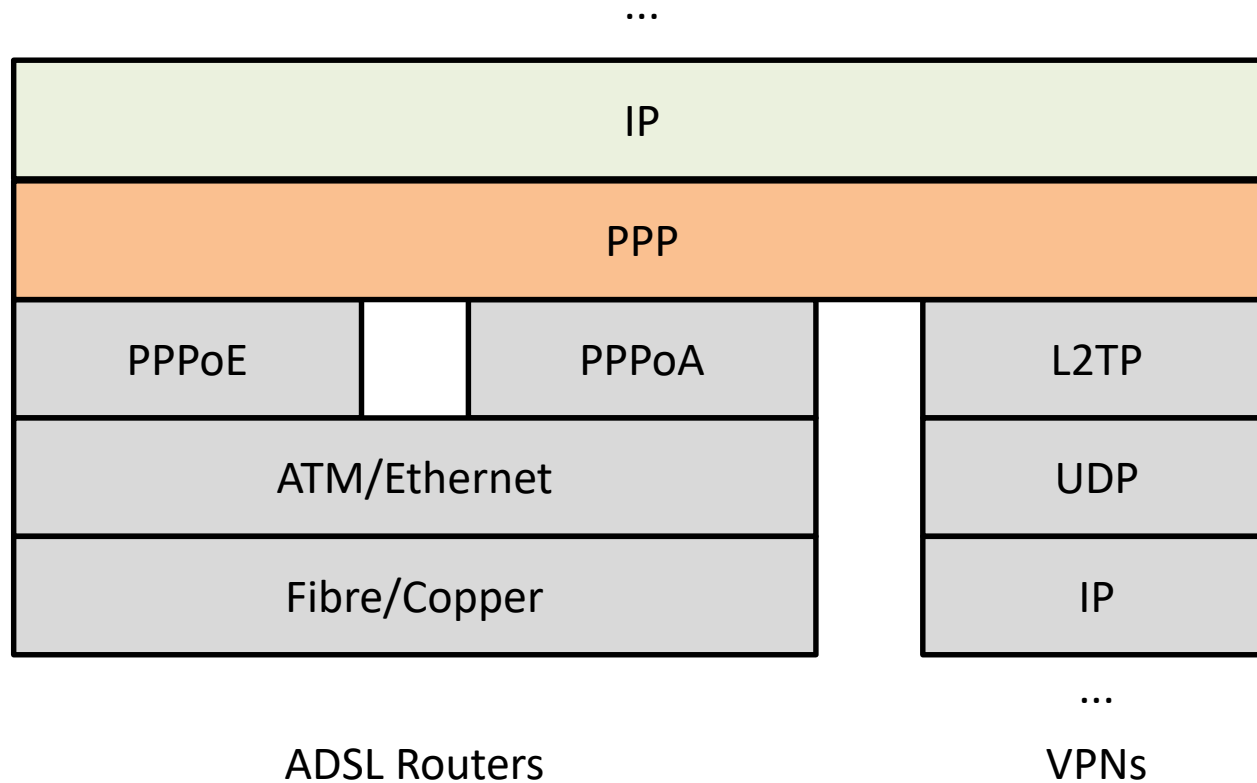


PPP – Why???

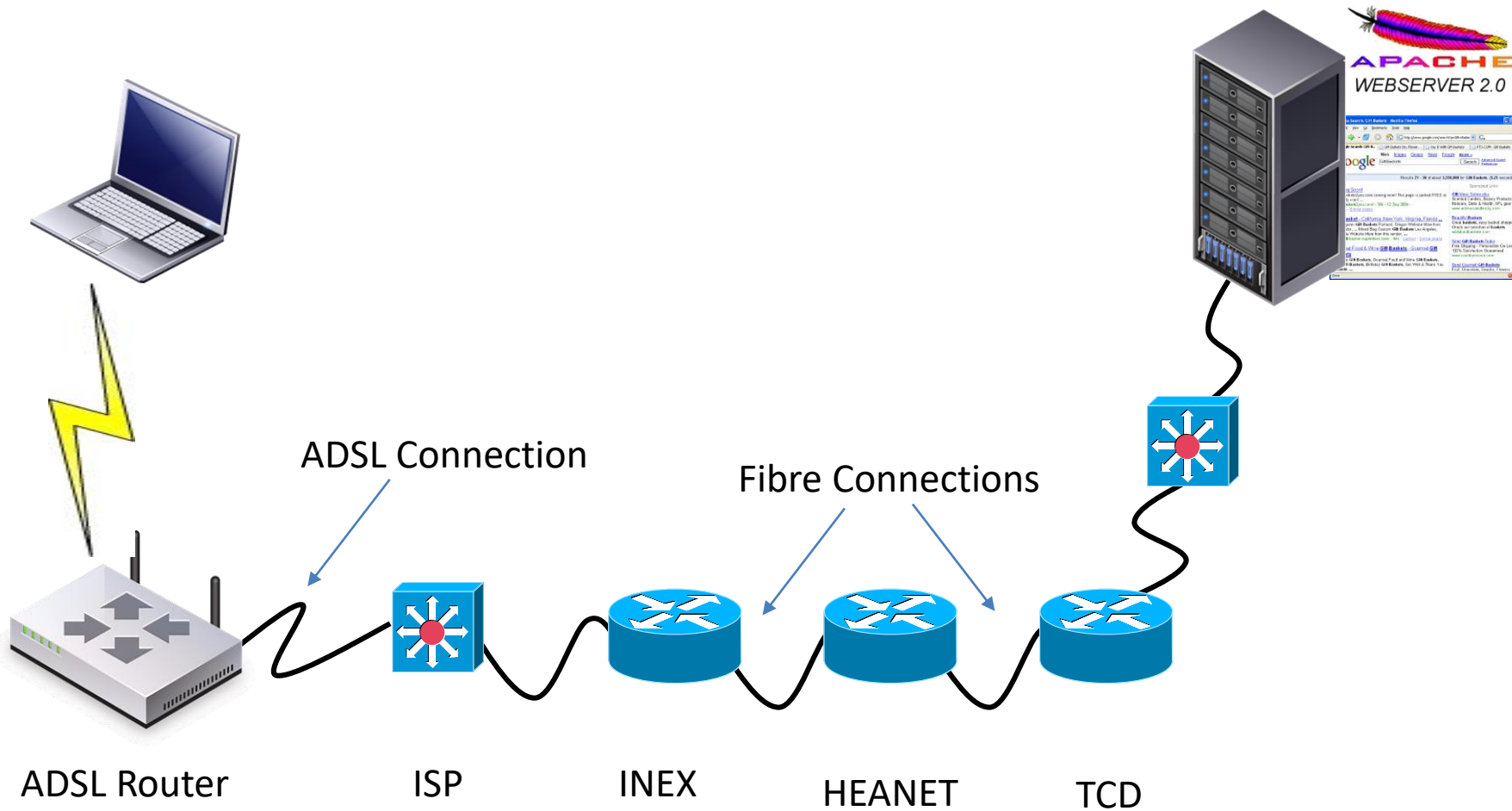


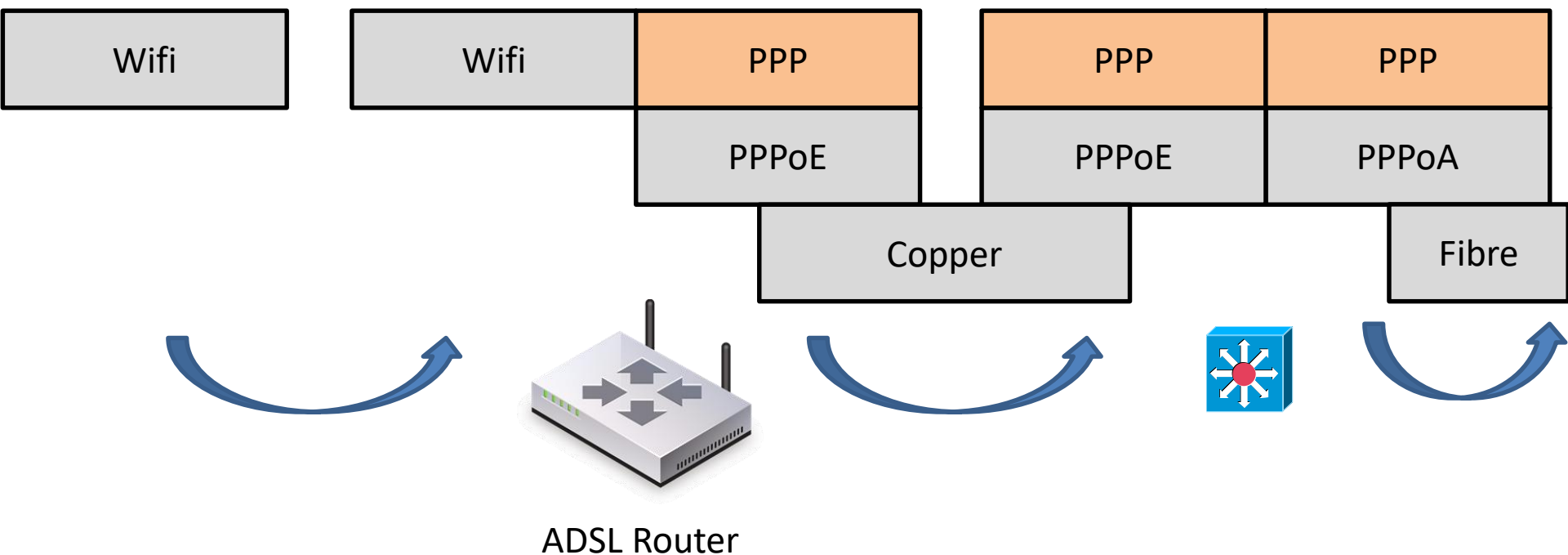
* Figure is courtesy of A. Tanenbaum

PPP – As Foundation for IP

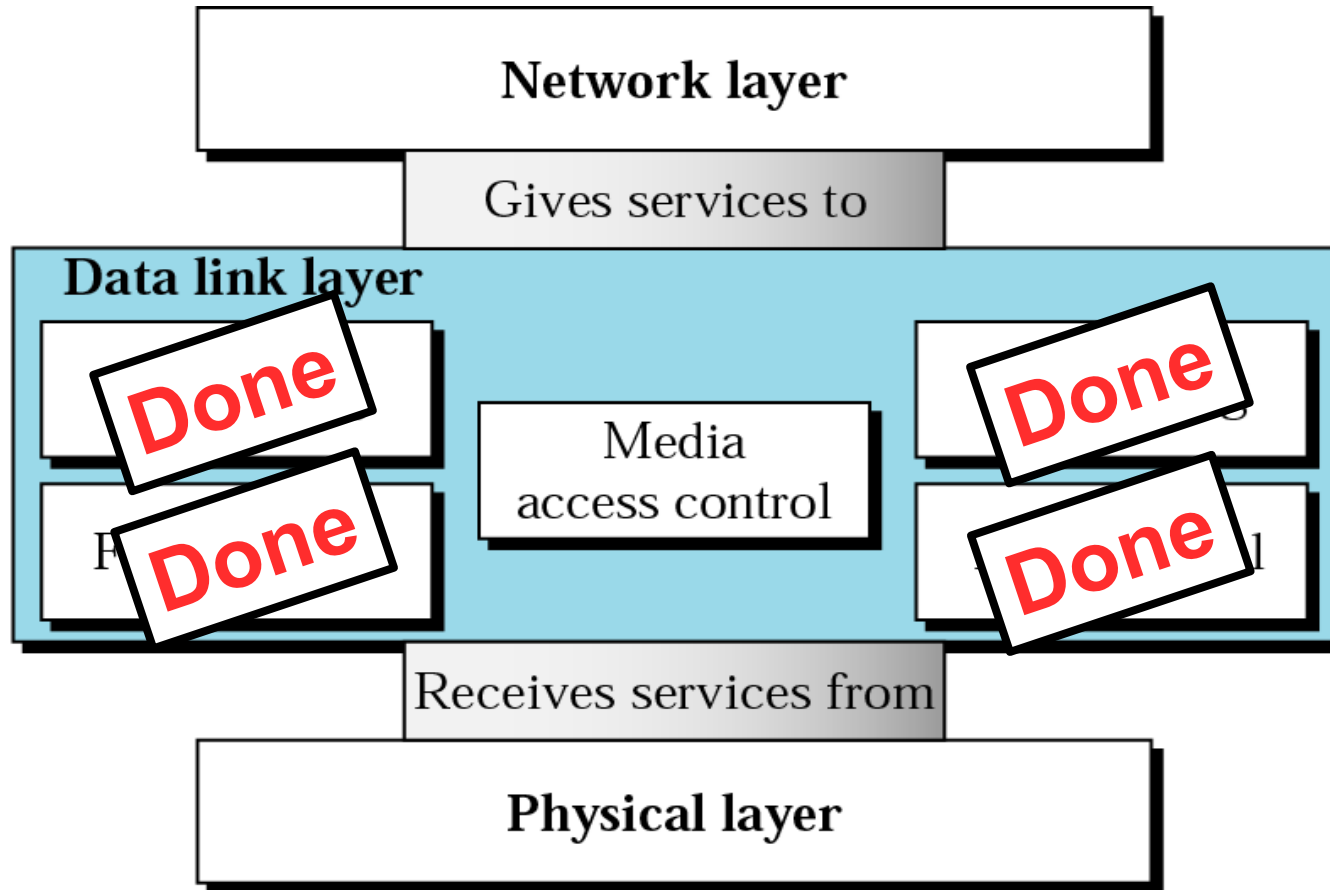


PPP in the HTML Use Case





Link Layer



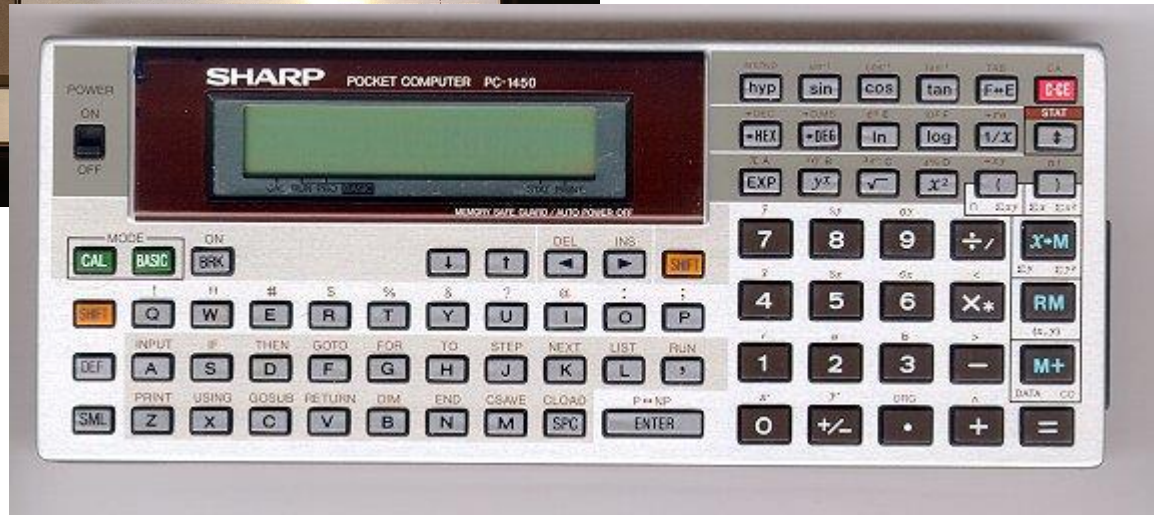
* Figure is courtesy of B. Forouzan

Why We do What We do...

- Assignment
- Exposure to programming
 - Threads, multiple programs, multiple classes, etc
- Programming is the tool of your job

When you graduate, you should be confident that you can handle anything that gets thrown at you.

Sharp PC-1450



Research in Food Prod. Technology



Intestines & Sausages



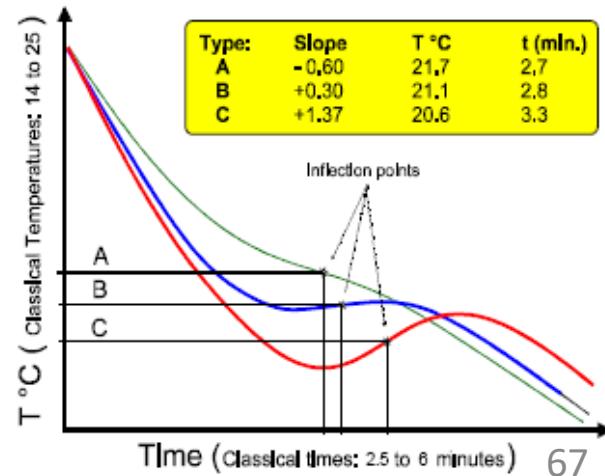
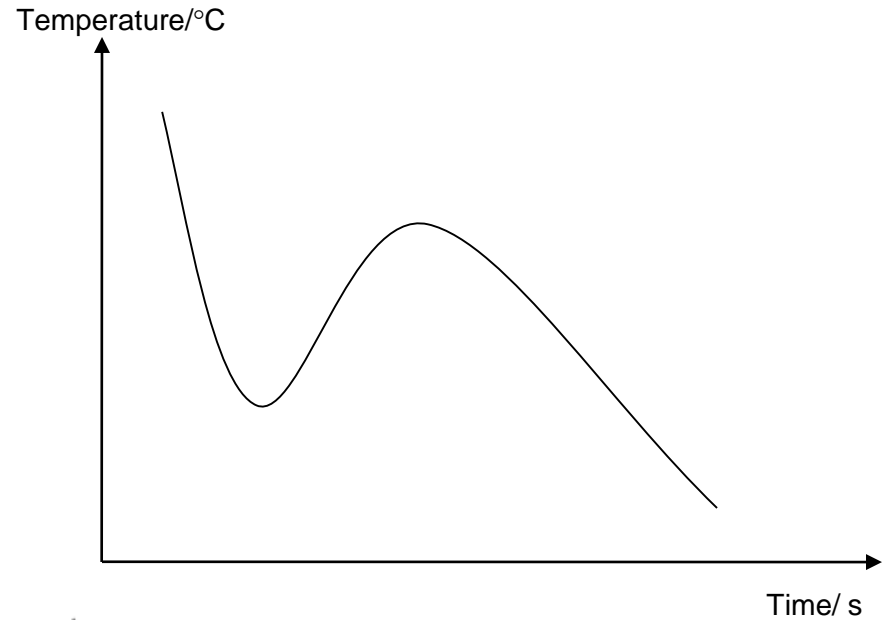
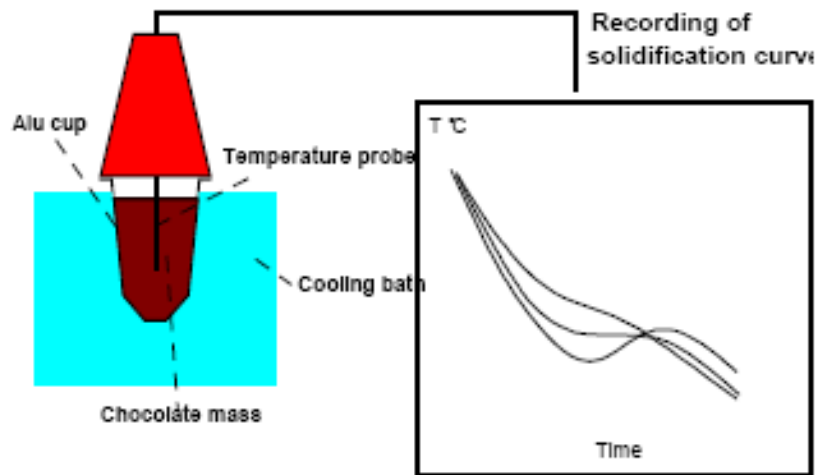
Chocolate-covered Marshmallows!!!



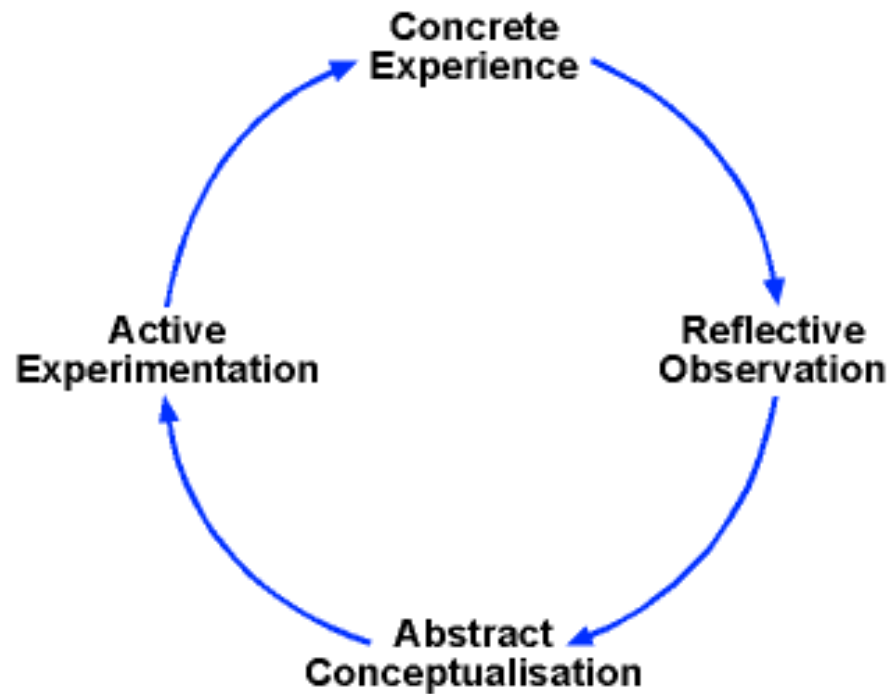
Quality of Chocolate



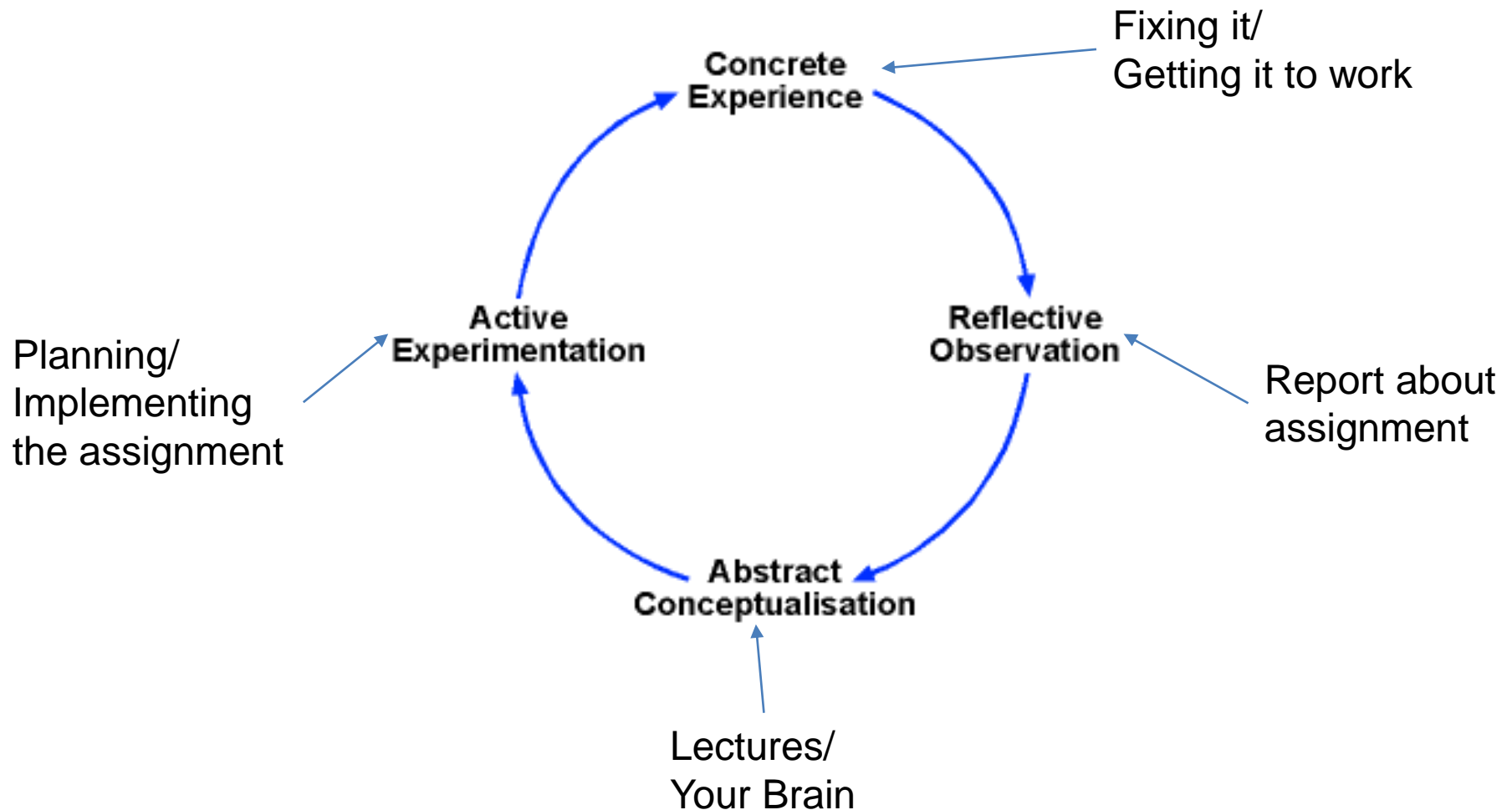
Principle Scheme



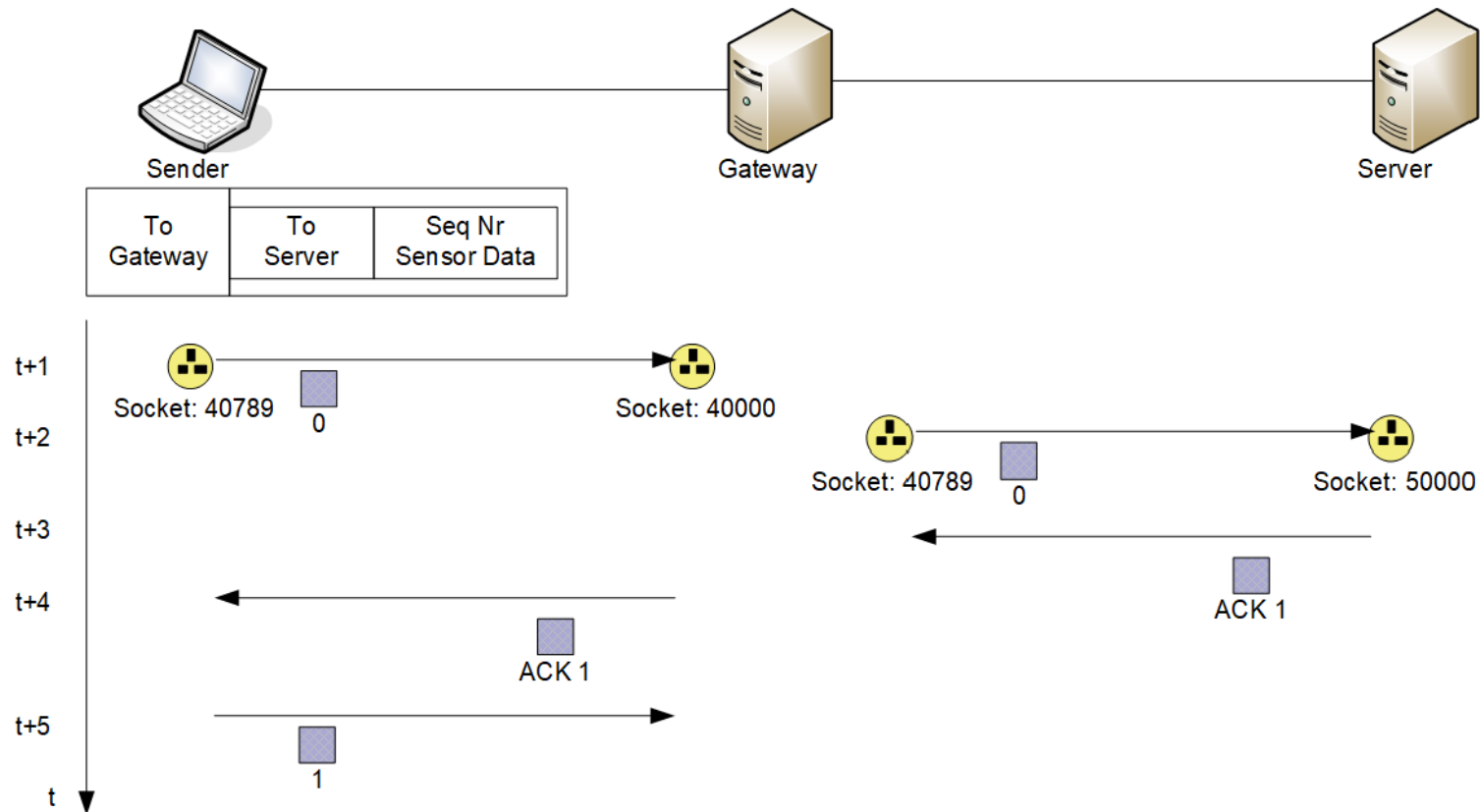
Kolb's Theory of Learning



Kolb's Theory of Learning



Assignment Scenario



Starting Steps for Assignment 1

1. Communication between client and gateway
 - a) Creating packet from client to server
 - b) Creating packet from client to gateway, encapsulating the packet to the server
 - c) Unpacking of packet for server at the gateway
2. Communication between gateway and server
3. Response from server to client
4. Stop & Wait between client and server
 - a) Timeouts
5. Two clients
6. 3+ clients
- ...
1000. Error Detection 😊



That's all
folks