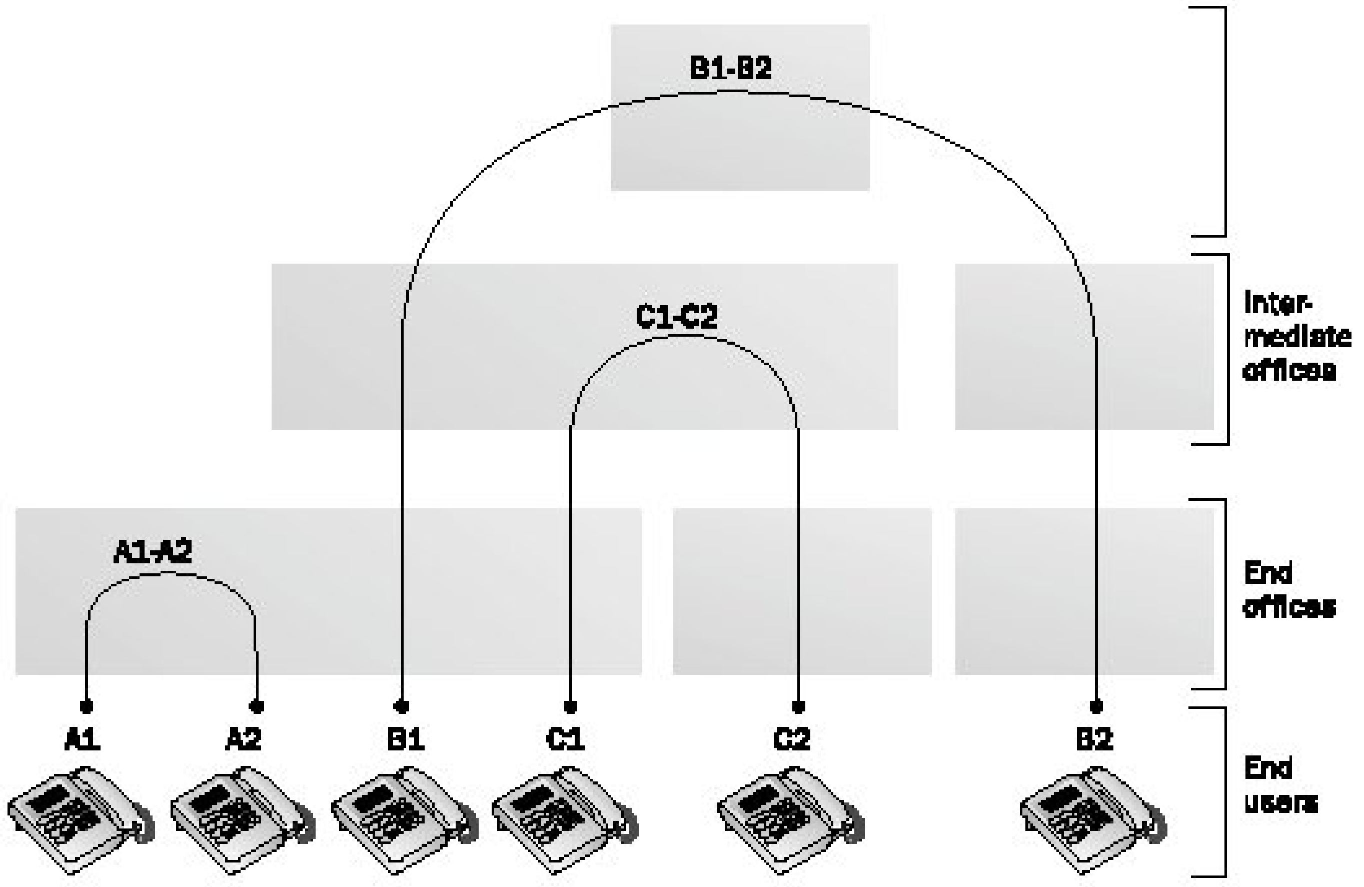
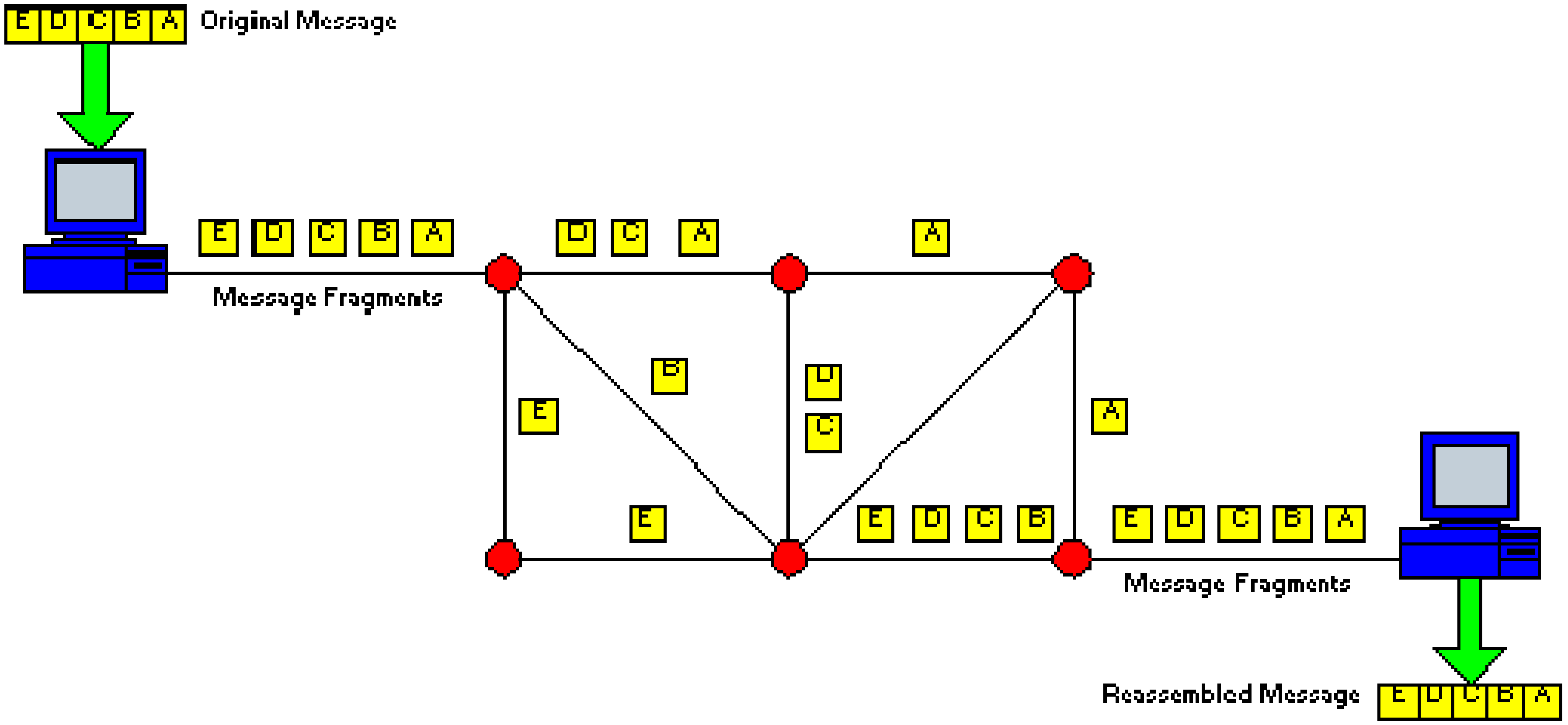
Chapter 2: TCP/IP Protocol Suit (Part A)

I. Packet Switching vs. Circuit Switching

- Circuit Switching
  - In circuit switching (e.g., telephone system), a caller must first establish a connection to a callee before any communication is possible.
  - During the connection establishment, resources remain allocated during the full length of a communication, after a circuit is established and until the circuit is terminated and the allocated resources are freed.
  - Wasting link capacity when a circuit does not carry as much traffic as the allocation permits.
- Packet Switching
  - **Packet switching:** the message is divided into segments, and each segment is routed from the source to the destinations independently.
  - Packet switching eliminates single point-of-failure problems.
  - Network communication resources are statistical multiplexing and an upper limit on the size of a transmitted entity result in fast, economical networks.



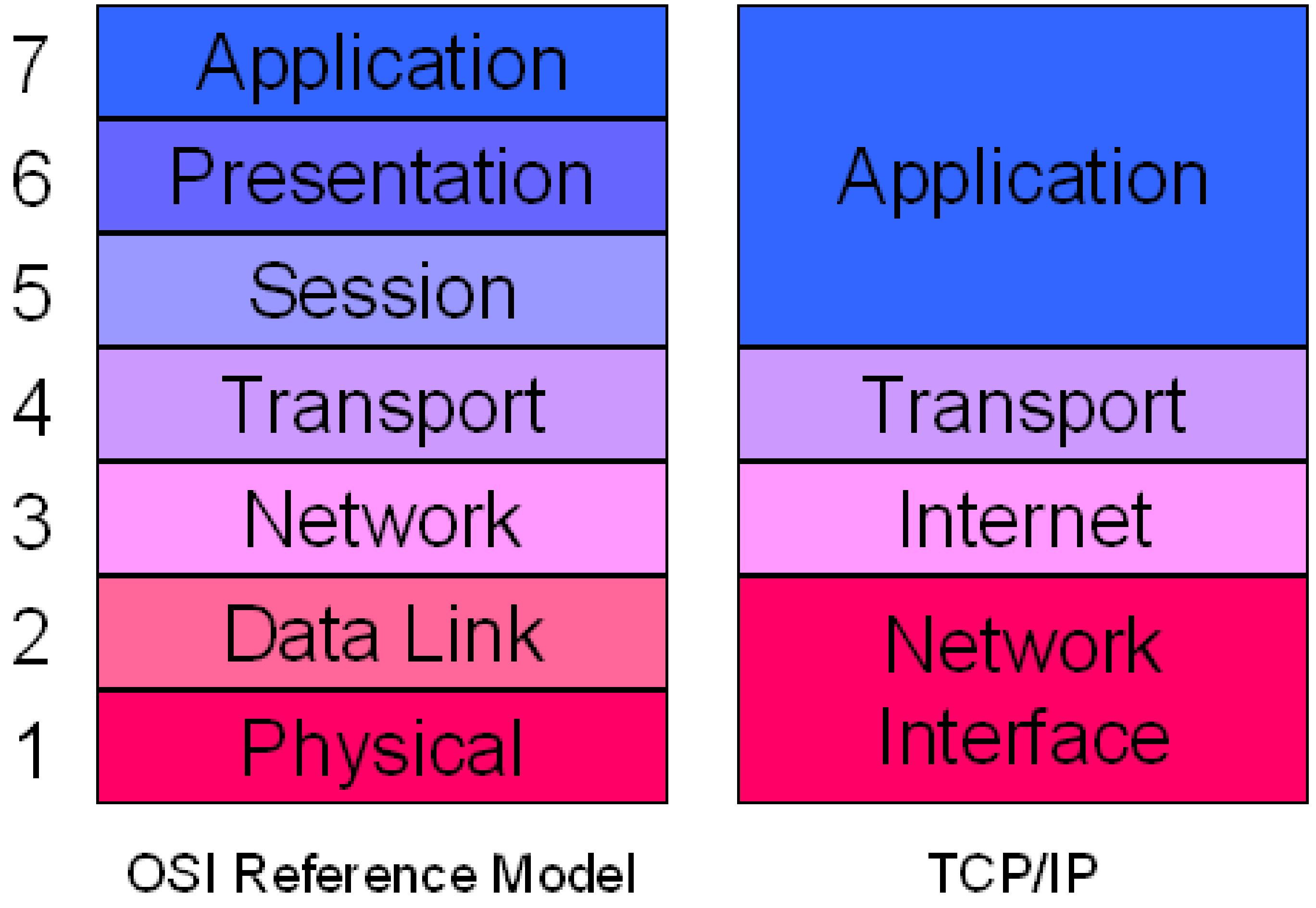
Circuit Switching



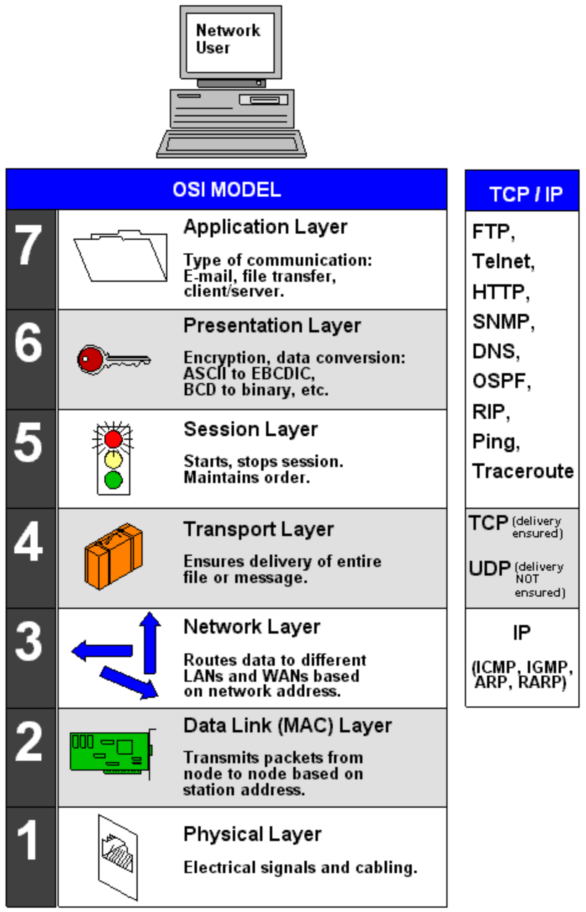
Packet Switching

III. Packet Switching: TCP/IP

(1) Layers

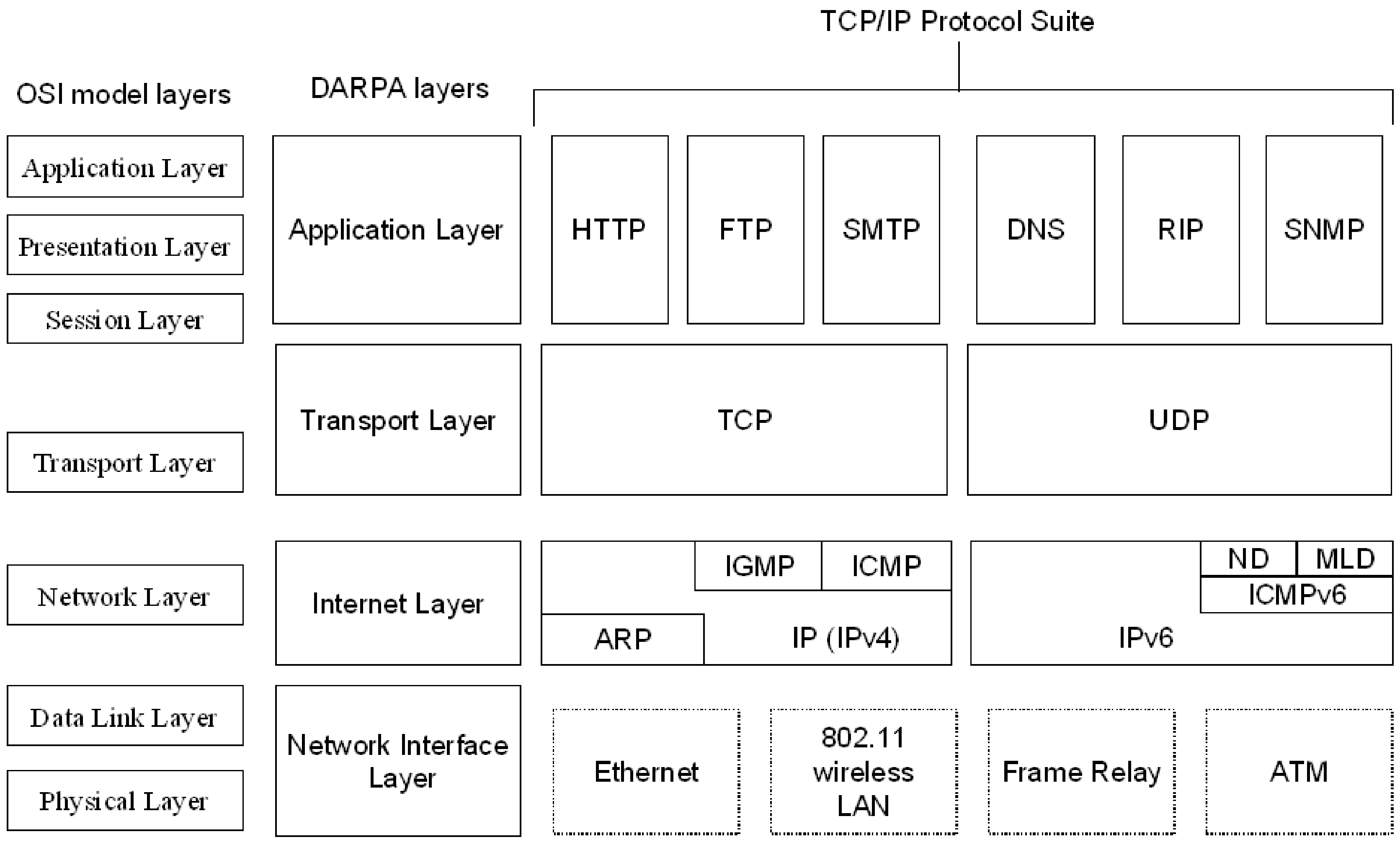


(2) Main functions provided by each layer

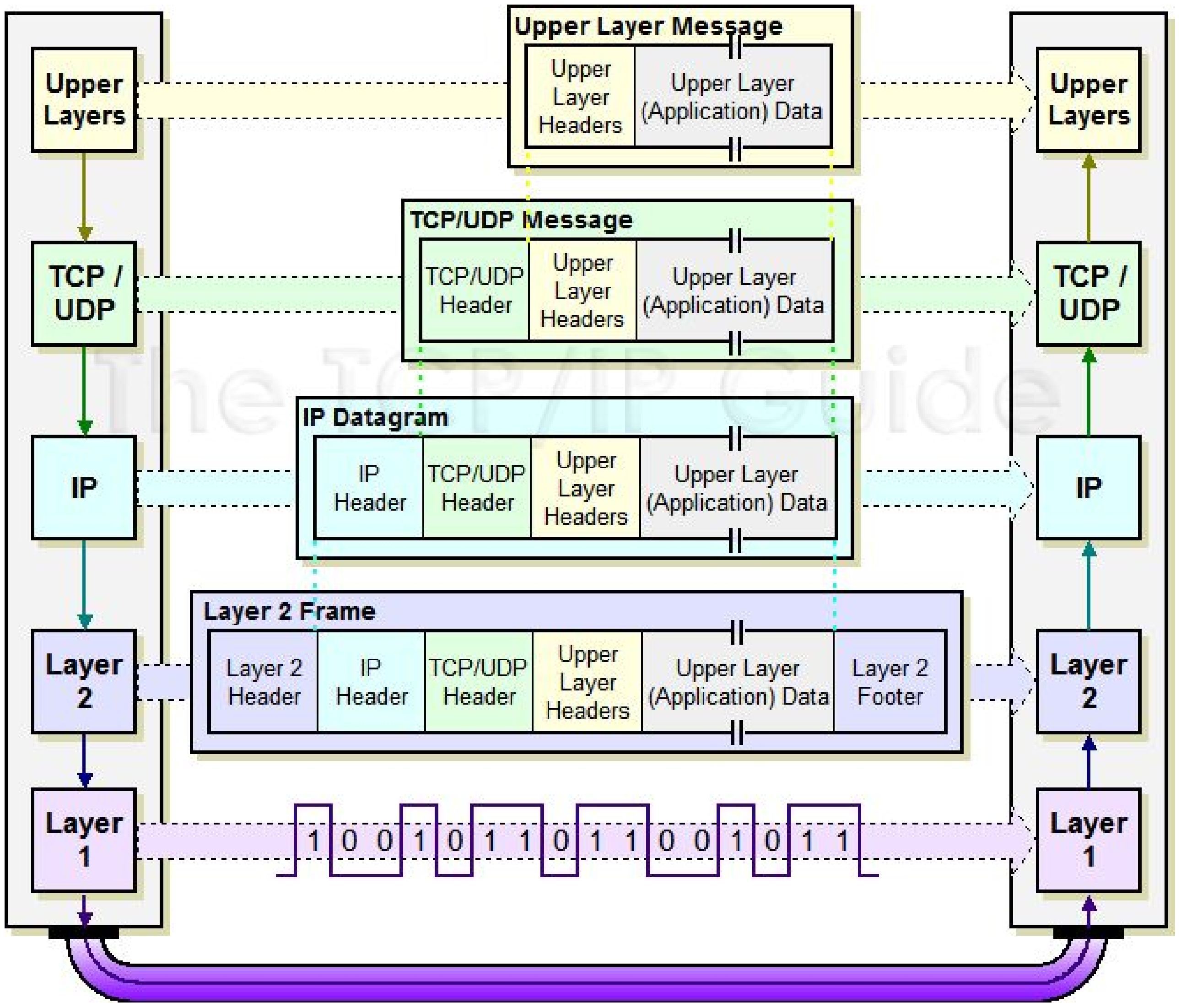


	OSI MODEL	TCP / IP
7	**Application Layer** Type of communication: E-mail, file transfer, client/server.	FTP, Telnet, HTTP, SNMP, DNS, OSPF, RIP, Ping, Traceroute
6	**Presentation Layer** Encryption, data conversion: ASCII to EBCDIC, BCD to binary, etc.	
5	**Session Layer** Starts, stops session. Maintains order.	
4	**Transport Layer** Ensures delivery of entire file or message.	TCP (delivery ensured) UDP (delivery NOT ensured)
3	**Network Layer** Routes data to different LANs and WANs based on network address.	IP (ICMP, IGMP, ARP, RARP)
2	**Data Link (MAC) Layer** Transmits packets from node to node based on station address.	
1	**Physical Layer** Electrical signals and cabling.	

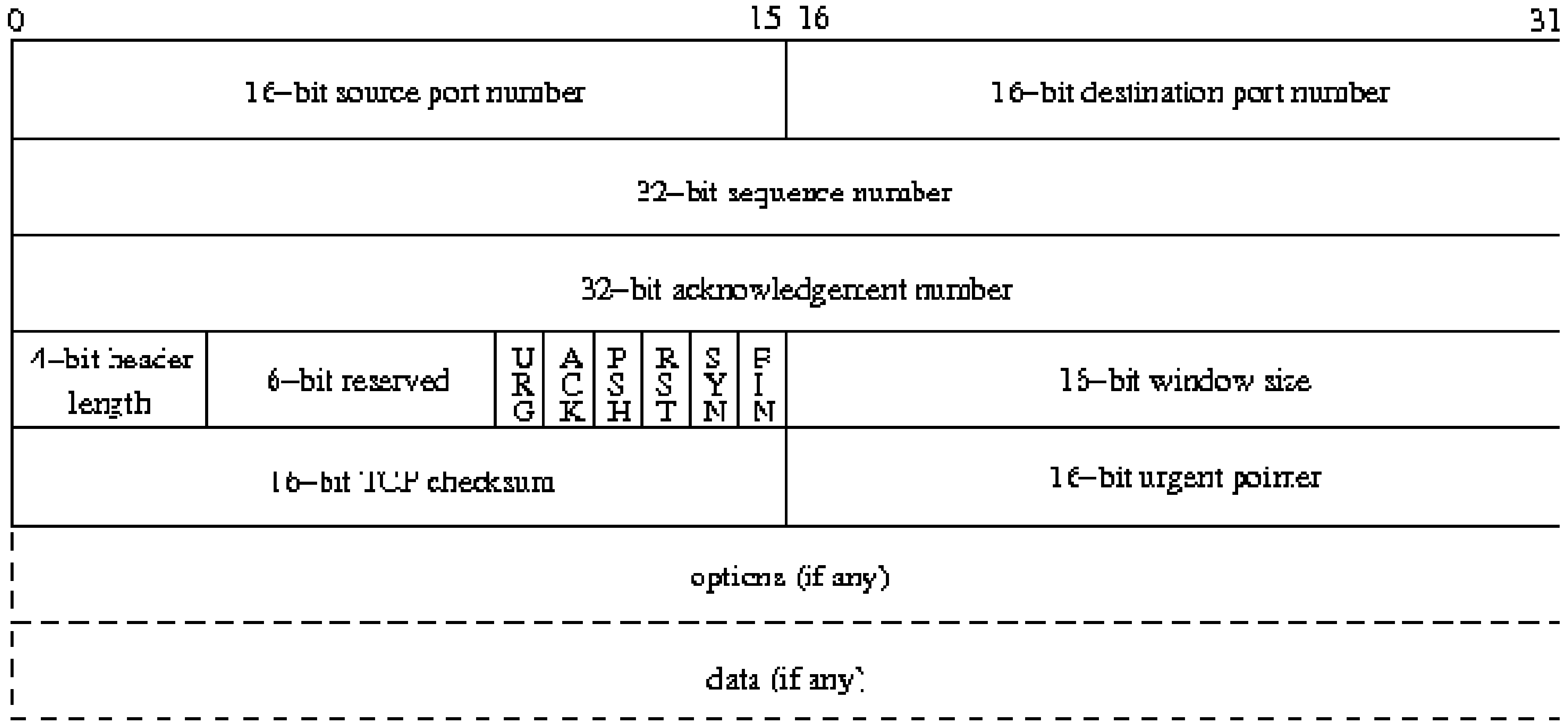
(3) TCP/IP Protocol Suit



- **Connection Oriented (TCP and IP)**
  - Two separate protocols are involved in handling TCP/IP datagrams.
  - TCP is responsible for breaking up the message into datagrams, reassembling them at the other end, resending anything that gets lost, and putting things back in the right order.
  - IP is responsible for routing individual datagrams.
  - TCP simply hands IP a datagram with a destination. IP doesn't know how this datagram relates to any datagram before it or after it.
- **TCP to send a datagram**
  Suppose there is a file you are trying to send to some other computer:
  - TCP breaks it up into manageable chunks.
  - TCP puts a header at the front of each datagram. This header actually contains at least 20 octets, but the most important ones are a source and destination "port number" and a "sequence number".

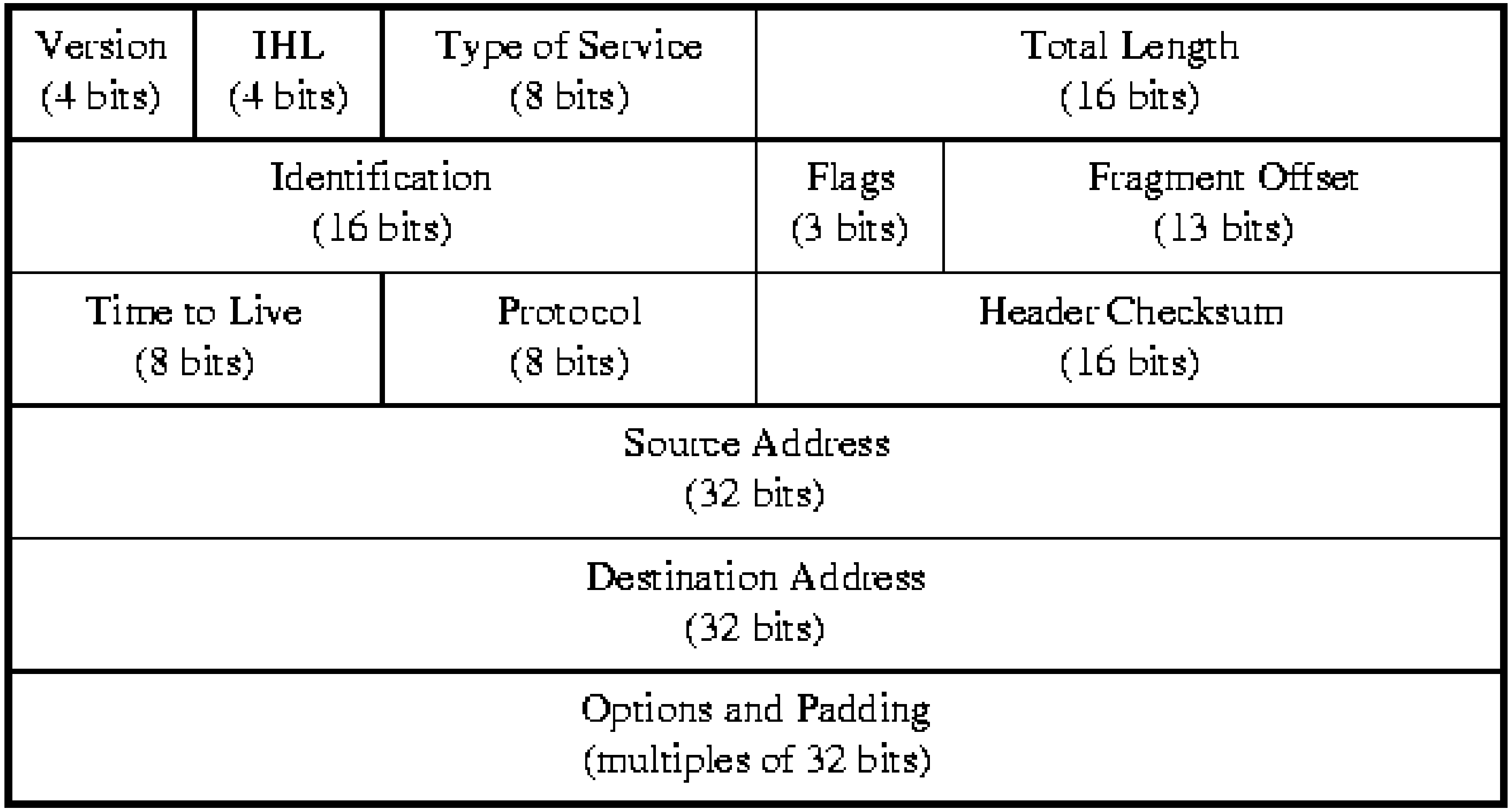

Upper Layers
TCP / UDP
IP
Layer 2
Layer 1
Upper Layer Message
Upper Layer Headers
Upper Layer (Application) Data
TCP/UDP Message
TCP/UDP Header
IP Datagram
IP Header
Layer 2 Frame
Layer 2 Header
Layer 2 Footer
1 0 0 1 0 1 1 0 1 1 0 0 1 0 1 1
The TCP/IP Guide

TCP Header


0
15
16
31
16–bit source port number
16–bit destination port number
32–bit sequence number
32–bit acknowledgement number
4–bit header length
6–bit reserved
URG
ACK
PSH
RST
SYN
FIN
16–bit window size
16–bit TCP checksum
16–bit urgent pointer
options (if any)
data (if any)

Internet Protocol (IP)

- TCP sends each of these datagrams to IP.
- IP doesn't care about what is in the datagram, or even in the TCP header.
- IP's job is simply to find a route for the datagram and get it to the other end.
- IP adds its own header which includes the source and destination Internet address (32-bit addresses, like 158.182.6.1), the protocol number, and another checksum (20 bytes for a basic header).
  - (1) **Source Internet address**: the address of your machine.
  - (2) **Destination Internet address**: the address of the other machine.
  - (3) **Protocol number**: tells IP at the other end to send the datagram to TCP. Although most IP traffic uses TCP, there are other protocols that can use IP, so you have to tell IP which protocol to send the datagram to.
  - (4) **Checksum**: allows IP at the other end to verify that the header wasn't damaged in transit. Note that TCP and IP have separate checksums.
  - (5) **Time to live**: a number that is decremented whenever the datagram passes through a system. When it goes to zero, the datagram is discarded.
- Finally IP sends the datagrams out on the line.



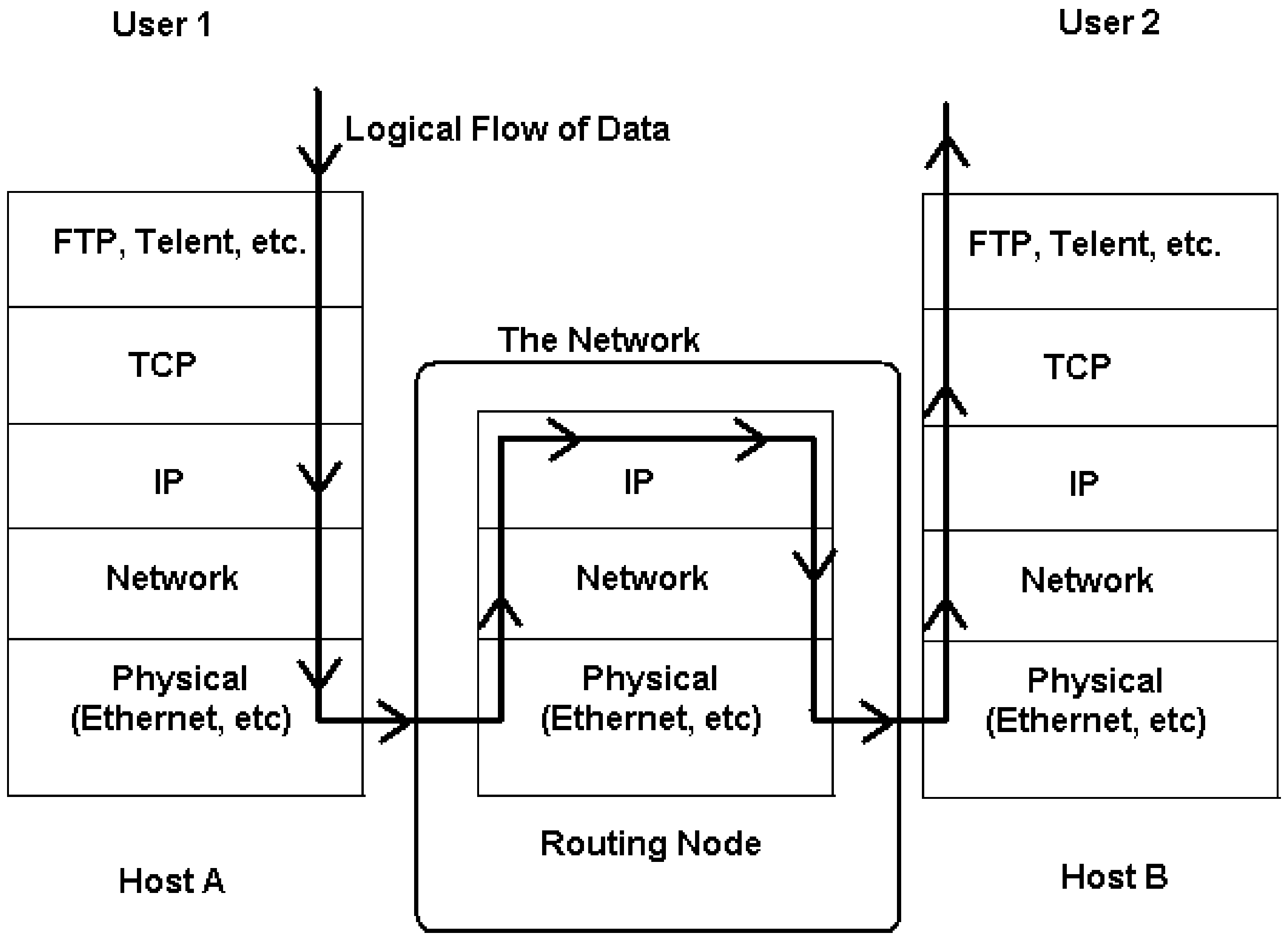
Version (4 bits)	IHL (4 bits)	Type of Service (8 bits)	Total Length (16 bits)
Identification (16 bits)	Flags (3 bits)	Fragment Offset (13 bits)	
Time to Live (8 bits)	Protocol (8 bits)	Header Checksum (16 bits)	
Source Address (32 bits)			
Destination Address (32 bits)			
Options and Padding (multiples of 32 bits)			

- **The Ethernet Level**
  - Most of our networks these days use Ethernet. So now we use Ethernet as an example.
  - Ethernet has its own addressing scheme. Each Ethernet controller comes with a 48 bits address built-in.
  - Each machine is supposed to pay attention only to packets with its own Ethernet address in the destination field.

    (1) **Ethernet**: When these packets are received, the Ethernet interface removes the Ethernet header and the checksum. The payload is the IP datagram, the Ethernet device driver passes the datagram up to IP.

    (2) **IP**: IP removes the IP header. IP looks at the IP protocol field. Since the protocol type is TCP, it passes the datagram up to TCP.

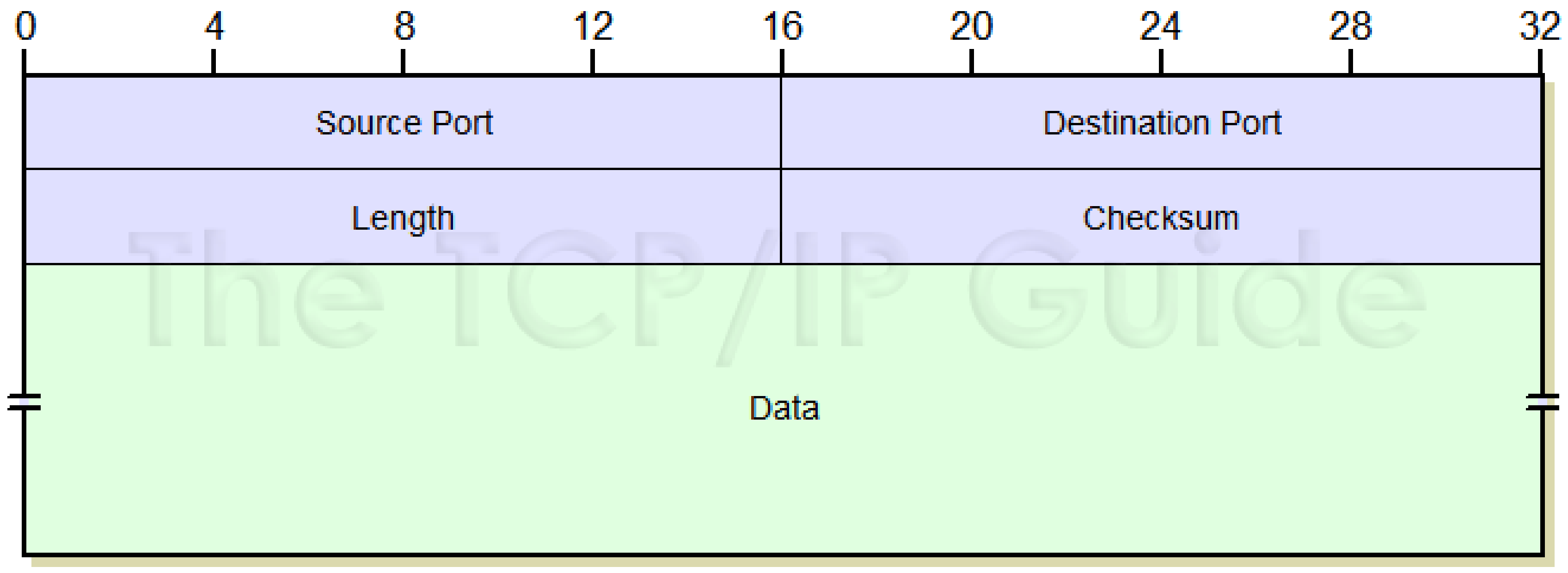
    (3) **TCP**: TCP looks at the sequence number. It uses the sequence numbers and other information to combine all the datagrams into the original file.


User 1
User 2
Logical Flow of Data
FTP, Telent, etc.
TCP
IP
Network
Physical (Ethernet, etc)
Host A
The Network
IP
Network
Physical (Ethernet, etc)
Routing Node
FTP, Telent, etc.
TCP
IP
Network
Physical (Ethernet, etc)
Host B

- **Connectionless Services: UDP (User Datagram Protocol)**
- UDP sits directly on top of the base Internet Protocol (IP).
- In general, UDP implements a fairly “lightweight” layer above the Internet Protocol.
- UDP's main purpose is to abstract network traffic in the form of datagrams.
- Applications that use UDP include client/server programs like video conferencing systems.

- **UDP Headers**

The UDP header consists of 8 bytes:



- The fields of UDP header including:
  (1) Source Port of the application that sent the data to UDP,
  (2) Destination Port of the intended recipient;
  (3) Checksum value may also be calculated;
  (4) Total length of a datagram.

- UDP port numbers allow different applications to maintain their own "channels" for data; both UDP and TCP use this mechanism to support multiple applications sending and receiving data concurrently.
- Some applications use static port numbers that are reserved for or registered to the application (e.g., port 80 for WWW, port 25 for Email, port 21 for FTP).
- In UDP, checksum is optional, as opposed to TCP where checksum is mandatory.

▪ UDP to send a datagram

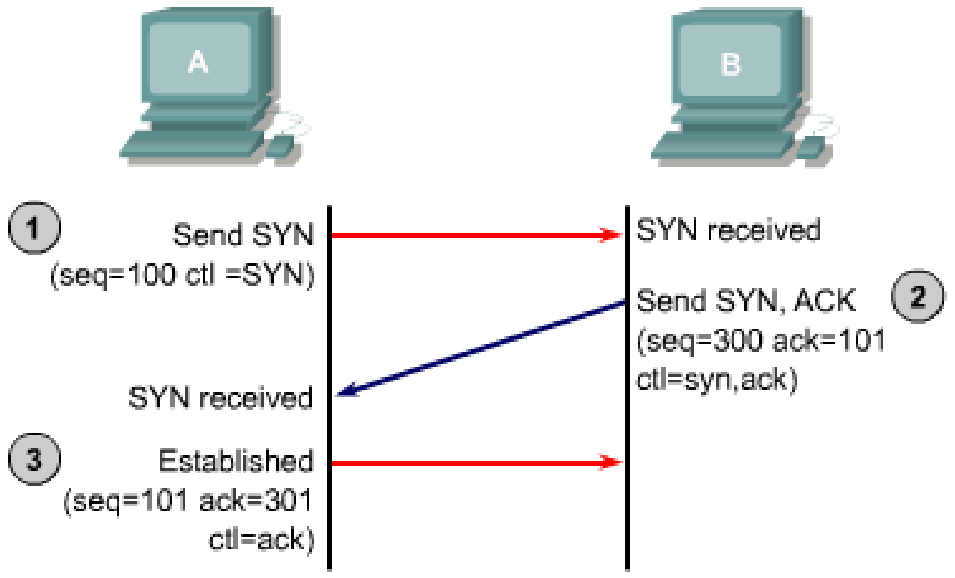
- UDP's only real task is to take data from higher-layer protocols and place it in UDP messages, which are then passed down to the Internet Protocol for transmission. The basic steps for transmission using UDP are:
  - (a) **Higher-Layer Data Transfer:** An application sends a message to the UDP software.
  - (b) **Message Encapsulation:** The higher-layer message is encapsulated into the **data field** of a UDP message.
  - (c) **Transfer Message to IP:** The UDP message is passed to IP for transmission.

IV. Differences between UDP and TCP

- **Characteristics of UDP**
  - ***No connection establishment:*** UDP does not need to establish a connection for transmission.
  - ***No connection state:*** TCP maintains connection state in the end systems (including Receive and Send buffers, Congestion Control Parameters, and Sequence and Acknowledgment Number Parameters, etc). UDP does not maintain connection state and does not track any of these parameters.
  - ***Smaller overhead:*** TCP has 20 bytes of header overhead in every segment, whereas UDP only has 8 bytes of overhead.
  - ***No flow control:*** The data speed of UDP is only constrained by the rate at which the application generates data, the capabilities of the source and the access bandwidth to the Internet.
  - ***No acknowledgement:*** UDP does not provide acknowledgments for data receiving
  - **No Guarantee for the delivery:** UDP does not provide any guarantees that its messages will arrive.

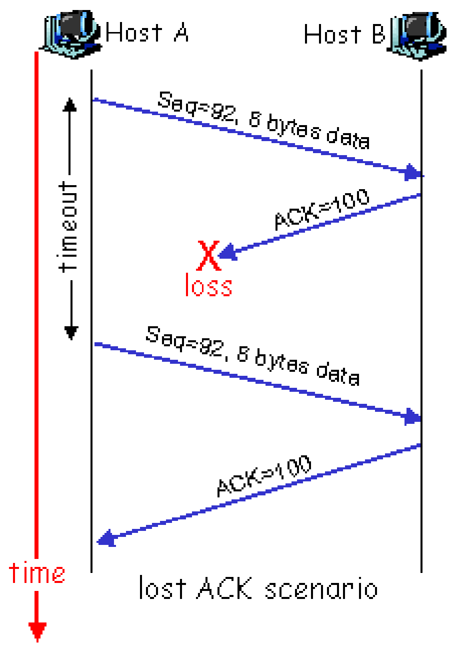
- ***No detection for the lost of messages:*** UDP does not detect lost messages and retransmit them.
- ***No guarantee for the datagrams' sequence:*** UDP does not ensure that data is received in the same order that they were sent.

- **Characteristics of TCP**
  - ***Addressing/Multiplexing:*** TCP is used by many different applications for their transport protocol. TCP will multiplex the data received from these different processes, and send out using the underlying network-layer protocol.
  - ***Connection Establishment, Management and Termination:*** TCP provides a set of procedures that devices follow to negotiate and establish a connection over which data can travel. TCP includes logic for managing connections and handling problems that may result with them.

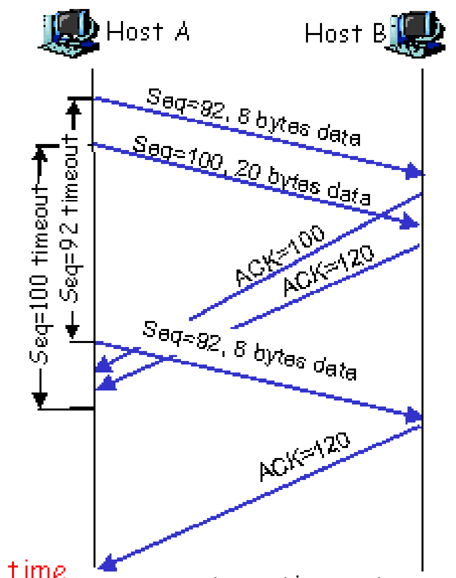


- ***Data Handling and Packaging:*** TCP defines a mechanism for interactions with higher layers. When receiving data from higher layers, the data will be packaged into messages to be sent to the destination. The destination software un-packages the data and gives it to the application on the destination machine.
- ***Data Transfer:*** The TCP implementation is responsible for the transfer of packaged data to the TCP process on the other device.
- ***Providing Reliability and Transmission Quality Services:*** TCP includes a set of services and features that allow an application to consider the sending of data using the protocol to be **reliable**.

TCP: retransmission scenarios

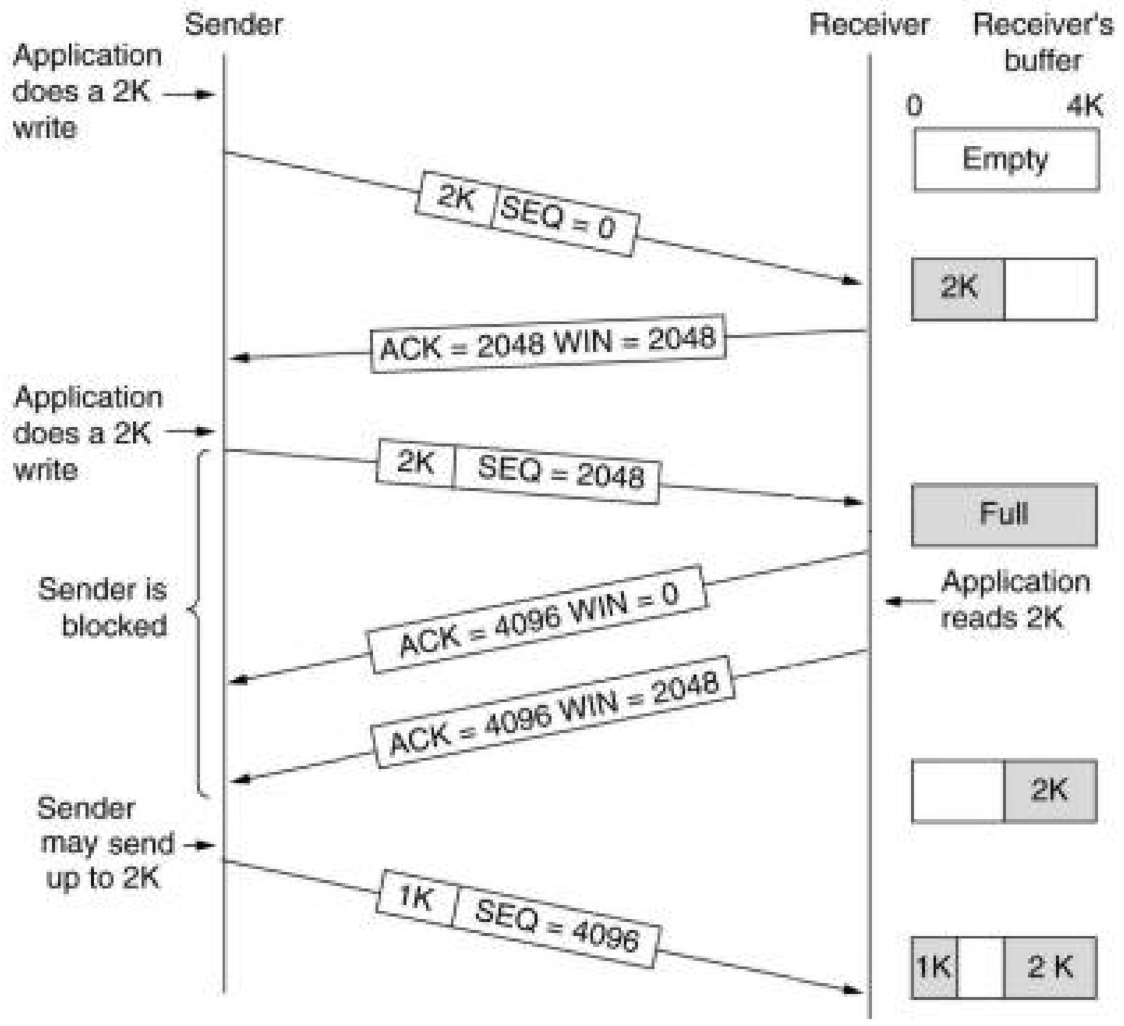


lost ACK scenario

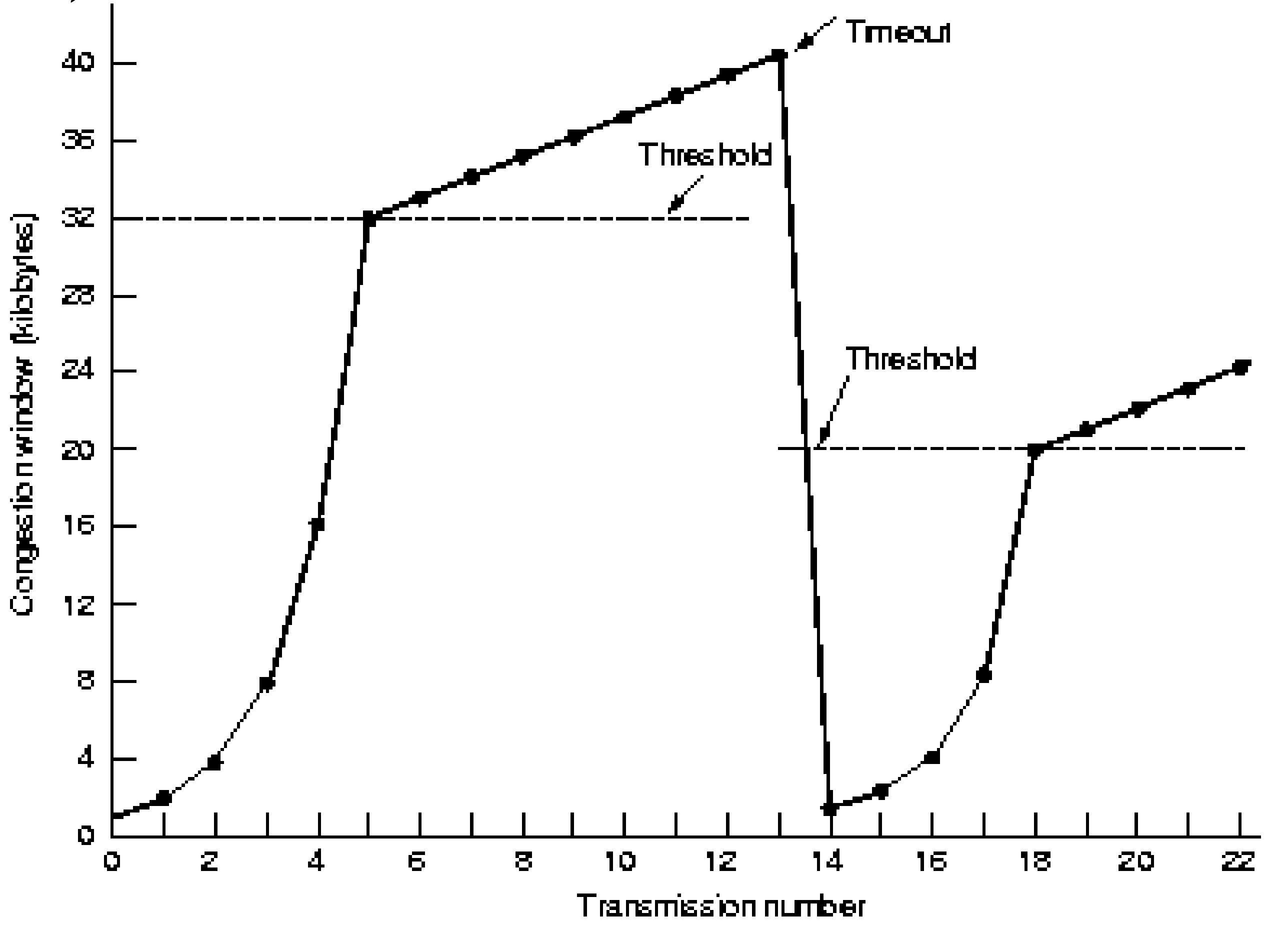


premature timeout, cumulative ACKs

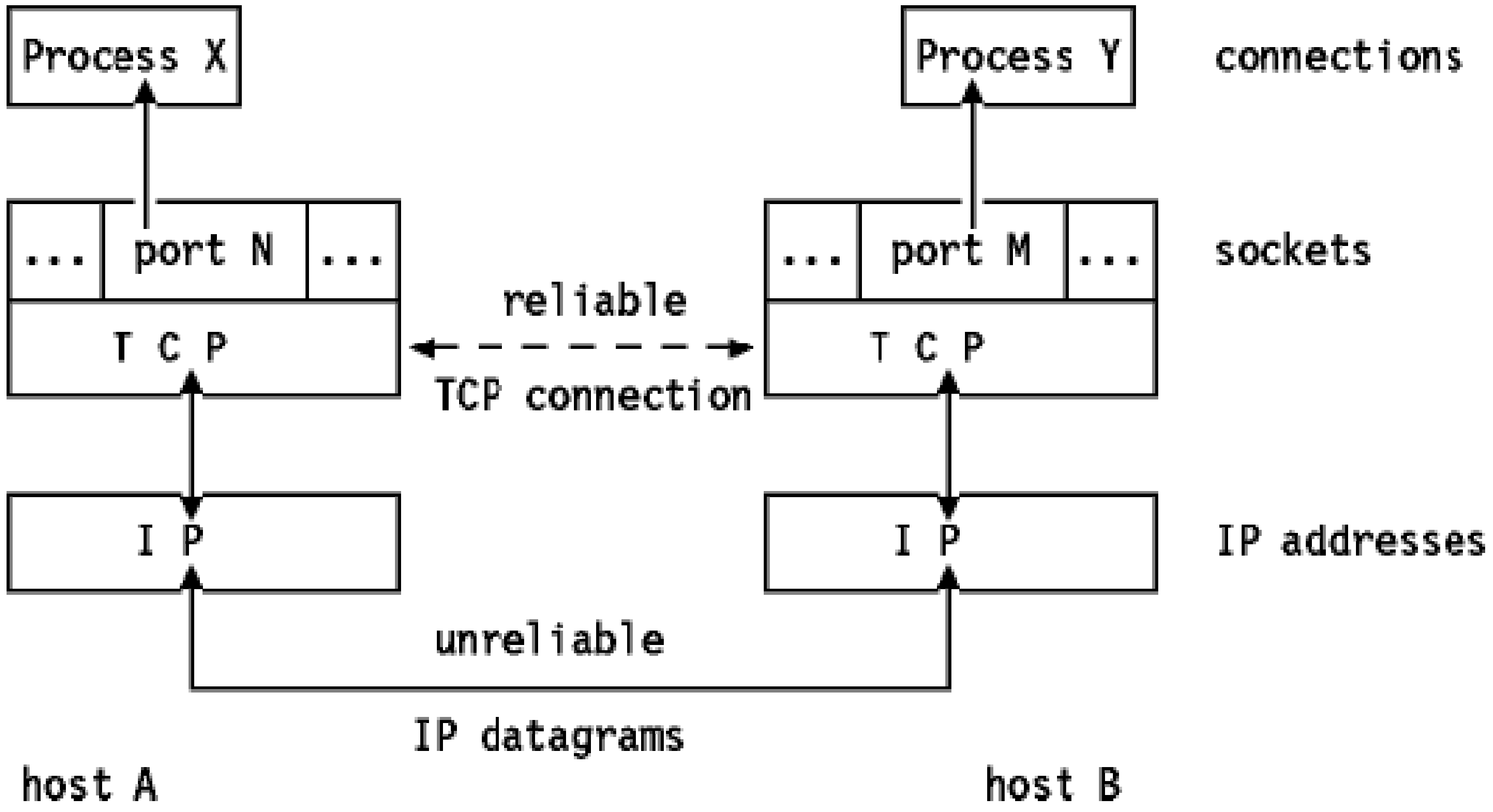
3: Transport Layer 3b-7


Sender
Receiver
Receiver's buffer
0
4K
Application does a 2K write
Empty
2K
SEQ = 0
2K
ACK = 2048 WIN = 2048
Application does a 2K write
2K
SEQ = 2048
Full
Sender is blocked
ACK = 4096 WIN = 0
Application reads 2K
ACK = 4096 WIN = 2048
2K
Sender may send up to 2K
1K
SEQ = 4096
1K
2 K

- ***Providing Flow Control and Congestion Avoidance Features:*** TCP allows the flow of data between two devices to be controlled and managed (e.g., network congestion).



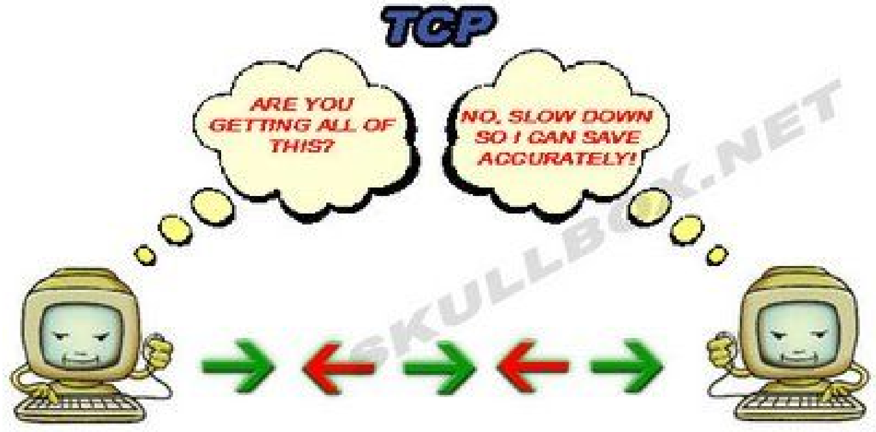
- **Functions Not Performed By TCP**
  - ***Does not Specify Application Use:*** TCP defines the transport protocol. It does not describe specifically how applications are to use TCP.
  - ***Does not Provide Security:*** TCP does not provide any mechanism for ensuring the authenticity or privacy of data it transmits. The security should be provided on the application layer (such as IPSec, etc).
  - ***Does not Maintain Message Boundaries:*** TCP sends data as a continuous stream, not as discrete messages. It is up to the application to specify where one message ends and the next begins.
  - ***Does not Guarantee Communication:*** TCP provides reliable communication only by detecting failed transmissions and re-sending them.

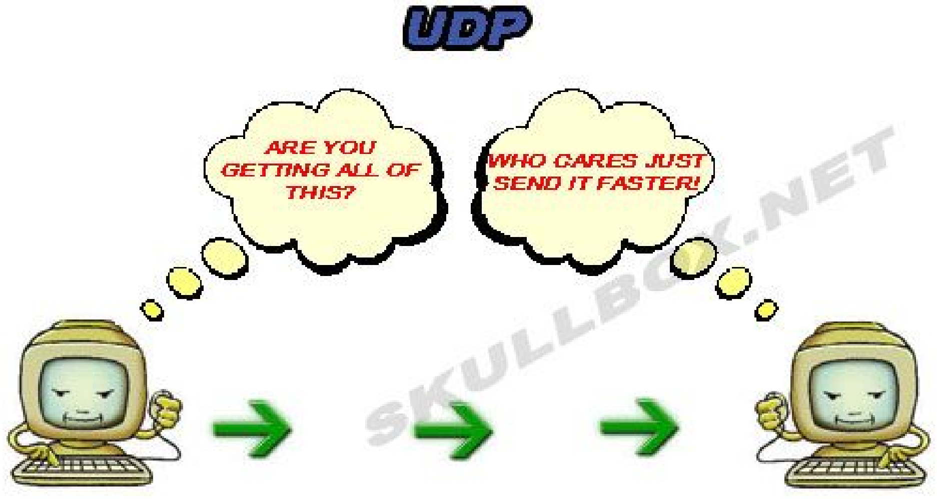

Process X
Process Y
connections
... port N ...
... port M ...
sockets
reliable
T C P
TCP connection
T C P
I P
I P
IP addresses
unreliable
IP datagrams
host A
host B

- **UDP vs. TCP: Speed vs. Reliability**
  - TCP includes support for guaranteed delivery: **acknowledgement**, **retransmission** and **arranging packets in sequence**.
  - UDP does not implement guaranteed message delivery. A UDP datagram can get **lost** on the way from sender to receiver, and the protocol itself does nothing to detect or report this condition.
  - Unlike TCP, UDP provides no guarantees that the order of delivery is preserved. For example, a client application might send the following four datagrams to a server: *Data1, Data2, Data3, Data4;*

    But UDP may present the datagrams to the server-side application in this order instead: ***Data3***, *Data1*, ***Data4***, *Data2*.
  - The key advantages of UDP are speed and simplicity.

TCP vs. UDP


TCP
ARE YOU GETTING ALL OF THIS?
NO, SLOW DOWN SO I CAN SAVE ACCURATELY!
SKULLBOX.NET


UDP
ARE YOU GETTING ALL OF THIS?
WHO CARES JUST SEND IT FASTER!
SKULLBOX.NET

Application	**Application-layer protocol**	**Underlying Transport Protocol**
Electronic mail	SMTP	TCP
Remote terminal access	Telnet	TCP
Web	HTTP	TCP
File transfer	FTP	TCP
Remote file server	NFS	typically UDP
Streaming multimedia	proprietary	typically UDP
Internet telephony	proprietary	typically UDP
Network Management	SNMP	typically UDP
Routing Protocol	RIP	typically UDP
Name Translation	DNS	typically UDP