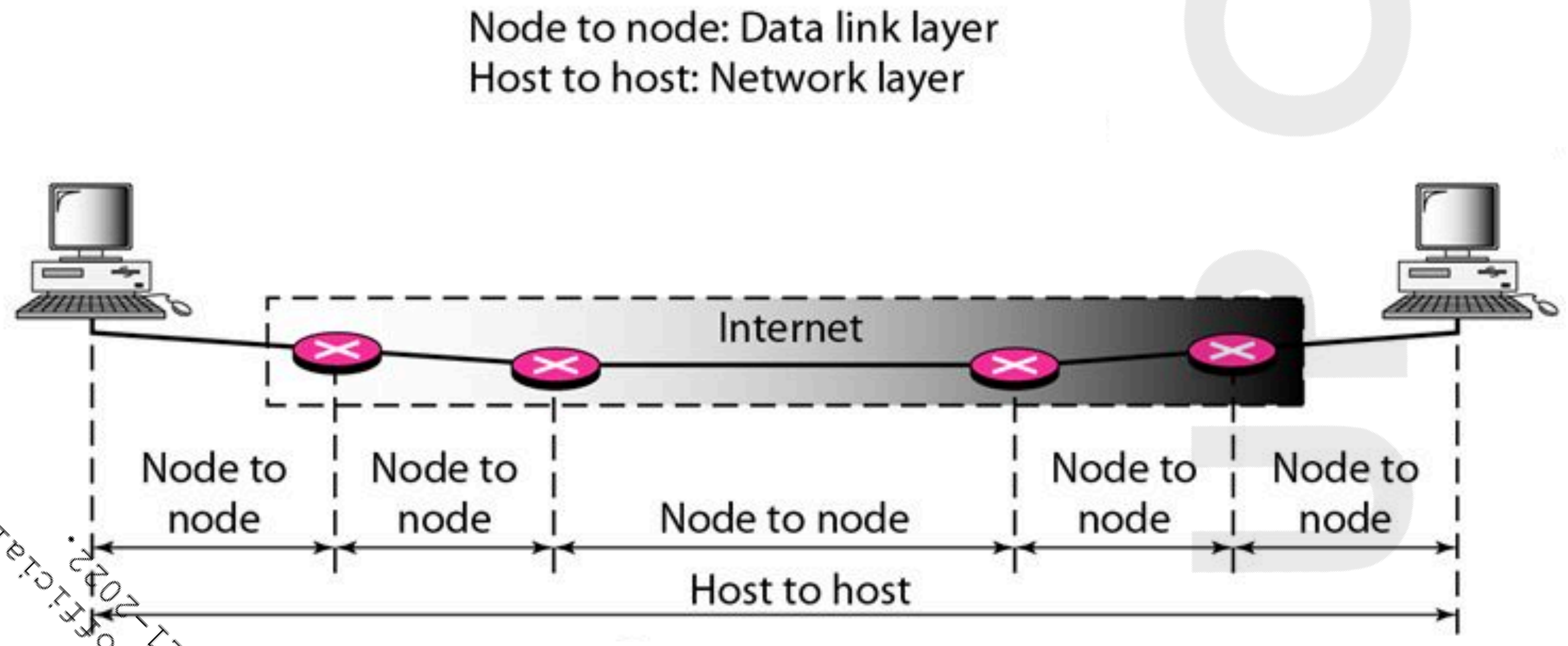
The Transport Layer

Prof. Dhong Fhel K. Gom-os

Recap

- **Data Link Layer:** node-to-node (hop-to-hop) delivery; the same link.
- **Network Layer:** source-to-destination (host-to-host) delivery; multiple networks (links).



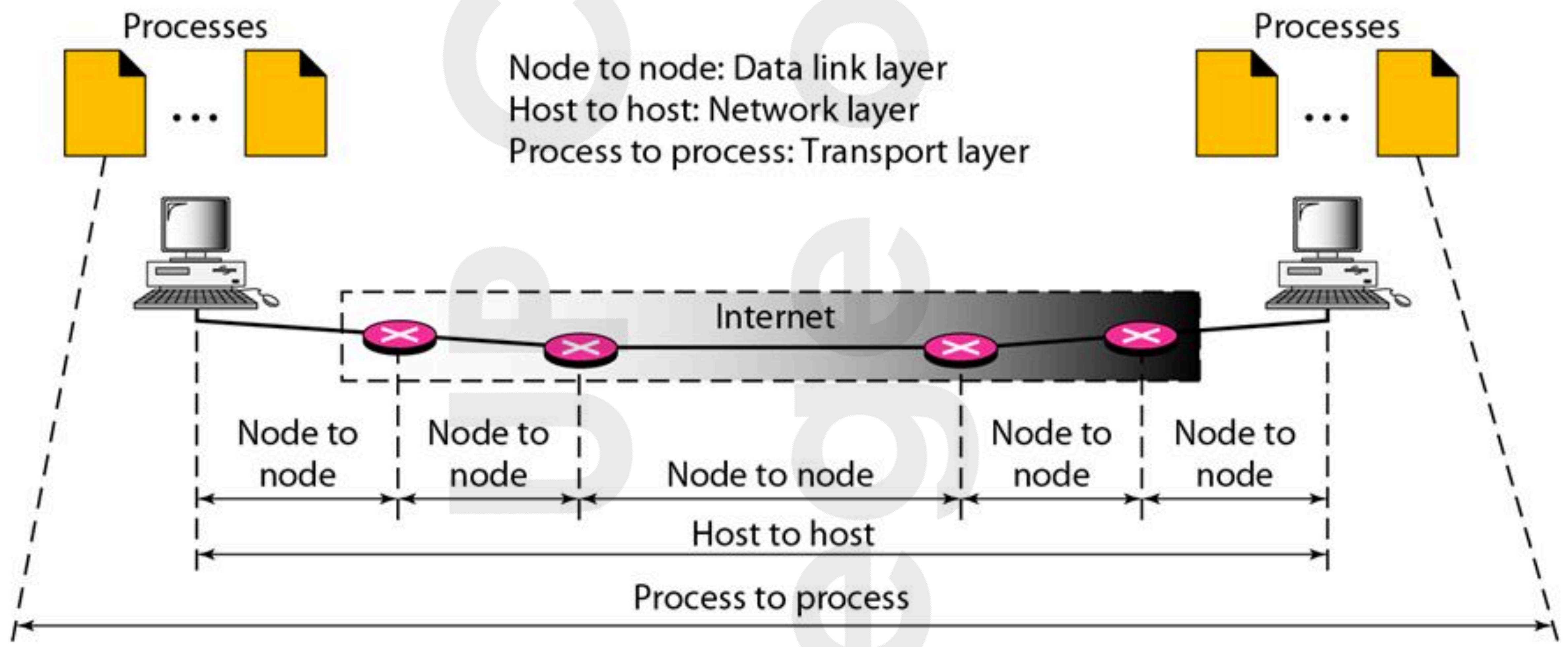
What do you think Transport Layer is responsible for?

Transport Layer

- Responsible for **process-to-process** delivery of the entire message.
- A **process** is an application program running on a host.
- Network Layer does **not recognize** relationship between packets; treats each one **independently**.
- Transport Layer ensures **whole message** arrives intact and in order.
  - Delivery not only from one computer to another but **specific process** on one computer to a specific process on the other.
  - **Real** communication takes place between two processes not between nodes.

Process-To-Process Delivery

- Concern: Multiple processes may be running on the source and destination hosts.
- Goal: Deliver data from a process on a source host to the corresponding process running on the destination host.



Node to node: Data link layer
Host to host: Network layer
Process to process: Transport layer

Client/Server Paradigm

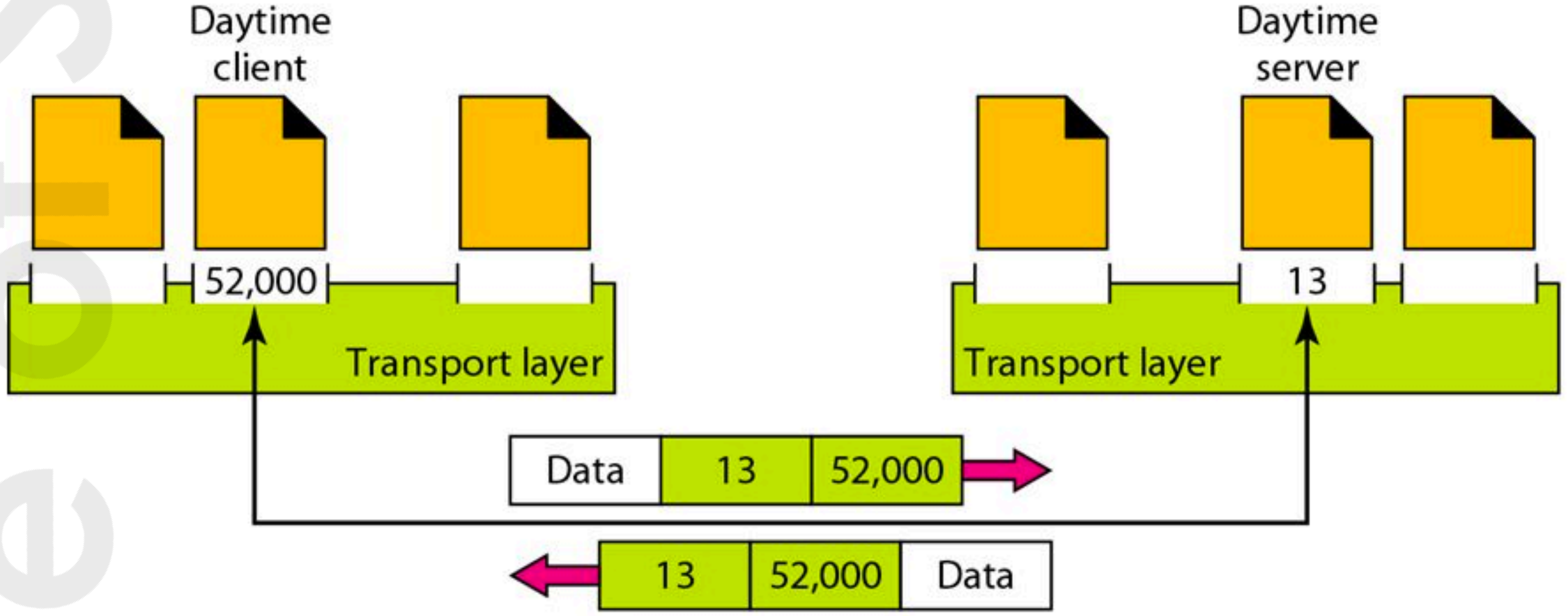
- Most common way to achieve a process-to-process communication.
- A process on the local host, called a **client**, needs services from a process usually on the remote host, called a **server**.
- Example: To get the day and time from a remote machine, we need a Daytime client process running on the local host and a Daytime server process running on a remote machine.
- Modern OS supports both multiuser and multiprogramming environments; servers can run several programs at the same time just as the local computers can run several client programs at the same time.
- If in Network Layer we need IP address, how about in Transport Layer?

Port Number

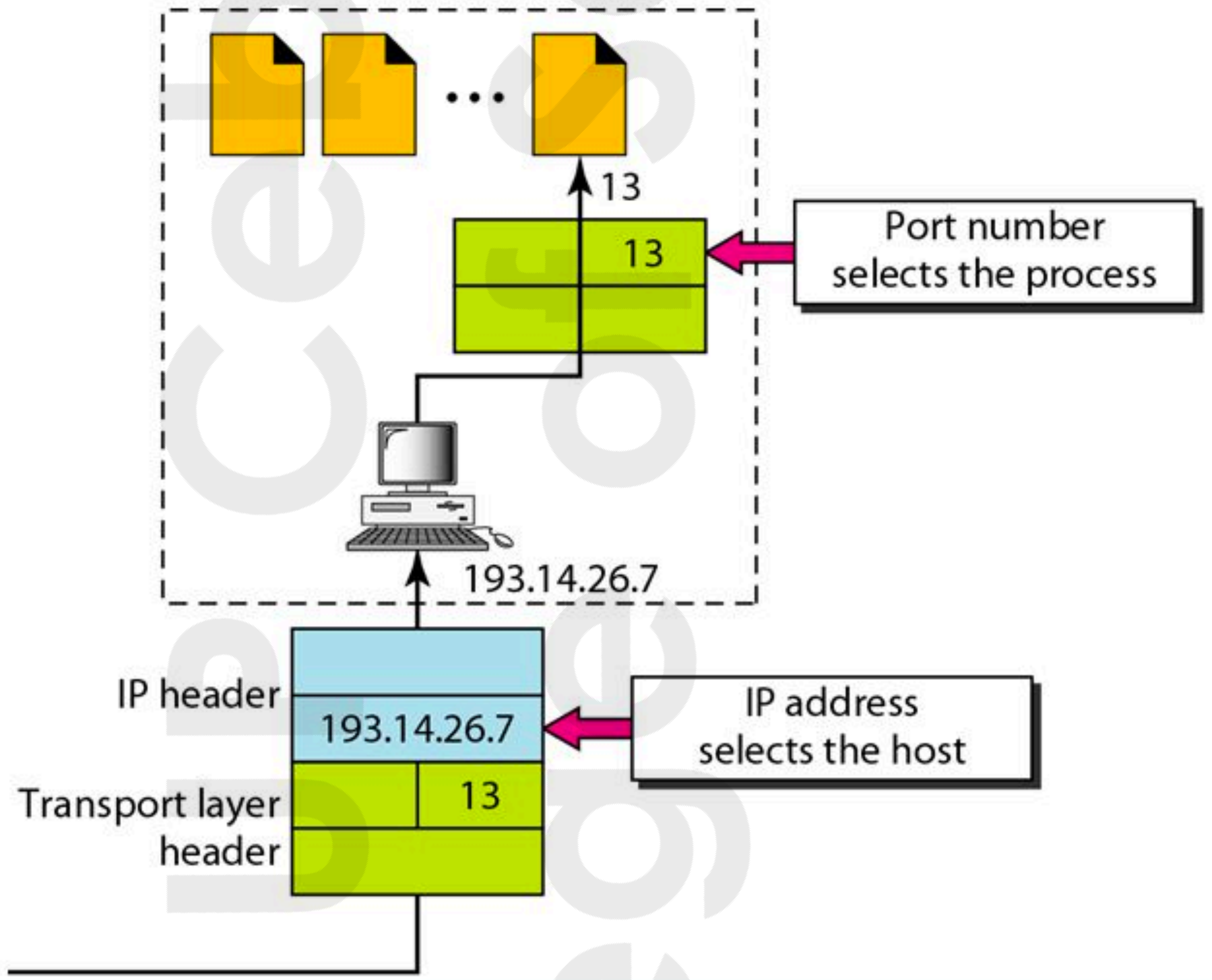
- In Internet model, 16-bit integer between 0 and 65535.
- Used to choose among multiple processes running on the destination host.
- **Destination port number** is needed for delivery; **source port number** is needed for the reply.
- Randomly assigned by transport layer software on the client (ephemeral).
- Question: Why is it not practical to randomly assign port number to processes on a server?
- Process at client site will not know port number of specific process to communicate.
- Possible solution: Send special packet to request for port number of a specific process.
- Disadvantage?
- Efficient solution?

Universal Port Numbers for Servers

- Well-known port numbers.
- Some exceptions: (for example) there are clients that are assigned well-known port numbers.
- Every client process knows the well-known port number of the corresponding server process.
- Example, Daytime client process can use ephemeral (temporary) port number 52000, Daytime server process must use the well-known (permanent) port number 13.

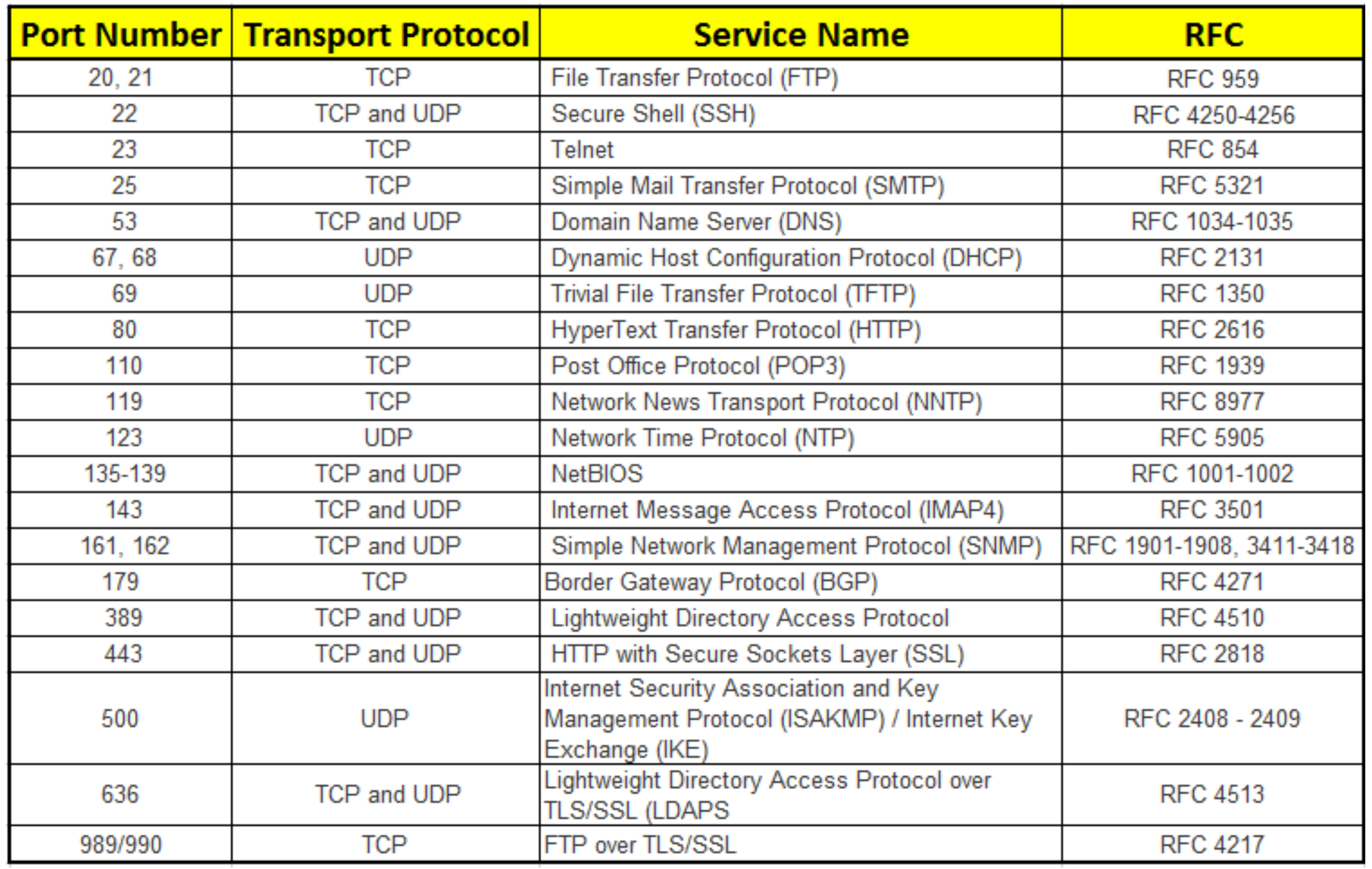


IP address defines destination host, port defines process…



IANA Ranges

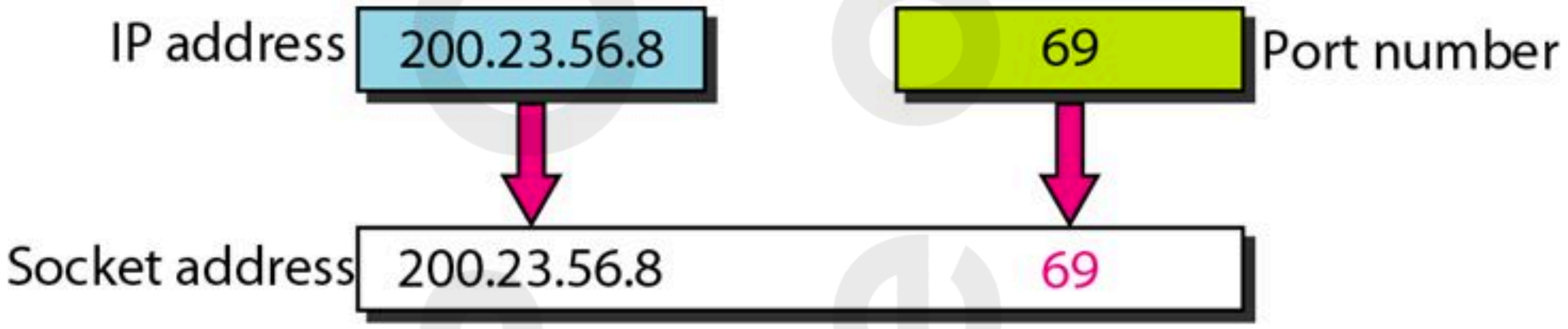
- **Internet Assigned Number Authority** has divided port numbers into three ranges:
    1. **Well-Known Ports** - Range from 0 to 1023; assigned and controlled by IANA.
    2. **Registered Ports** - Range from 1024 to 49151; not assigned or controlled by IANA; can be registered with IANA to prevent duplication.
    3. **Dynamic Ports** - Range from 49152 to 65535; neither controlled nor registered; used by any process; ephemeral ports.



Port Number	Transport Protocol	Service Name	RFC
20, 21	TCP	File Transfer Protocol (FTP)	RFC 959
22	TCP and UDP	Secure Shell (SSH)	RFC 4250-4256
23	TCP	Telnet	RFC 854
25	TCP	Simple Mail Transfer Protocol (SMTP)	RFC 5321
53	TCP and UDP	Domain Name Server (DNS)	RFC 1034-1035
67, 68	UDP	Dynamic Host Configuration Protocol (DHCP)	RFC 2131
69	UDP	Trivial File Transfer Protocol (TFTP)	RFC 1350
80	TCP	HyperText Transfer Protocol (HTTP)	RFC 2616
110	TCP	Post Office Protocol (POP3)	RFC 1939
119	TCP	Network News Transport Protocol (NNTP)	RFC 8977
123	UDP	Network Time Protocol (NTP)	RFC 5905
135-139	TCP and UDP	NetBIOS	RFC 1001-1002
143	TCP and UDP	Internet Message Access Protocol (IMAP4)	RFC 3501
161, 162	TCP and UDP	Simple Network Management Protocol (SNMP)	RFC 1901-1908, 3411-3418
179	TCP	Border Gateway Protocol (BGP)	RFC 4271
389	TCP and UDP	Lightweight Directory Access Protocol	RFC 4510
443	TCP and UDP	HTTP with Secure Sockets Layer (SSL)	RFC 2818
500	UDP	Internet Security Association and Key Management Protocol (ISAKMP) / Internet Key Exchange (IKE)	RFC 2408 - 2409
636	TCP and UDP	Lightweight Directory Access Protocol over TLS/SSL (LDAPS	RFC 4513
989/990	TCP	FTP over TLS/SSL	RFC 4217

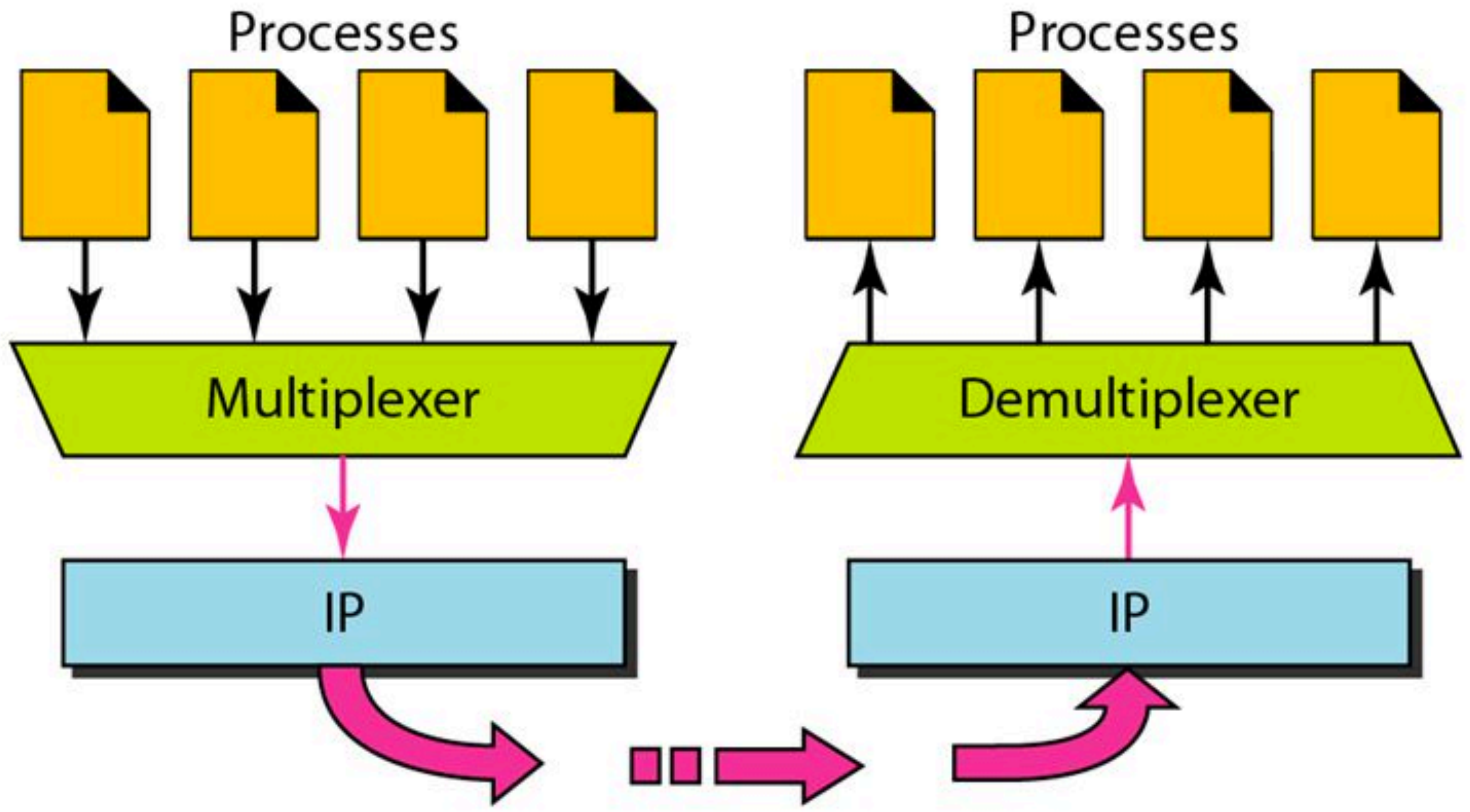
Socket Addresses

- Process-to-process delivery needs two identifiers: (1) **IP address** and (2) **port number** (collectively known as socket address).
- Client socket address defines client process as the server socket address defines server process uniquely.



Multiplexing and Demultiplexing

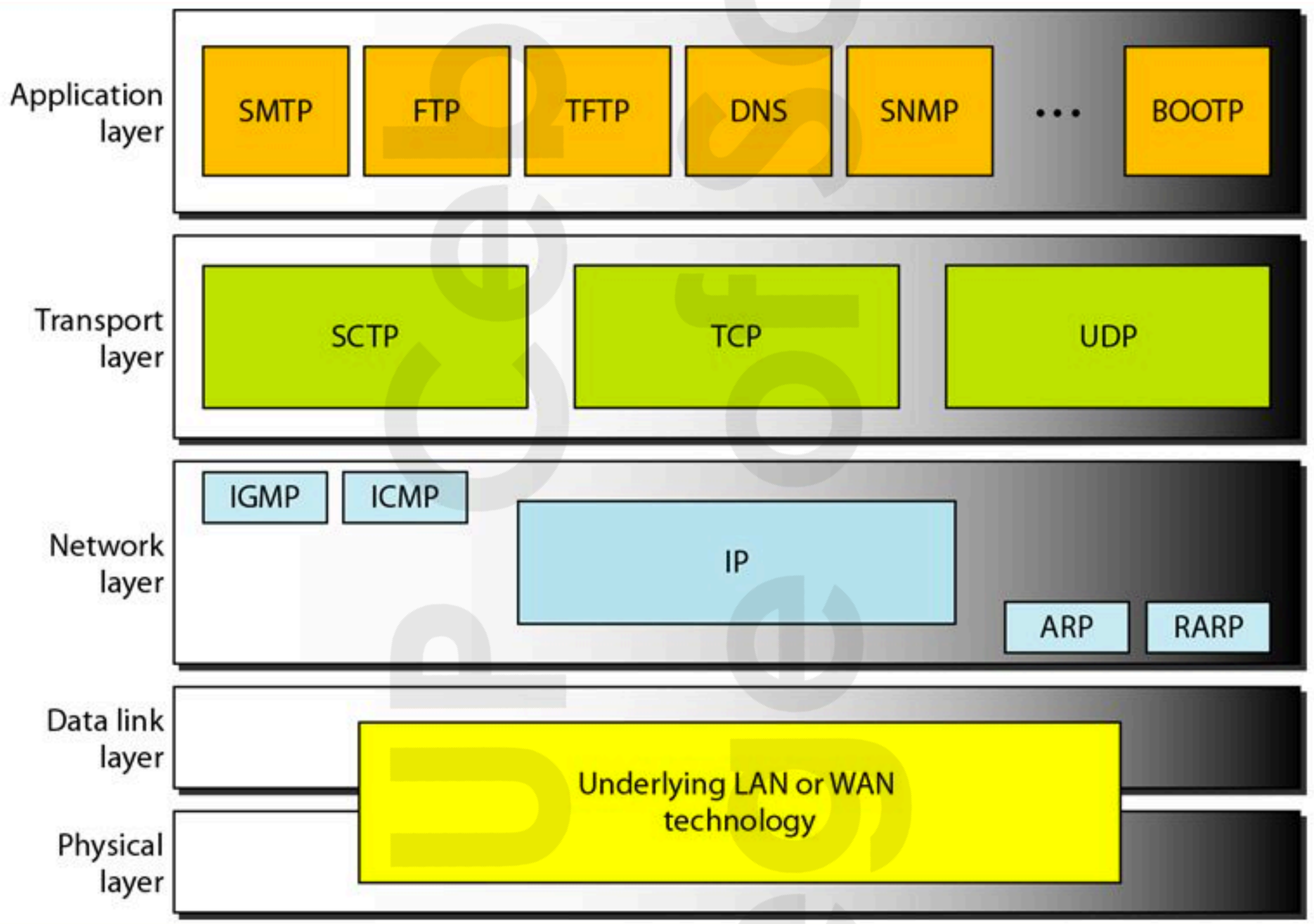
- **Multiplexing** - At sender site, there may be several processes that need to send packets but there is only one transport layer protocol at any time; many-to-one relationship and requires multiplexing. Protocol accepts messages from different processes, differentiated by assigned port numbers. After adding header, the transport layer passes packet to network layer.
- **Demultiplexing** - At receiver site, relationship is one-to-many and requires demultiplexing. Transport layer receives datagrams from network layer. After error checking and dropping of header, transport layer delivers each message to appropriate process based on the port number.



Connectionless Versus Connection-Oriented Service

- **Connectionless Service**
  - Packets are sent from one party to another with no need for connection establishment or connection release.
  - Packets are not numbered; maybe delayed or lost or may arrive out of sequence.
  - No acknowledgment.
  - UDP protocol.
- **Connection-Oriented Service**
  - Connection is first established between sender and receiver then data are transferred, at the end, connection is released.
  - TCP and SCTP protocols.

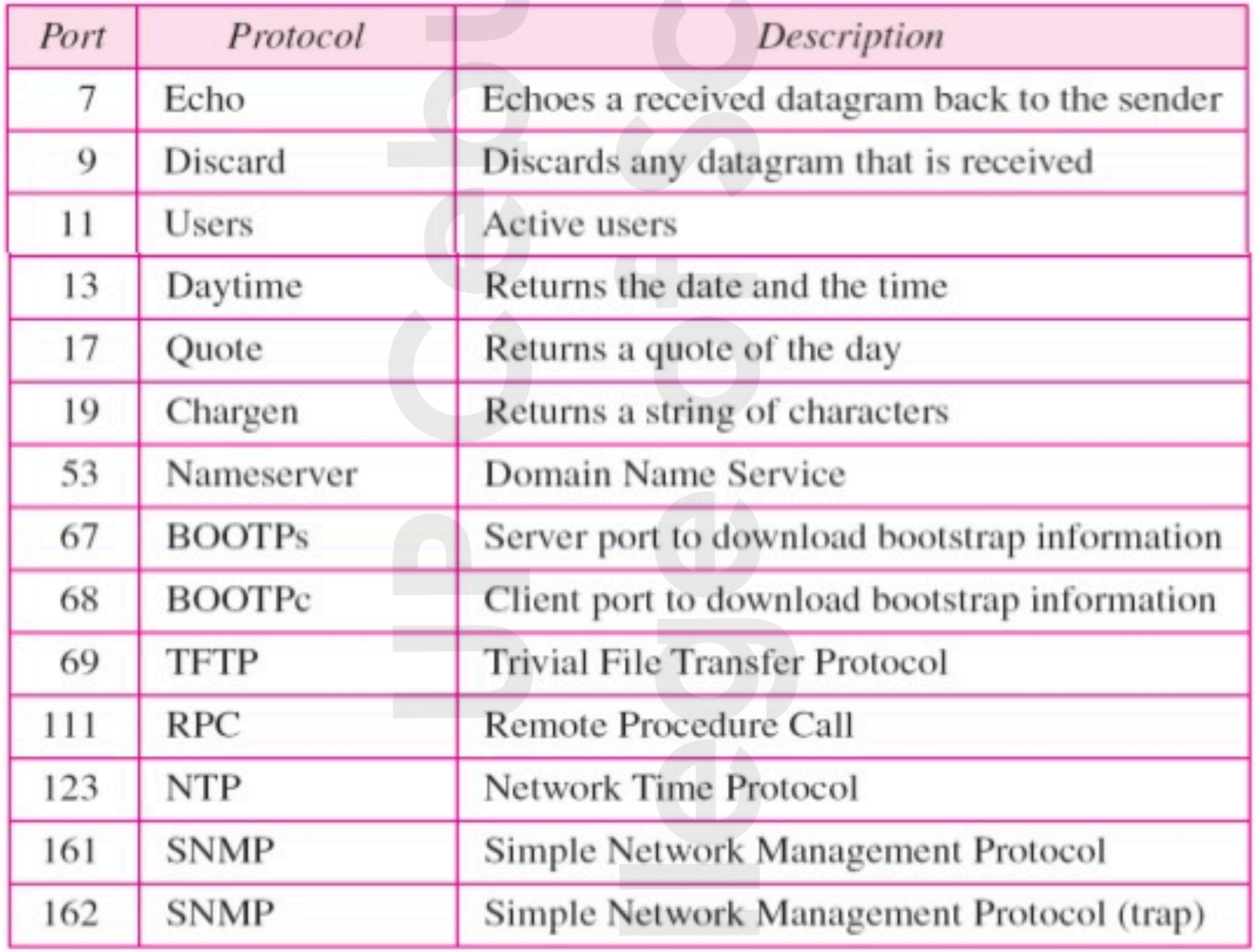
Three Protocols



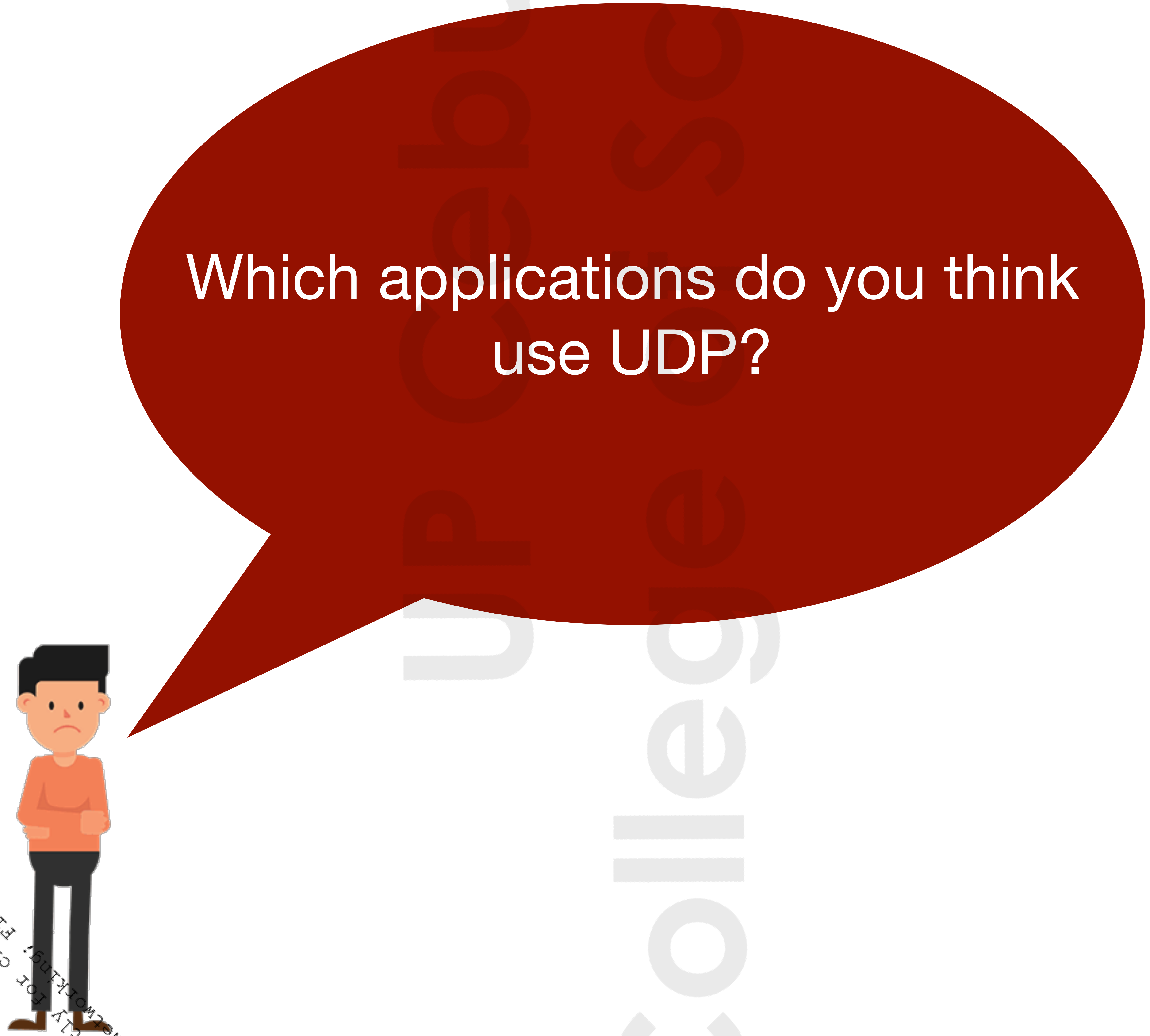
1. User Datagram Protocol (UDP)

- Connectionless, unreliable transport protocol.
- Performs very limited error checking.
- Question: If UDP is so powerless, why would a process want to use it?
- UDP is very simple protocol using a minimum overhead.
- Suitable for a process that wants to send a small message and does not care much about reliability.
- Takes much less interaction between sender and receiver than using TCP or SCTP.

Well-Known Ports for UDP



Port	*Protocol*	*Description*
7	Echo	Echoes a received datagram back to the sender
9	Discard	Discards any datagram that is received
11	Users	Active users
13	Daytime	Returns the date and the time
17	Quote	Returns a quote of the day
19	Chargen	Returns a string of characters
53	Nameserver	Domain Name Service
67	BOOTPs	Server port to download bootstrap information
68	BOOTPc	Client port to download bootstrap information
69	TFTP	Trivial File Transfer Protocol
111	RPC	Remote Procedure Call
123	NTP	Network Time Protocol
161	SNMP	Simple Network Management Protocol
162	SNMP	Simple Network Management Protocol (trap)



Which applications do you think use UDP?

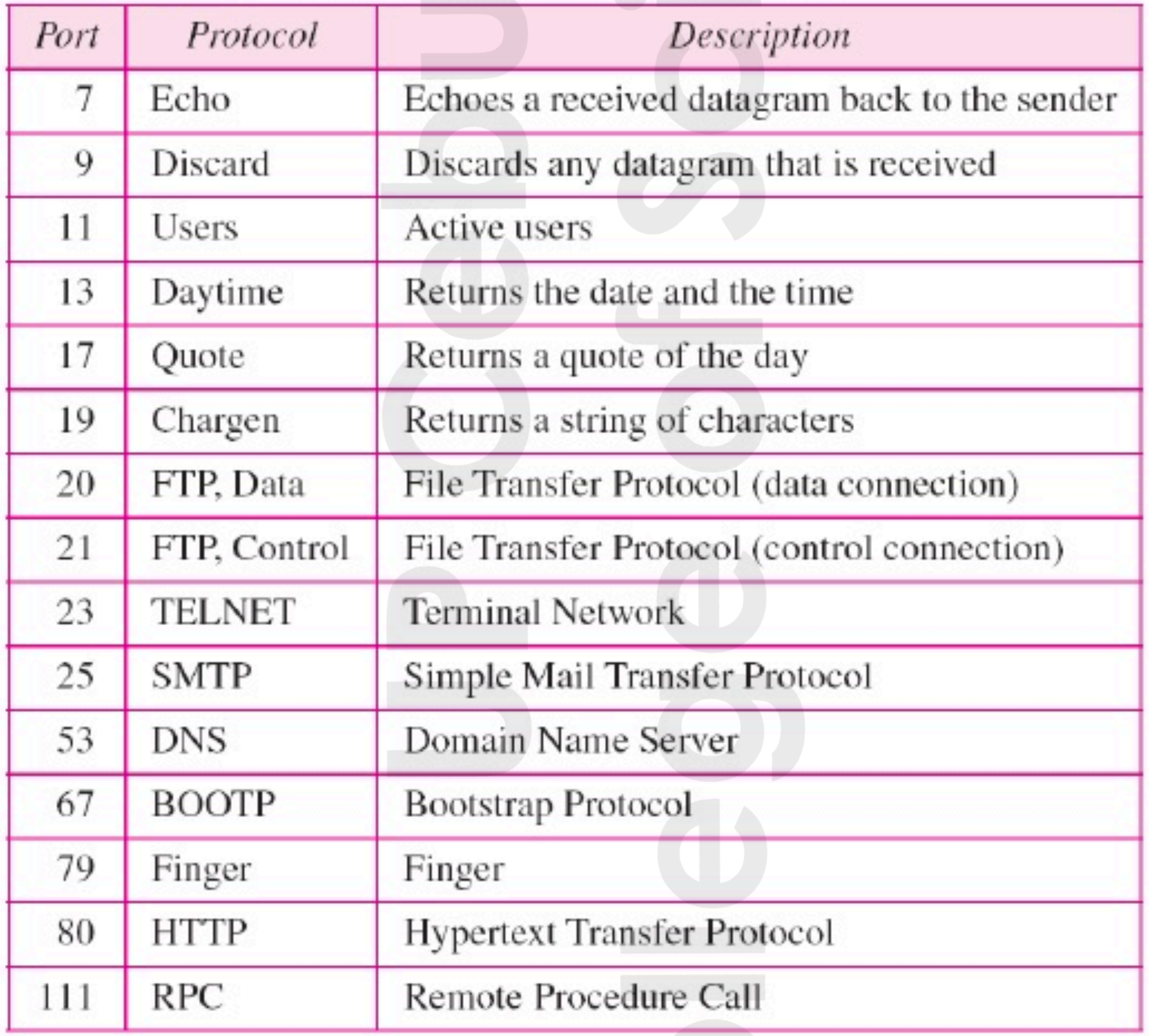
Applications Using UDP

- Multimedia Streaming
  - Retransmitting lost/corrupted packets is not worthwhile.
  - By the time the packet is retransmitted, it's too late.
  - For example, telephone calls, video conferencing, gaming.

2. Transmission Control Protocol (TCP)

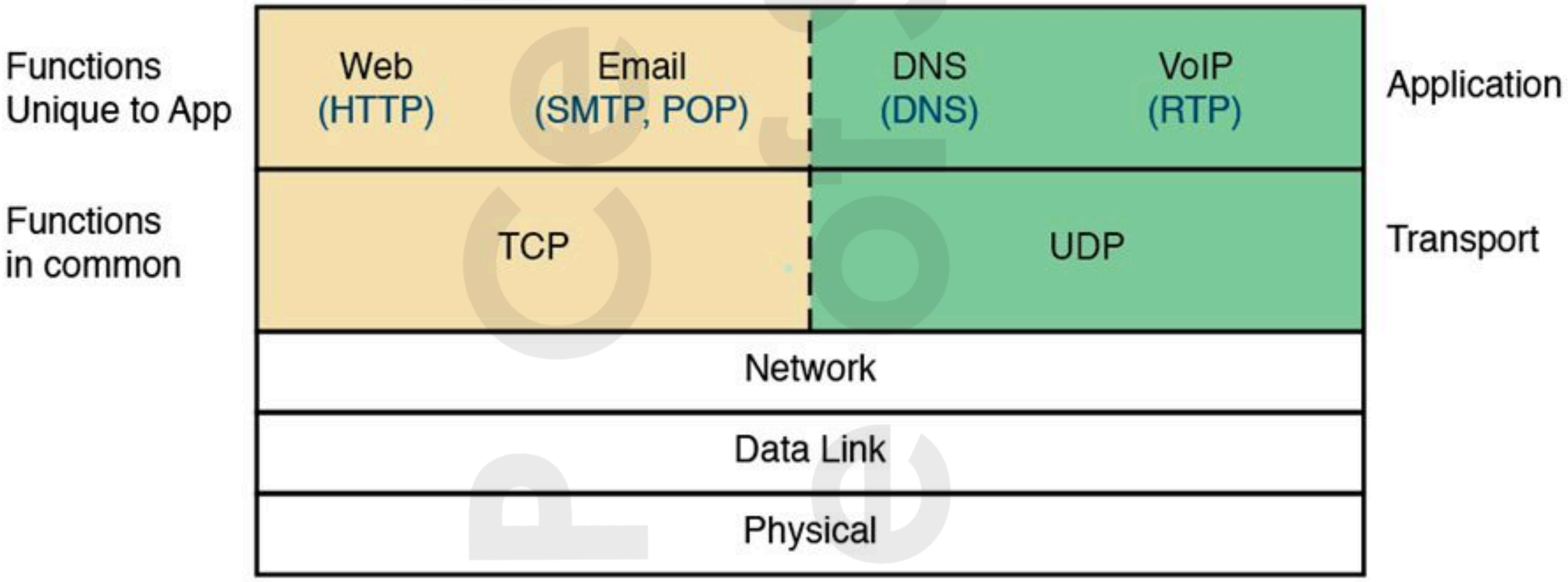
- Connection-oriented, reliable transport protocol.
- Creates a virtual connection between two TCPs to send data.
- Uses flow and error control mechanisms at the transport level.

Well-Known Ports for TCP



Port	Protocol	Description
7	Echo	Echoes a received datagram back to the sender
9	Discard	Discards any datagram that is received
11	Users	Active users
13	Daytime	Returns the date and the time
17	Quote	Returns a quote of the day
19	Chargen	Returns a string of characters
20	FTP, Data	File Transfer Protocol (data connection)
21	FTP, Control	File Transfer Protocol (control connection)
23	TELNET	Terminal Network
25	SMTP	Simple Mail Transfer Protocol
53	DNS	Domain Name Server
67	BOOTP	Bootstrap Protocol
79	Finger	Finger
80	HTTP	Hypertext Transfer Protocol
111	RPC	Remote Procedure Call

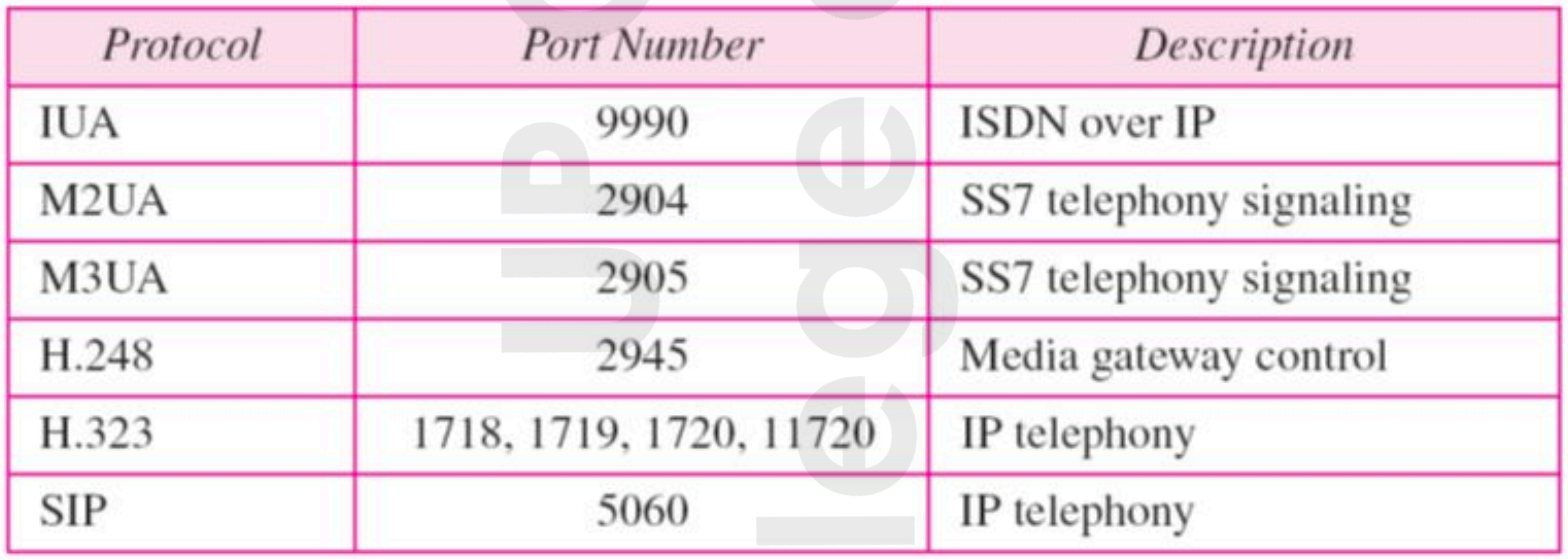
Applications Using TCP



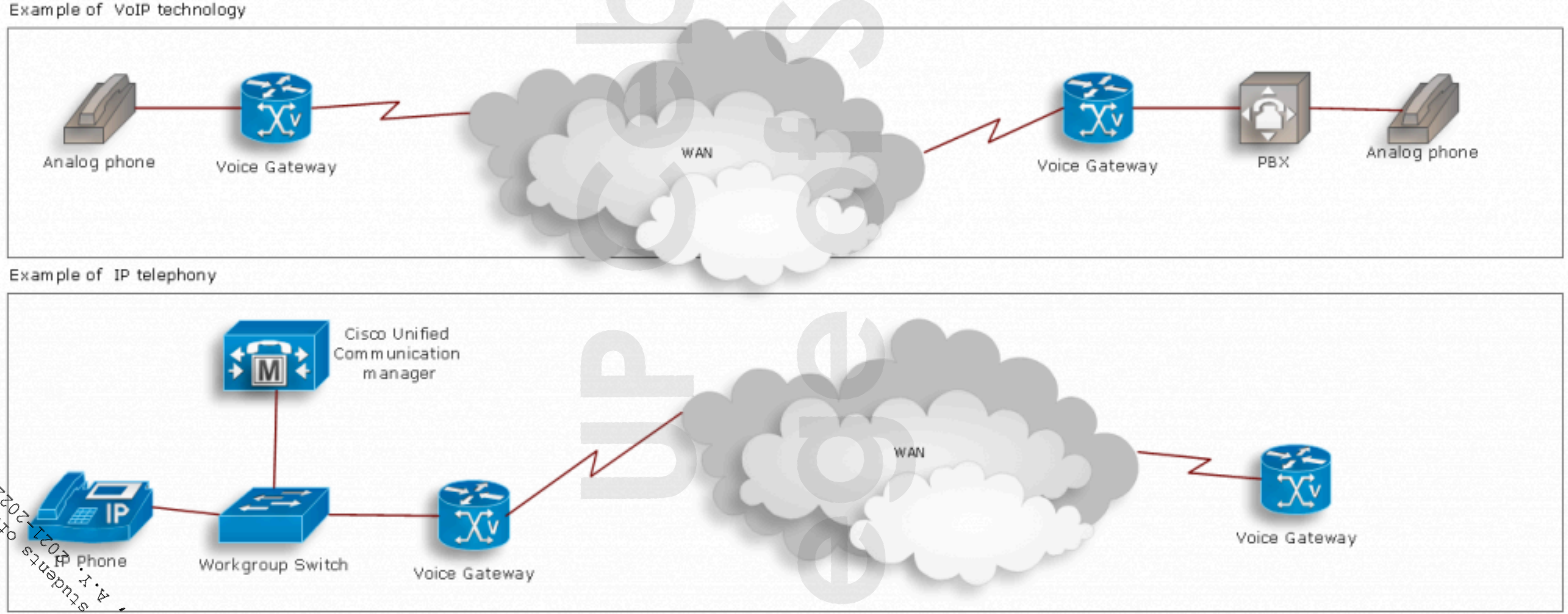
TCP	UDP
Keeps track of lost packets. Makes sure that lost packets are re-sent	Doesn't keep track of lost packets
Adds sequence numbers to packets and reorders any packets that arrive in the wrong order	Doesn't care about packet arrival order
Slower, because of all added additional functionality	Faster, because it lacks any extra features
Requires more computer resources, because the OS needs to keep track of ongoing communication sessions and manage them on a much deeper level	Requires less computer resources
Examples of programs and services that use TCP: - HTTP - HTTPS - FTP - Many computer games	Examples of programs and services that use UDP: - DNS - IP telephony - DHCP - Many computer games

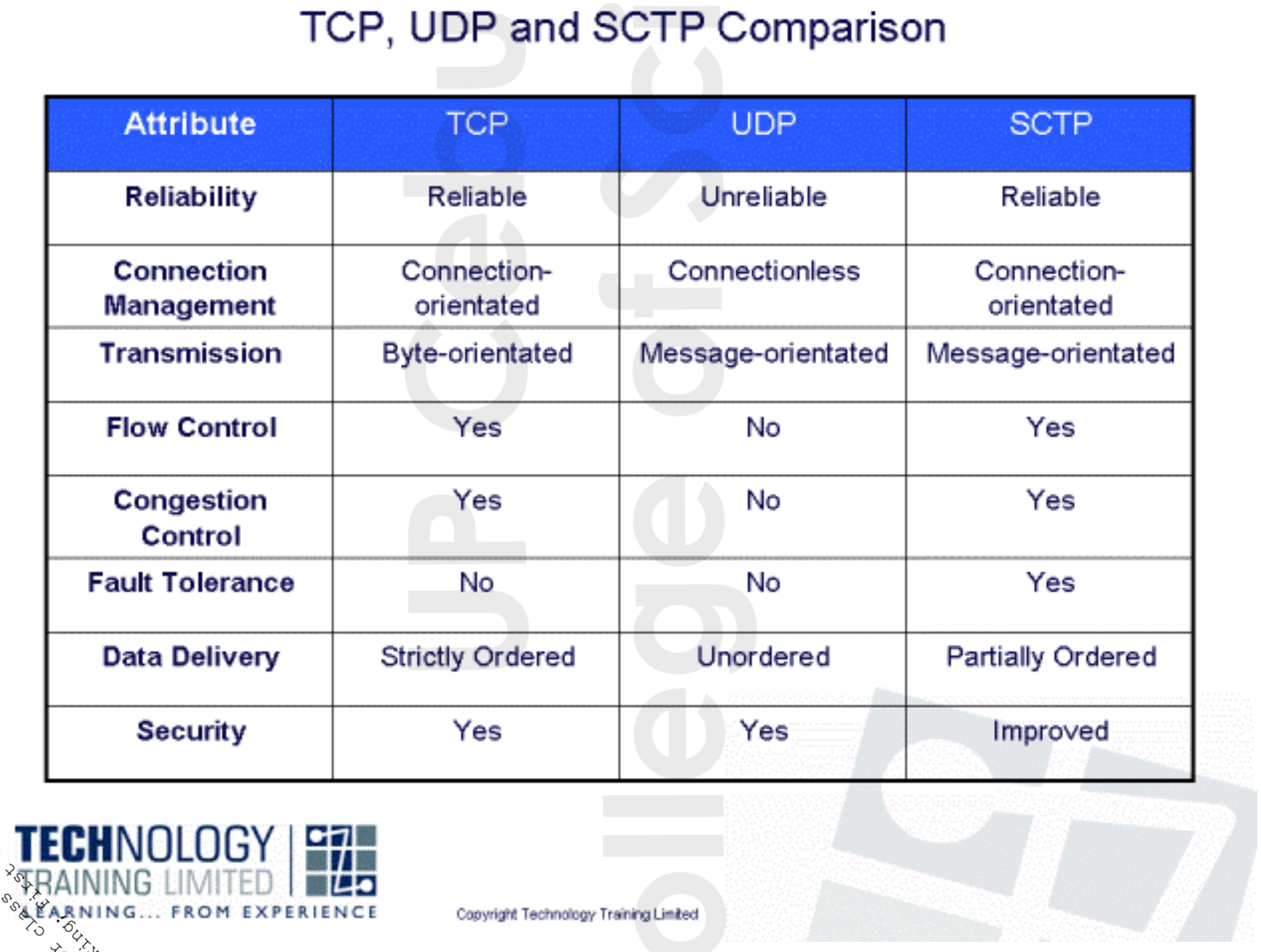
2. Stream Control Transmission Protocol (SCTP)

- New reliable, message-oriented transport layer protocol that combines the features of UDP and TCP.
- Mostly designed for Internet applications that have been recently introduced such as IUA (ISDN over IP), M2UA and M3UA (telephone signaling), H.248 (media gateway control), H.323 (IP telephony), and SIP (IP technology).



Protocol	*Port Number*	*Description*
IUA	9990	ISDN over IP
M2UA	2904	SS7 telephony signaling
M3UA	2905	SS7 telephony signaling
H.248	2945	Media gateway control
H.323	1718, 1719, 1720, 11720	IP telephony
SIP	5060	IP telephony


Example of VoIP technology
Analog phone
Voice Gateway
WAN
Voice Gateway
PBX
Analog phone
Example of IP telephony
Cisco Unified Communication manager
IP Phone
Workgroup Switch
Voice Gateway
WAN
Voice Gateway



TCP, UDP and SCTP Comparison

Attribute	TCP	UDP	SCTP
Reliability	Reliable	Unreliable	Reliable
Connection Management	Connection-orientated	Connectionless	Connection-orientated
Transmission	Byte-orientated	Message-orientated	Message-orientated
Flow Control	Yes	No	Yes
Congestion Control	Yes	No	Yes
Fault Tolerance	No	No	Yes
Data Delivery	Strictly Ordered	Unordered	Partially Ordered
Security	Yes	Yes	Improved