ASCII Character codes

	000	001	010	011	100	101	110	111
0000 0	NUL	DLE	SP	0	@	P	`	p
0001 1	SOH	DC1	!	1	A	Q	a	q
0010 2	STX	DC2	"	2	B	R	b	r
0011 3	ETX	DC3	#	3	C	S	c	s
0100 4	EOT	DC4	$	4	D	T	d	t
0101 5	ENQ	NAK	%	5	E	U	e	u
0110 6	ACK	SYN	&	6	F	V	f	v
0111 7	BEL	ETB	'	7	G	W	g	w
1000 8	BS	CAN	(	8	H	X	h	x
1001 9	HT	EM	)	9	I	Y	i	y
1010 A	LF	SUB	*	:	J	Z	j	z
1011 B	VT	ESC	+	;	K	[	k	{
1100 C	FF	FS	,	<	L	\	l	|
1101 D	CR	GS	-		M	]	m	}
1110 E	SO	RS	.	>	N	^	n	~
1111 F	SI	US	/	?	O	_	o	DEL

ASCII control codes (*more important ones in red*)

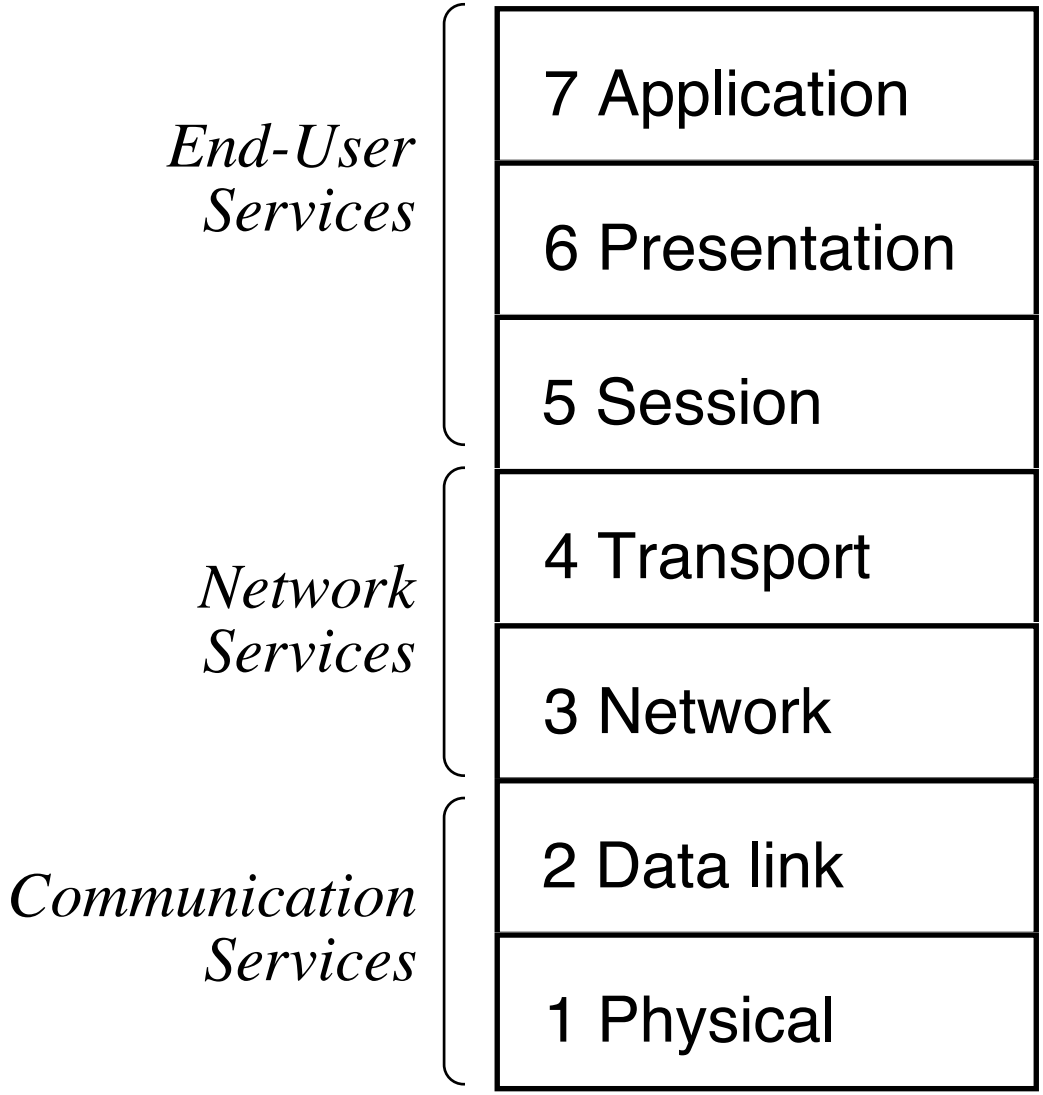
NUL	Null	All-zero "filler" character
SOH	Start of Header	*Indicates start of control information*
STX	Start of Text	*Separates heading and text in a message*
ETX	End of Text	*End of the text which started with STX (or final end, c.f. ETB)*
EOT	End of Transmission	*End of transmission with one or more Texts; OR "reset line"*
ENQ	Enquire	*Request for response, recipient should reply*
ACK	Acknowledge	*Positive acknowledgement of message ("OK to proceed")*
BEL	Bell	Activate a visible or audible alarm
BS	Backspace	Move cursor or print mechanism backward one position
HT	Horizontal Tab	Move cursor or print mechanism right to next Tab stop
LF	Line Feed	*Move cursor or print mechanism to same position on next line*
VT	Vertical Tab	Feed paper to next predefined line
FF	Form Feed	Advance to top of next page
CR	Carriage Return	*Move cursor or print mechanism to start of present line*
SO	Shift Out	Assign special meanings to all following characters
SI	Shift In	End of "Shift Out"; all characters revert to the normal meaning

DLE	Data Link Escape	*Assign a special meaning to the following character*
DC1	Device Control 1	*General device control; often "XON" to enable transmission*
DC2	Device Control 2	General device control
DC3	Device Control 3	*General device control; often "XOFF" to stop transmission*
DC4	Device Control 4	General device control
NAK	Negative Acknowledge	*Negative response, e.g. bad parity, or device busy*
SYN	Synchronous Idle	*Establish character synchronisation; also synch. filler code*
ETB	End Transmission Block	*End of a transmission (more yet to come, c.f. ETX)*
CAN	Cancel	Cancel (i.e. ignore) previous data in this message
EM	End of Medium	End of card deck, paper tape, cassette tape etc.
SUB	Substitute	Used to replace a bad or invalid character in message
ESC	Escape	Start of a special character sequence for device control
FS	File Separator	Separates files or other major data units
GS	Group Separator	Separates groups of records within a File
RS	Record Separator	Separator between records
US	Unit Separator	Separator between items in a record

Hierarchical Communication Models

- Communications systems are best handled with a hierarchical control structure, much like a hierarchy or nest of subroutines.
- Some systems, such as TCP/IP, have a very flexible layering, with layers reaching into adjoining layers or even skipping over an adjoining layer.
- We will start with the OSI model (Open Systems Interconnection) which has a very strict seven layer structure — each layer communicates only with its adjoining layers.
- Most later work will be with TCP/IP.
- A half serious comment is that TCP/IP is the way you *do* communications, but OSI is the way you *teach* it!

The OSI model

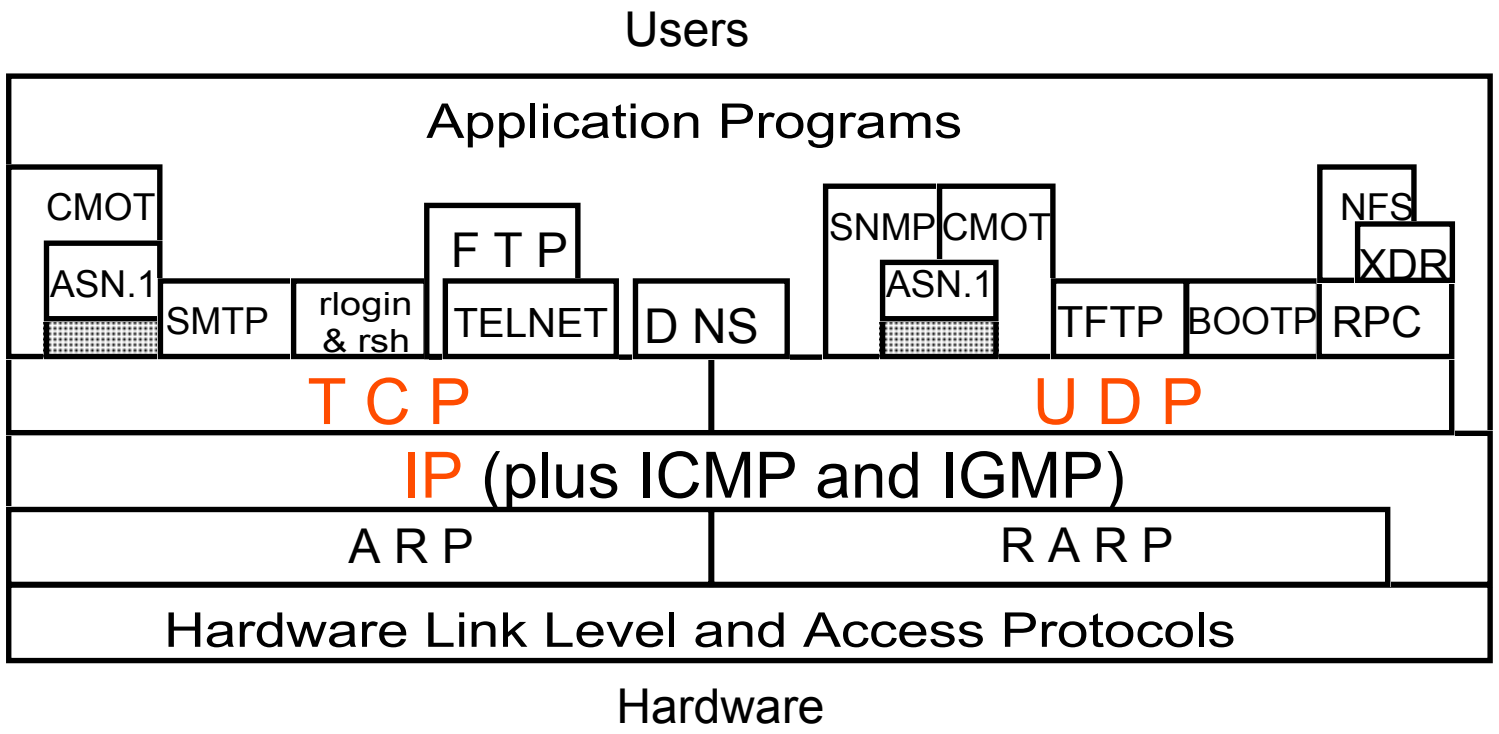


Levels 5 – 7 are more the concern of MSIS and the final use of communications.

We concentrate on levels 1 to 4, and specially levels 2 to 4.

The details of level 1 are more the concern of Electrical Engineering.

The TCP/IP Protocol suite (to be discussed later)



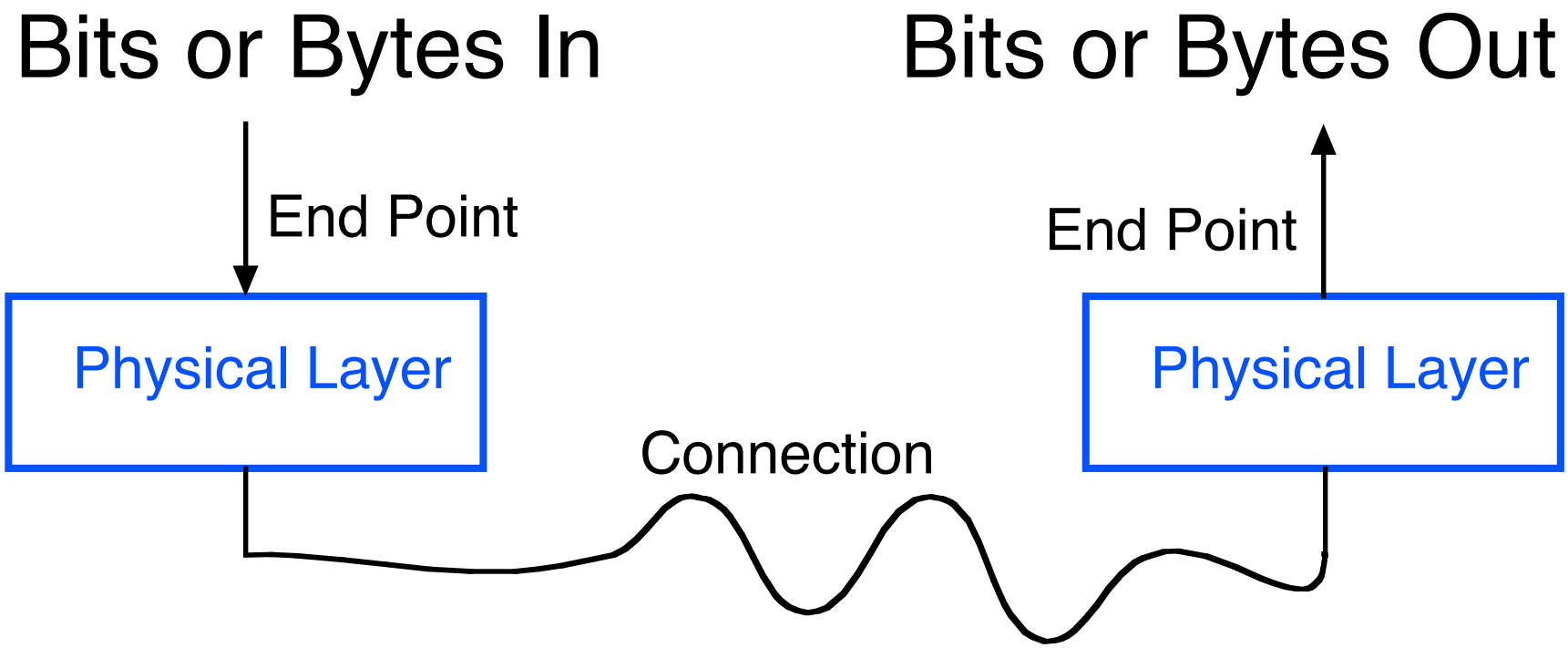
TELNET	Terminal emulator	FTP	File Transfer Protocol
SMTP	Simple Mail Transfer Protocol	CMOT	CMIP over TCP
DNS	Domain Name Server	SNMP	Simple Network Maintenance Protocol
TFTP	Trivial File Transfer Protocol	BOOTP	Boot Protocol
RPC	Remote Procedure Call	NFS	Network File System

Given

- A “connection” (wire, optical cable, radio, etc) between two “end-points”

Problem

- Convey a stream of bits or bytes from one end to the other.



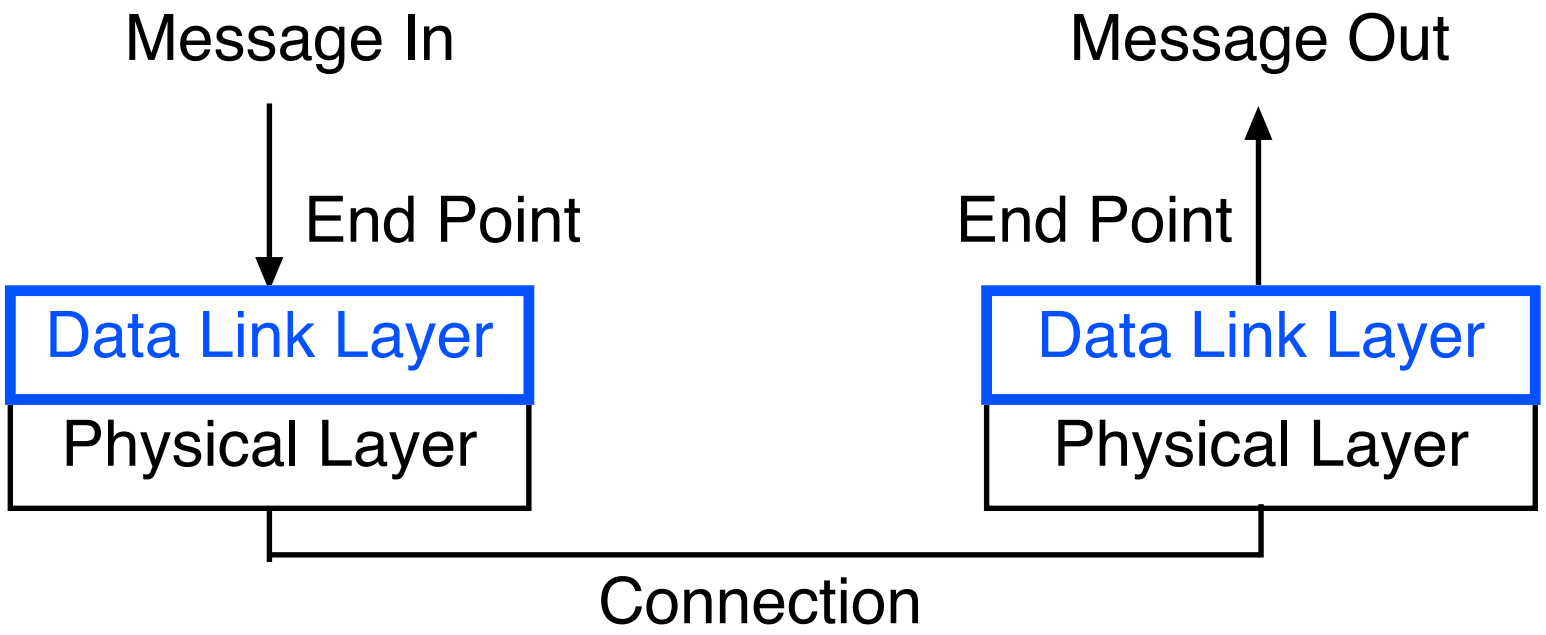
- Problems of data coding, transmission, clock recovery … …

Given

- A *Physical Layer* able to transmit bits or bytes end to end

Problem

- Convey recognised sequences (messages or frames) of bytes



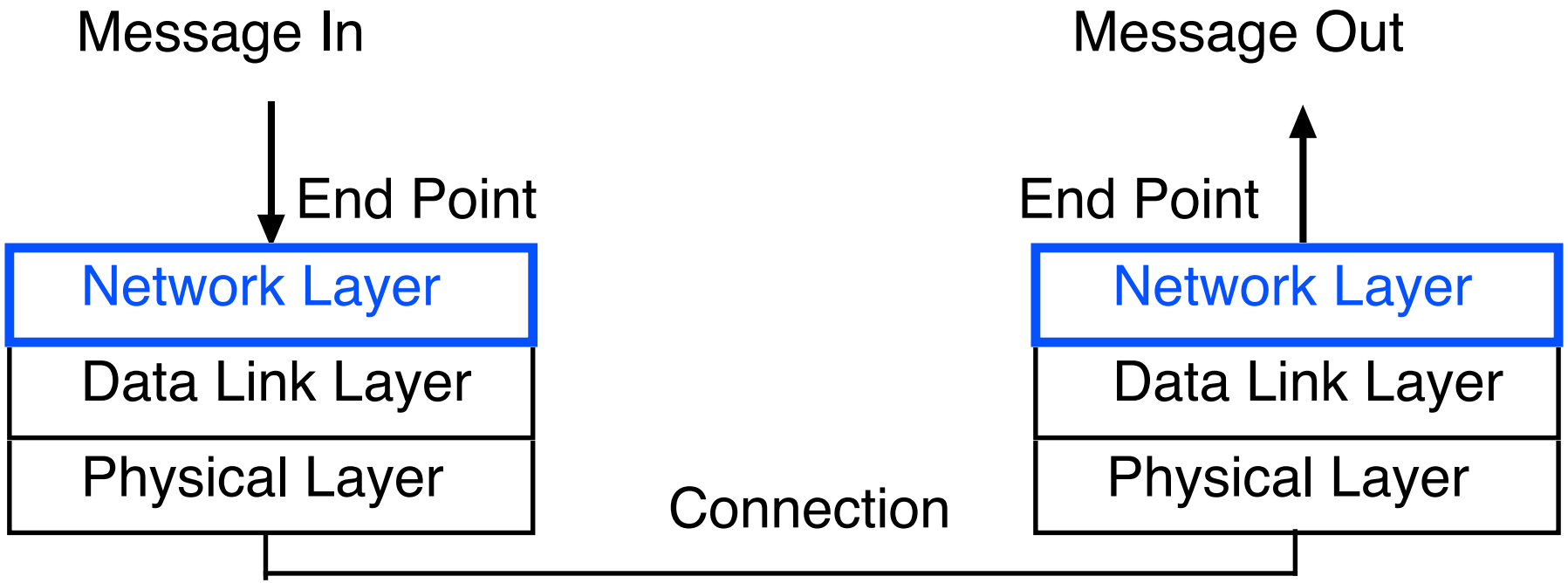
- Problems of assembling bits into bytes, indicating start and finish of message, handling lost or corrupted data

Given

- A *Data Link Layer* for reliable transmission between end-points

Problem

- Communicate between any pair of a mesh of "nodes"



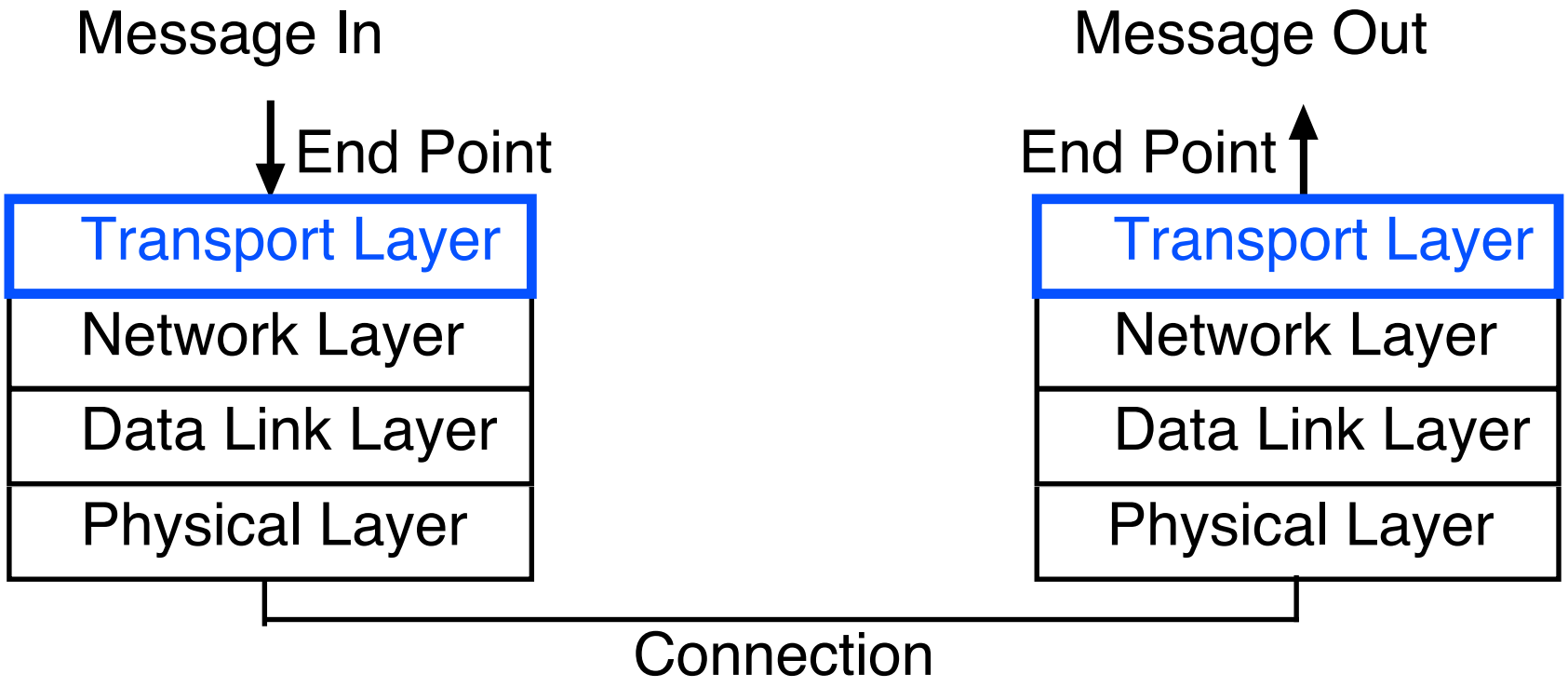
- Find paths ("routes") between nodes, recover from broken paths

Given

- A *Network Layer* to send between network nodes

Problem

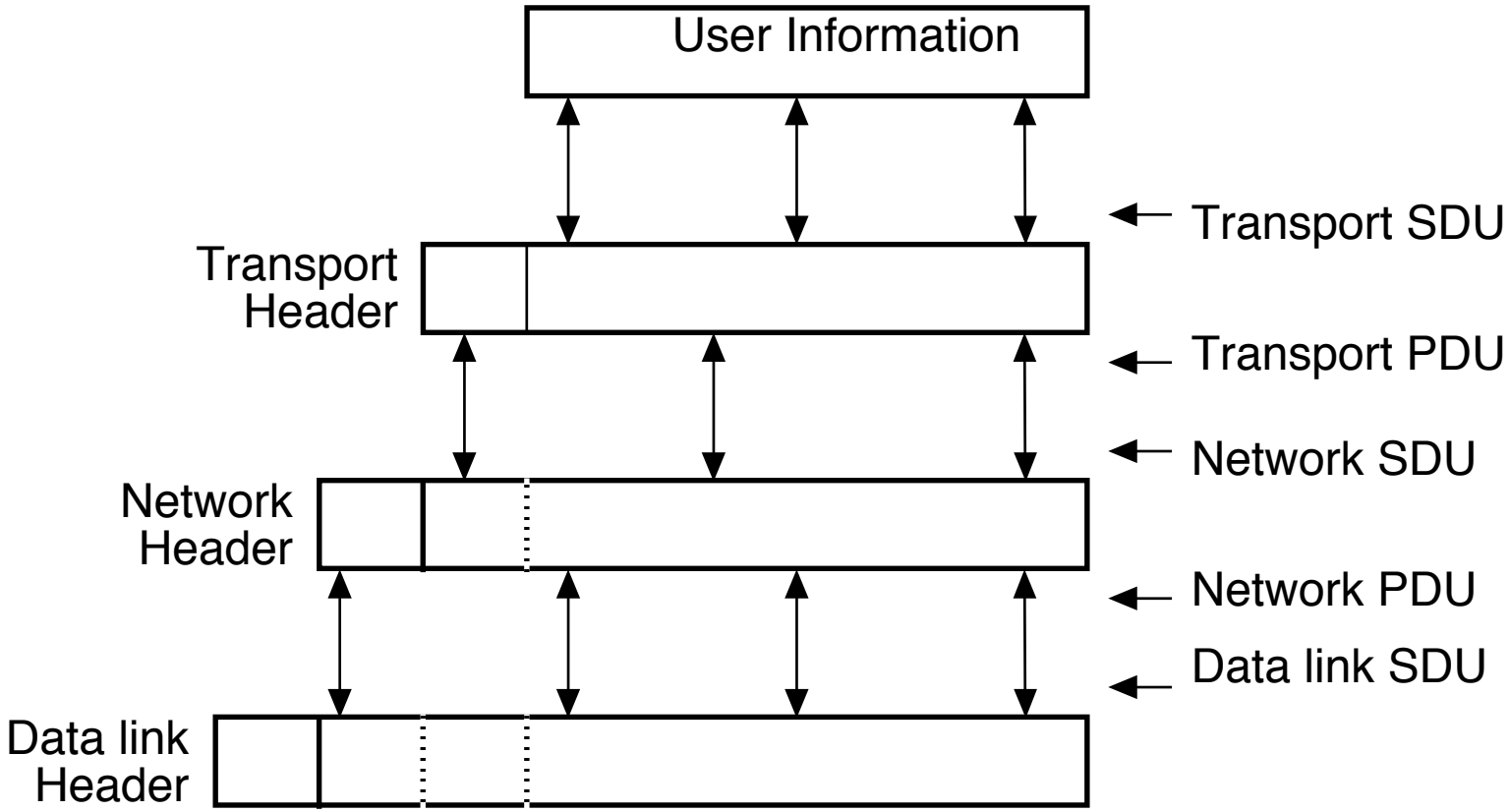
- Guarantee reliable end-to-end transmission through network



- Handle lost messages, divide large messages into small

Message Encapsulation

- As messages go down the protocol stack, each received Service Data Unit (SDU) has address and control information added, to become the outgoing Protocol Data Unit (PDU) (and vice versa when going up the stack).
- Some messages may be segmented, one SDU into several PDUs



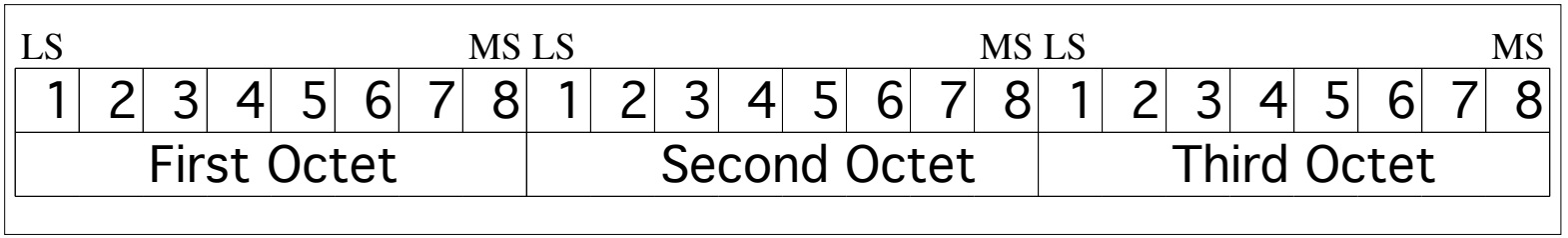
Data Communications Transfer rates

- Data sent over a communication link is converted into a byte stream.
- How long does it takes to transmit data or, what is the data rate as seen by the user?
- A fast, high-speed data link is no guarantee that users will see data transferred at that rate, or at anything approaching it.

  - Students are often confused by the word “rate”.
  - Rate can be defined (my dictionary) as “a fixed ratio between two things”.
  - Here it usually means how often something happens — as “a data rate of 1 megabits per second” means that 1 million bits are handled or delivered in each second.
  - And most important, a computer “1 megabyte” usually means 1,048,576 bytes, whereas a data communications “1 megabit” always means exactly 1,000,000 bits.

Bits, Bytes and Octets

- Many communications standards and documents call the 8-bit data unit an “octet”, rather than a “byte”. There are two other matters which may cause confusion
- As octets are usually transmitted least-significant bit first,documents often write the bits in transmission order, with the least significant bit on the left and most significant bit on the right – the reverse of the usual numeric conventions on computers.
- The bits are often numbered 1—8 rather than “computerese” 0—7.



Data Transmission

1. The sending computer prepares a block of bytes, of an appropriate length and delivers the block to the data communications software.
2. The communications software divides the user data into sizes more appropriate to communications.
   Or, many small pieces of user data may assembled into one longer entity.
3. The "packet" from step 2 is placed in an "envelope" for transmission, with a header including addresses, message type, length and control, and a trailer with check information.
   (In a multi-layer protocol, which is usual, steps 2 and 3 may be repeated several times, leaving the user data buried deeply in a nest of headers and trailers. At each stage the packet has a header, and trailer, enclosing a message body.)
4. The final packet, including its header and trailer are sent over the data link.

5. The receiver checks the whole packet for validity and correctness, and extracts the message body.
   The body may be assembled into part of a larger message or broken into smaller ones as needed.
   The resultant message or messages may be sent on to the final user or passed to another communications layer for further processing.
6. A reply or acknowledgement may be prepared to send back to the original sender.
   This is just another communications message and may be combined with others on the return path.
   It has all of the encapsulation and transmission times given above.

Each one of these steps takes a certain amount of time; these times must be combined to obtain the total effective time to process the message. Some texts develop complicated formulæ to describe these times, but it is better to show how to derive them in the general case and then apply the rules to a particular example.

Automatic Repeat Request (ARQ).

This is the simplest case where the sender sends one message and waits for a reply, either "correct" or "error – send again" before sending the next message. Assume that

- a user message of U bytes (octets) is enclosed in an envelope of $(H+T)$ octets for sending over a data link at R bits/second.
- the reply (positive or negative acknowledgement) is A octets, again enclosed in an envelope of $(H+T)$ octets.
- the distance between sender and receiver is D, and information travels at a velocity V.

Neglecting any processing delays, the time to get a reply and before the next message may be sent is

This page is duplicated - IGNORE version with "to send *A* bytes"

Time to send packet from sender $= \frac{8 \times (H + U + T)}{R}$

Time for packet to travel to receiver $= \frac{D}{V}$

Time for receiver to send reply $= \frac{8 \times (H + A + T)}{R}$

Time for reply to return to sender $= \frac{D}{V}$

Total time before next message may be sent $= \frac{8 \times (2H + A + U + 2T)}{R} + \frac{2D}{V}$

Effective or visible user data rate to send *U* bytes $= \frac{U}{\frac{8 \times (2H + A + U + 2T)}{R} + \frac{2D}{V}}$

Message generation time $= \frac{8 \times (2H + A + U + 2T)}{R}$

Link delay or link latency $= \frac{2D}{V}$

There are two components here.

- The message generation time is proportional to the length of the total data transmitted (user + reply), including overheads and is inversely proportional to the bit-rate on the link. For long messages it contributes a time $^{(U+A)}/_{R}$. which is clearly reduced by choosing a higher link bit rate R. For very short user messages the overheads $2(H+T)$ may become important.

  - ◇ Choosing a "faster link" affects only this time, allowing the data to be put into the link at more bits per second.
  - ◇ A faster link NEVER affects the transmission velocity or link latency; this something which we just have to live with.

- The transmission delay or link latency is dependent only on the distance or link length D.
  The transmission velocity V is almost always
  300,000,000 m/s (3×10^8 m/s,the speed of light c) for radio.
  200,000,000 m/s (2×10^8 m/s,about $^2/_3c$) for cables
  (either copper or optical)

  ◇ For transoceanic links the latency can be a very important value indeed.

Example (reply message is 30 octets plus envelope)

User data size U = 1000 bytes = 8000 bits

Header/trailer overhead (T+H) = 30 octet = 240 bits

Reply message size = 30 octet = 240 bits

End to end cable length D = 1000 metres

signalling data rate R = 10 Mbit/s (10^7 bit/s)

Signal velocity in cable V = 2×10^8 m/s

Then

end-to-end latency = 5 μs

time to send message = $8\times1030/10^7$ = 824 μs

time to send reply = $8\times60/10^7$ = 48.0 μs

Total time to send 1000 user bytes

(send + outward + reply + reverse) = 882.0 μs,

Effective user data rate = 1.134 byte/μs

Compare with naive prediction $10^7/_8$ = 1.25 byte/μs ($\approx$ 9.3% reduction)

This example is for a relatively short and (by modern standards) slow link with a moderately large packet.

A longer link, faster signalling rate and smaller packet can give a marked reduction in performance compared with the raw link speed, for ARQ protocols.

- Extreme precision is seldom needed or even appropriate in these examples. Packet or data sizes vary widely and cable velocities are seldom known to better than ±1% anyway.
- What is important is that you can give a good estimate of the effects of latency, or packet overheads, etc

Systems with overlapped transmission

- Send-and-wait signalling systems or protocols are unsuitable for high speed transmission with high latency or large distance.
- For these it is usual to send continuously, with continuous packets.
- Replies will eventually come back to say that a packet has been received, but each packet must be buffered at the sender until it is acknowledged.
- The data rate is now close to R×U/(U+T+H), but we need to know how much to buffer.

Example.

A transoceanic fibre-optic cable (D = 10,000 km) transmits ATM cells (U = 48, H = 5, T = 0) at 620 Mb/s.
How much user data must retained at the sender?

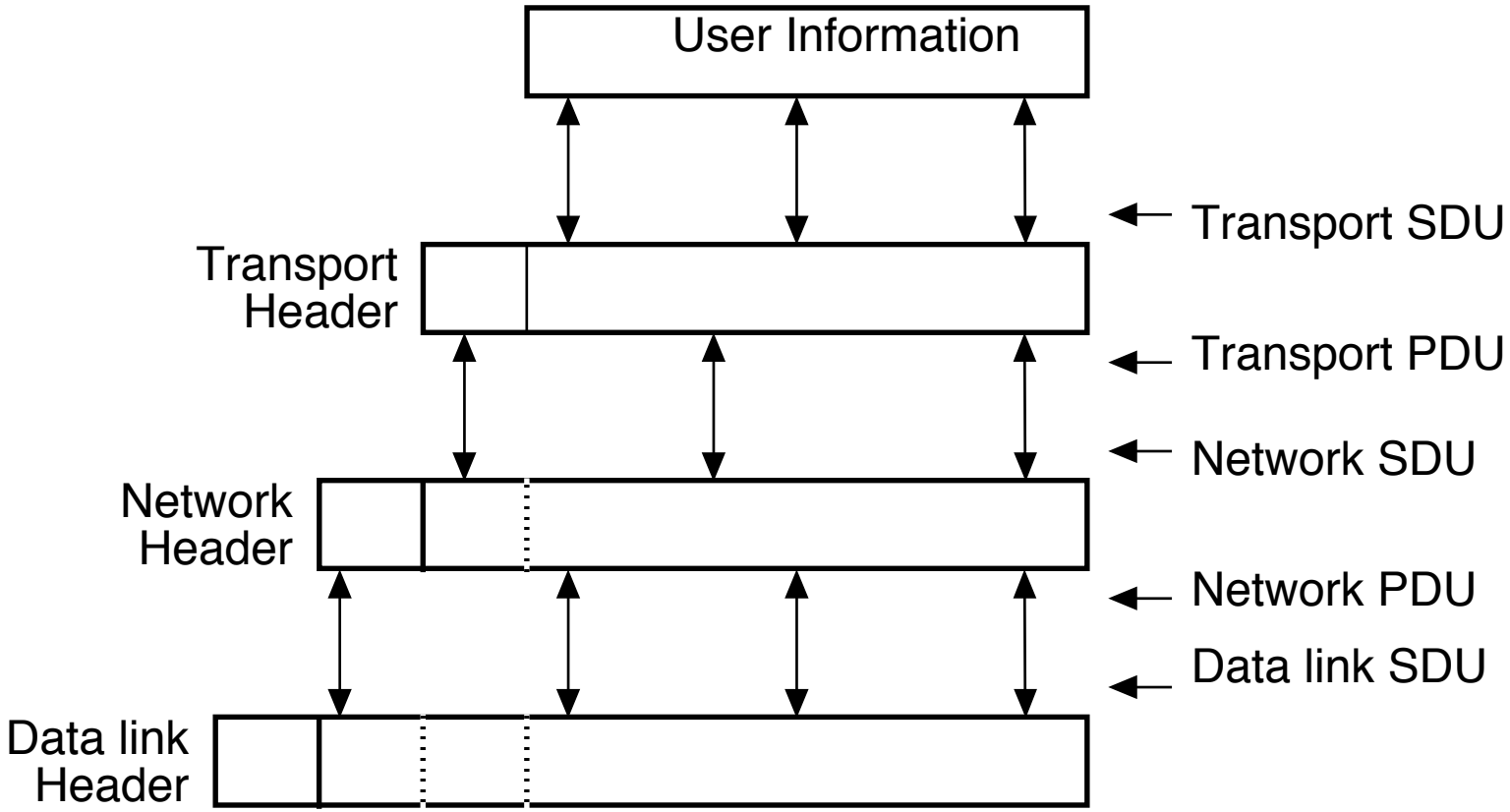
- The cable velocity is 200,000 km/s;
  with a round-trip distance of 20,000 km, the round-trip delay is 0.1 s.
- The raw data rate on the cable is $\frac{620}{8} \times \frac{48}{53}$ octet/μs = 70.2 octet/μs.
- In 0.1 s, we send 7.02 M octet.
  This much data must be buffered or held at the sender to allow the packet to be resent in the event of a packet error.

Consider the Network Layer

- The sending Network Layer receives data from its Transport Layer and passes it down to its Data Link Layer for transmission.
- The receiving Network Layer receives data from its Data Link Layer and passes it up to its Transport Layer.
- The Network Layer is written to exchange messages with the other Network Layer (its “peer” layer), without worrying how that message exchange occurs.
- Provided that it presents a standard interface to the Transport Layer (above) and to the DataLink Layer (below) and performs its defined functions, the Network Layer is independent of the layers above and below, and even of its peer layer.

Message Encapsulation

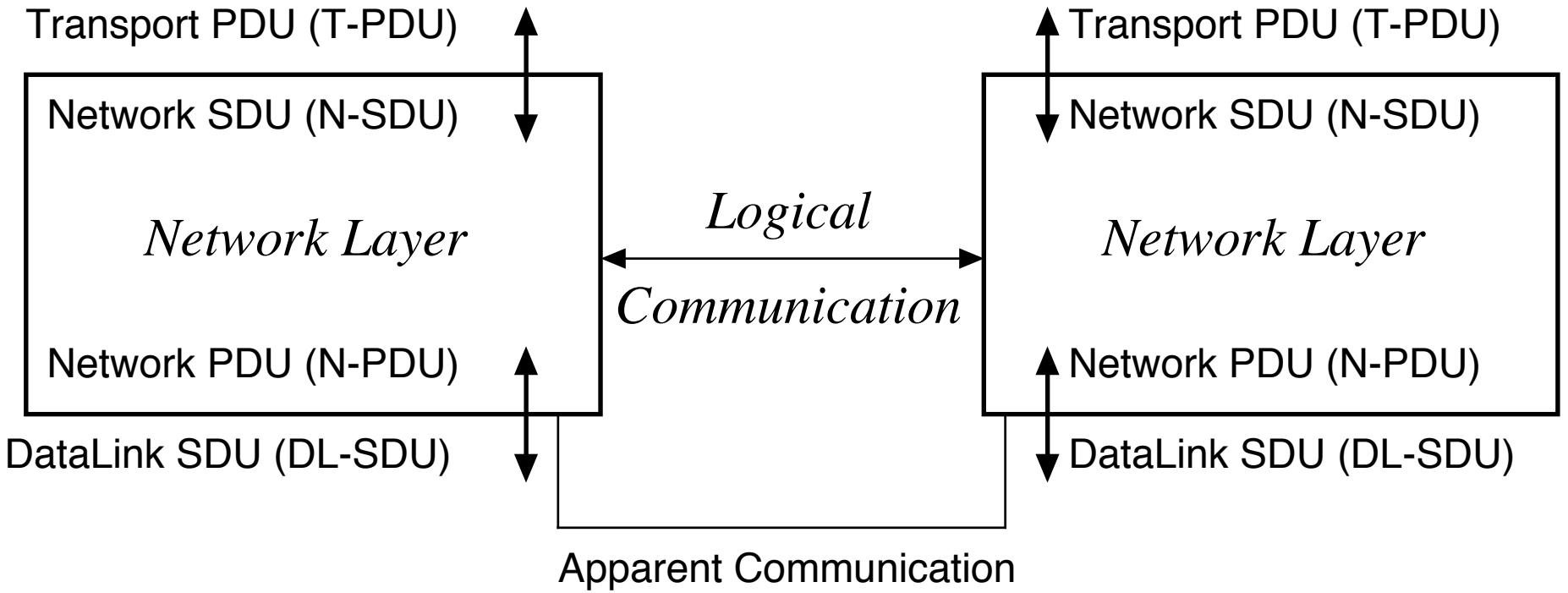
- As messages go down the protocol stack, each received Service Data Unit (SDU) has address and control information added, to become the outgoing Protocol Data Unit (PDU) (and vice versa when going up the stack).
- Some messages may be segmented, one SDU into several PDUs



Inter-layer Interface

- A given layer (layer *n*) exchanges *Service Data Units* (SDUs) with the layer above it (layer *n*+1).
- Layer *n* exchanges *Protocol Data Units* (PDUs) with the layer below it (layer *n*–1).
- Layer *n* assumes that it is exchanging PDUs with its corresponding peer layer *n* at the other end.
- At the boundary between layers the same message is regarded as a PDU by layer *n* and a SDU by layer *n*–1.

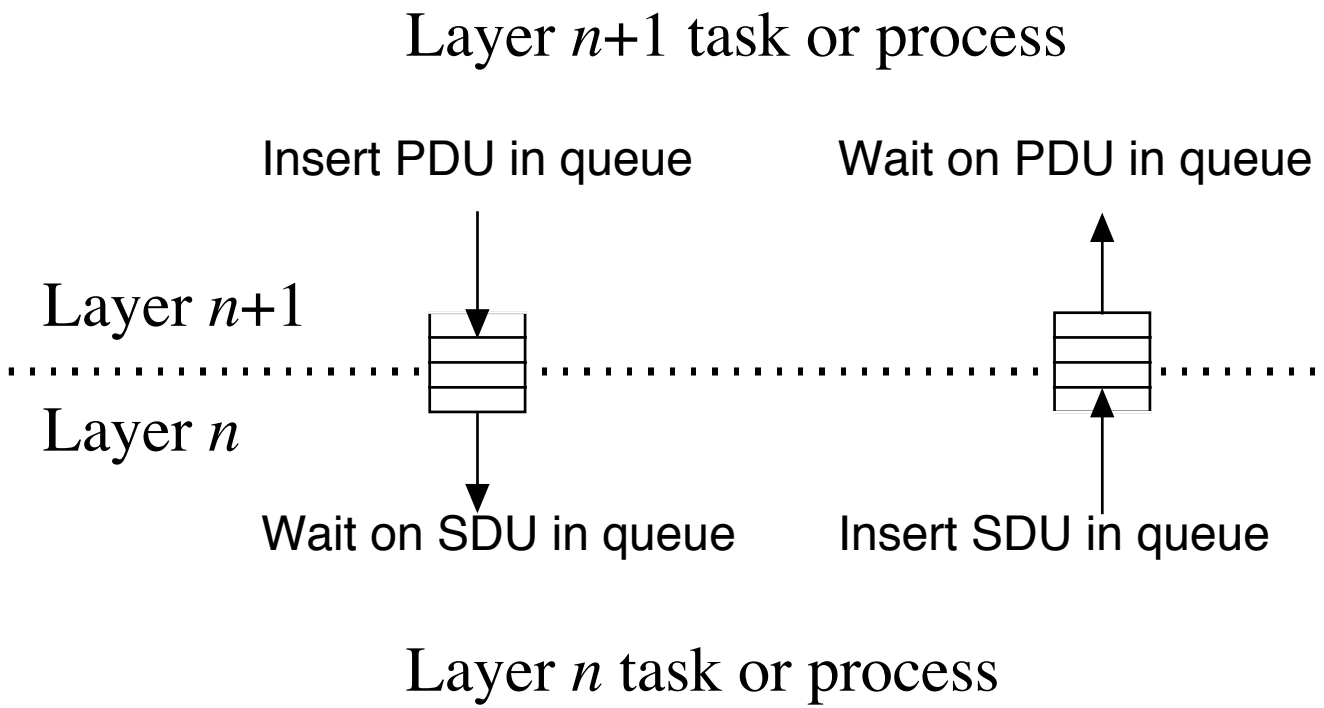
• At the Network Layer the data units and communication paths are —



• Note that the *same* message is a
Protocol Data Unit (PDU) above the layer boundary and a
Service Data Unit (SDU) below the boundary.

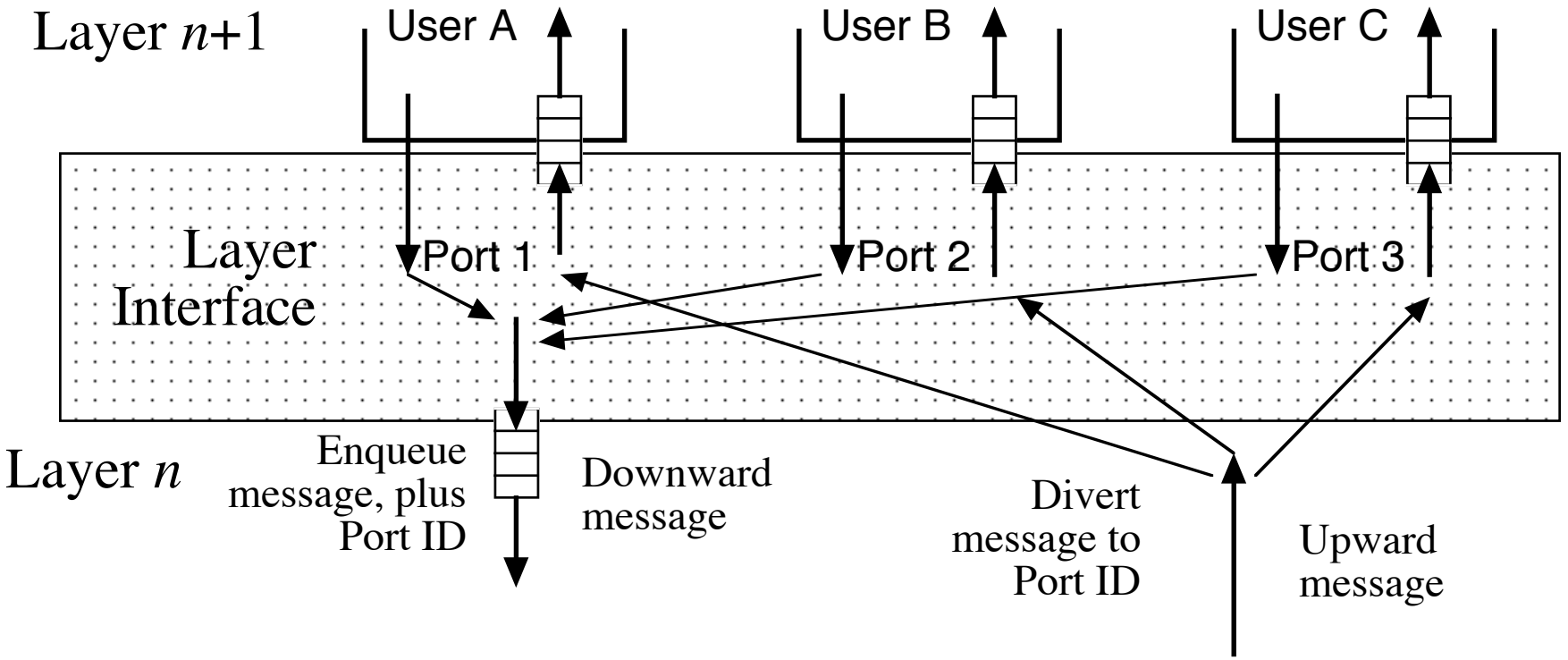
Inter-layer Interface (contd)

- The layers can be implemented as subroutines, but this gives an inflexible, synchronous system with little or no overlap of user functions and data transfer.
- It is better to implement layers as independent tasks or processes, communicating with neighbours through queues.



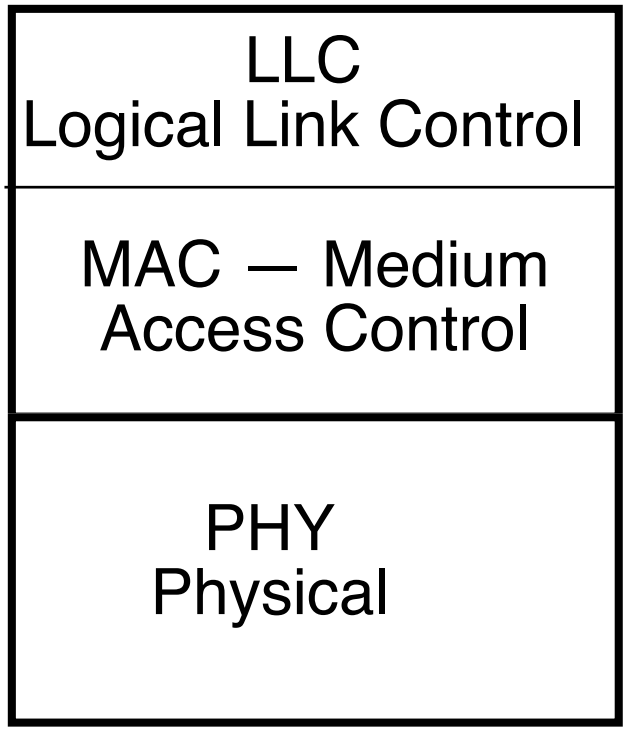
Multiple users and access points or ports

- There are often several users above a level, to allow multiple users of a communication path.
- Communication is through "access points" or "ports", each identified by an address or identifier. The address must be added to the message in the lower level.



Special Data link and Physical Layers

- The OSI model was designed for point-to-point links, carrying only one type of traffic.
- For Local Area Networks (LANs) and many other newer networks, layers 1 and 2 are modified, by splitting the Data link Layer into two.



LLC Provides a standard interface to the MAC, including multiplexing

MAC framing/deframing data units, error checking, coordinates access to medium (physical protocol)

PHY Dependent on network type(MAC and PHY operate as a unit)

Connectionless vs Connection-Oriented Services

- Modern communication almost always uses small bundles of data called "packets" (100 – 10,000 bytes), which allow links to be shared between many competing users.
- *Connectionless* services send packets (or datagrams) into the network, with no prior notice. Most LAN and Internet services (IP) are connectionless. Each packet must have a complete end-point address.
- *Connection-oriented* services first establish a "Virtual Circuit" from the source to the destination end-points. Packets have only the V-C number; tables in intermediate nodes maintain the connections for the virtual circuit. The source and destination may then use the V-C much as though there was a physical link.