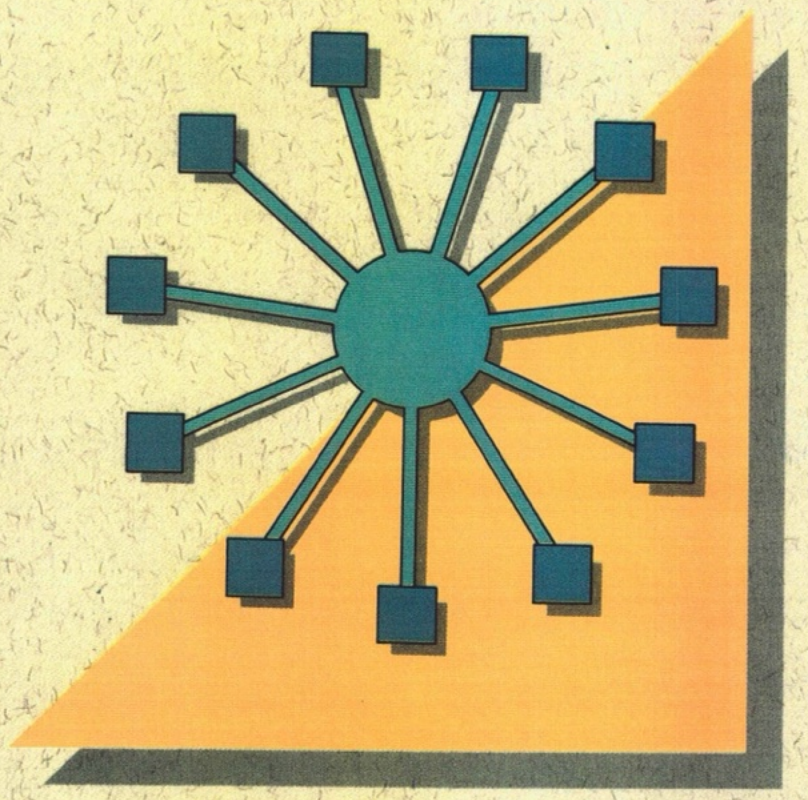
DATA NETWORKS

SECOND EDITION



Dimitri Bertsekas/Robert Gallager

IPR2025-00756, OnePlus Ex. 1013
Page 1 of 29

Second Edition

Data Networks

DIMITRI BERTSEKAS
Massachusetts Institute of Technology

ROBERT GALLAGER
Massachusetts Institute of Technology



PRENTICE HALL, Englewood Cliffs, New Jersey 07632

IPR2025-00756, OnePlus Ex. 1013
Page 2 of 29

Library of Congress Cataloging-in-Publication Data

Bertsekas, Dimitri P.
Data networks / Dimitri Bertsekas, Robert Gallager. -- 2nd ed.
p. cm.
Includes bibliographical references and index.
ISBN 0-13-200916-1
1. Data transmission systems. I. Gallager, Robert G. II. Title.
TK5105.B478 1992
004.6--dc20 91-35561
CIP

Acquisitions editor: Pete Janzow
Production editor: Bayani Mendoza de Leon
Copy editor: Zeiders & Associates
Cover designer: Butler/Udell
Prepress buyer: Linda Behrens
Manufacturing buyer: Dave Dickey
Editorial assistant: Phyllis Morgan



©1992, 1987 by Prentice-Hall, Inc.
A Simon & Schuster Company
Englewood Cliffs, New Jersey 07632

All rights reserved. No part of this book may be
reproduced, in any form or by any means,
without permission in writing from the publisher.

Printed in the United States of America

10 9 8 7 6 5 4 3

ISBN 0-13-200916-1

Prentice-Hall International (UK) Limited, *London*
Prentice-Hall of Australia Pty. Limited, *Sydney*
Prentice-Hall Canada Inc., *Toronto*
Prentice-Hall Hispanoamericana, S. A., *Mexico*
Prentice-Hall of India Private Limited, *New Delhi*
Prentice-Hall of Japan, Inc., *Tokyo*
Simon & Schuster Asia Pte. Ltd., *Singapore*
Editora Prentice-Hall do Brasil, Ltda., *Rio de Janeiro*

through the network according to the current information available at the nodes visited. Virtual circuit routing is generally used in practice, although there are many interesting intermediate positions between virtual circuit routing and dynamic routing. The general issue of routing is treated in Chapter 5.

1.3 LAYERING

Layering, or layered architecture, is a form of hierarchical modularity that is central to data network design. The concept of modularity (although perhaps not the name) is as old as engineering. In what follows, the word *module* is used to refer either to a device or to a process within some computer system. What is important is that the module performs a given function in support of the overall function of the system. Such a function is often called the *service* provided by the module. The designers of a module will be intensely aware of the internal details and operation of that module. Someone who uses that module as a component in a larger system, however, will treat the module as a "black box." That is, the user will be uninterested in the internal workings of the module and will be concerned only with the inputs, the outputs, and, most important, the functional relation of outputs to inputs (*i.e.*, the service provided). Thus, a black box is a module viewed in terms of its input–output description. It can be used with other black boxes to construct a more complex module, which again will be viewed at higher levels as a bigger black box.

This approach to design leads naturally to a hierarchy of modules in which a module appears as a black box at one layer of the hierarchy, but appears as a system of lower-layer black boxes at the next lower layer of the hierarchy (see Fig. 1.6). At the overall system level (*i.e.*, at the highest layer of the hierarchy), one sees a small collection of top-layer modules, each viewed as black boxes providing some clear-cut service. At the next layer down, each top-layer module is viewed as a subsystem of lower-layer black boxes, and so forth, down to the lowest layer of the hierarchy. As shown in Fig. 1.6, each layer might contain not only black boxes made up of lower-layer modules but also simple modules that do not require division into yet simpler modules.

As an example of this hierarchical viewpoint, a computer system could be viewed as a set of processor modules, a set of memory modules, and a bus module. A processor module could, in turn, be viewed as a control unit, an arithmetic unit, an instruction fetching unit, and an input–output unit. Similarly, the arithmetic unit could be broken into adders, accumulators, and so on.

In most cases, a user of a black box does not need to know the detailed response of outputs to inputs. For example, precisely when an output changes in response to an input is not important as long as the output has changed by the time it is to be used. Thus, modules (*i.e.*, black boxes) can be specified in terms of tolerances rather than exact descriptions. This leads to standardized modules, which leads, in turn, to the possibility of using many identical, previously designed (*i.e.*, off-the-shelf) modules in the same system. In addition, such standardized modules can easily be replaced with new, functionally equivalent modules that are cheaper or more reliable.

IPR2025-00756, OnePlus Ex. 1013
Page 4 of 29

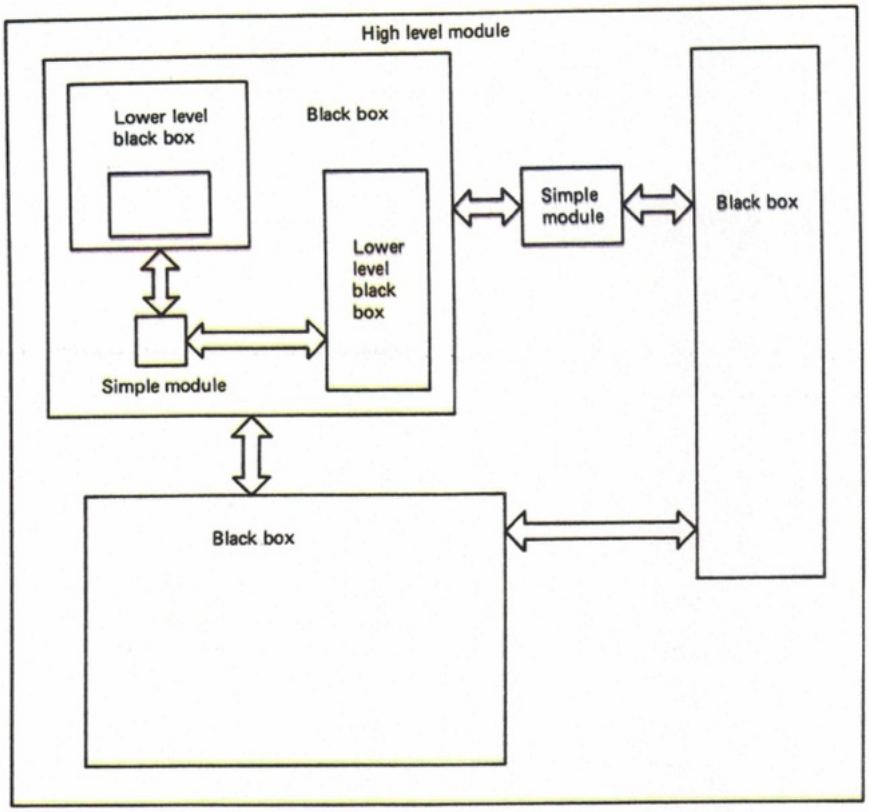


Figure 1.6 Hierarchy of nested black boxes. Each black box (except that at the lowest level) contains black boxes at a lower level, plus perhaps other modules.

All of these advantages of modularity (*i.e.*, simplicity of design; understandability; and standard, interchangeable, widely available modules) provide the motivation for a layered architecture in data networks. A layered architecture can be regarded as a hierarchy of nested modules or black boxes, as described above. Each given layer in the hierarchy regards the next lower layer as one or more black boxes which provide a specified service to the given higher layer.

What is unusual about the layered architecture for data networks is that the black boxes at the various layers are in fact distributed black boxes. The bottom layer of the hierarchy consists of the physical communication links, and at each higher layer, each black box consists of a lower-layer black box communication system plus a set of simple modules, one at each end of the lower-layer communication system. The simple modules associated with a black box at a given layer are called *peer processes* or *peer modules* (see Fig. 1.7).

In the simplest case, a black box consists of two peer processes, one at each of two nodes, and a lower-layer black box communication system connecting the two peer processes. One process communicates with its peer at the other node by placing a message

IPR2025-00756, OnePlus Ex. 1013
Page 5 of 29

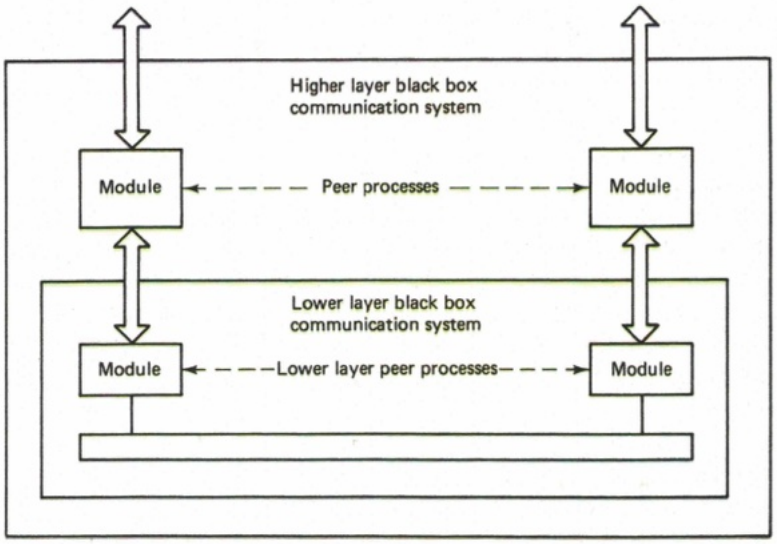


Figure 1.7 Peer processes within a black box communication system. The peer processes communicate through a lower-layer black box communication system that itself contains lower-layer peer processes.

into the lower-layer black box communication system. This lower-layer black box, as illustrated in Fig. 1.7, might in fact consist of two lower-layer peer processes, one at each of the two nodes, connected by a yet lower-layer black box communication system. As a familiar example, consider two heads of state who have no common language for communication. One head of state can then send a message to the peer head of state by a local translator, who communicates in a common language to a peer translator, who then delivers the message in the language of the peer head of state.

Note that there are two quite separate aspects to the communication between a module, say at layer n, and its layer n peer at another node. The first is the protocol (or distributed algorithm) that the peer modules use in exchanging messages or bit strings so as to provide the required functions or service to the next higher layer. The second is the specification of the precise interface between the layer n module at one node and the layer $n-1$ module at the same node through which the messages above are actually exchanged. The first aspect above is more important (and more interesting) for a conceptual understanding of the operation of a layered architecture, but the second is also vital in the actual design and standardization of the system. In terms of the previous example of communication between heads of state, the first aspect has to do with the negotiation between the heads of state, whereas the second has to do with each head of state ensuring that the translator can actually translate the messages faithfully.

Figure 1.8 illustrates such a layered architecture. The layers are those of the reference model of open systems interconnection (OSI) developed as an international standard for data networks by the International Standards Organization (ISO). Many existing networks, including SNA, DECNET, ARPANET, and TYMNET, have somewhat

IPR2025-00756, OnePlus Ex. 1013
Page 6 of 29

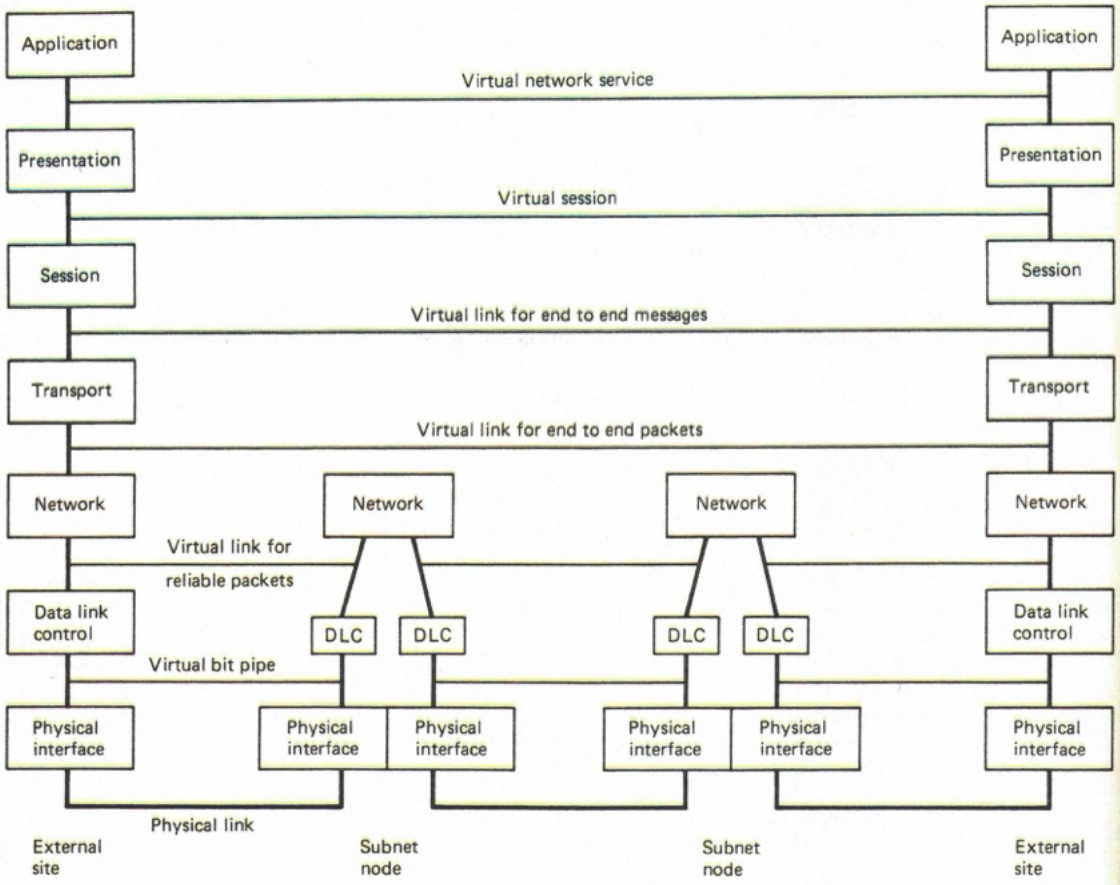


Figure 1.8 Seven-layer OSI network architecture. Each layer presents a virtual communication link with given properties to the next-higher layer.

different layers than this proposed standard. However, the OSI layers have a relatively clean structure that helps in understanding the concept of layering. Some of the variations used by these other networks are discussed later.

1.3.1 The Physical Layer

The function of the *physical layer* is to provide a virtual link for transmitting a sequence of bits between any pair of nodes (or any node and external site) joined by a physical communication channel. Such a virtual link is called a *virtual bit pipe*. To achieve this function, there is a physical interface module on each side of the communication channel whose function is to map the incoming bits from the next higher layer [*i.e.*, the data link control (DLC) layer] into signals appropriate for the channel, and at the receiving end, to map the signals back into bits. The physical interface module that

IPR2025-00756, OnePlus Ex. 1013
Page 7 of 29

performs these mapping functions is often called a *modem* (digital data *mo*dulator and *dem*odulator). The term *modem* is used broadly here to refer to any module that performs the function above, whether or not modulation is involved; for example, if the physical communication channel is a digital link (see Section 2.2), there is nothing for the modem to do other than interface with the DLC module.

Modems and communication channels are discussed in Section 2.2. The modem designer must be aware of the detailed characteristics of the communication channel (and different modems must be designed for different types of channels). To the higher layers, however, the black box formed by the modem–channel–modem combination appears as a bit pipe with the complexities of the physical channel hidden. Even viewed as a bit pipe, however, there are a few issues that must be discussed.

The first issue has to do with the timing of the bit sequence entering the bit pipe. There are three common situations. The first is that of a *synchronous* bit pipe where bits are transmitted and received at regular intervals (*i.e.*, 1 bit per t second interval for some t). The higher-layer DLC module must supply bits at this synchronous rate whether or not it has any real data to send. The second situation is that of an *intermittent synchronous* bit pipe where the DLC module supplies bits at a synchronous rate when it has data to send and stops sending bits when there are no data to send. The third situation is that of *asynchronous characters*, usually used with personal computers and low-speed terminals. Here, keyboard characters and various control characters are mapped into fixed-length bit strings (usually, eight-bit strings according to a standard mapping from characters to bit strings known as ASCII code), and the individual character bit strings are transmitted asynchronously as they are generated.

The next issue is that of the interface between the DLC module and the modem. One would think that not many problems should exist in delivering a string of bits from one module to another, especially if they are physically close. Unfortunately, there are a number of annoying details about such an interface. For example, the module on one end of the interface might be temporarily inoperable, and when both become operable, some initialization is required to start the flow of bits. Also, for synchronous operation, one side or the other must provide timing. To make matters worse, many different manufacturers provide the modules on either end, so there is a need for standardizing the interface. In fact, there are many such standards, so many that one applauds the effort but questions the success. Two of the better known are RS-232-C and the physical layer of X.21.

The RS-232-C interface approaches the problem by providing a separate wire between the two modules for each type of control signal that might be required. These wires from the modules are joined in a standard 25-pin connector (although usually many fewer wires are required). In communication jargon, the interface is between a DCE (data communication equipment), which is the modem in this case, and a DTE (data terminal equipment), which is the DLC layer and higher layers in this case.

As an example of the interface use, suppose that the DTE wants to start sending data (either on initialization or with a new data sequence in intermittent synchronous transmission). The DTE then sends a signal to the DCE on a "request-to-send" wire. The DCE replies with a signal on the "clear-to-send" wire. The DCE also sends a signal

IPR2025-00756, OnePlus Ex. 1013

Page 8 of 29

on the "DCE-ready" wire whenever it is operational and a signal on the "carrier detect" wire whenever it appears that the opposite modem and channel are operational. If the DTE receives all these signals (which are just level voltages), it starts to send data over the interface on the DTE-to-DCE data wire.

This interchange is a very simple example of a *protocol* or *distributed algorithm.* Each module performs operations based both on its own state and on the information received from the other module. Many less trivial protocols are developed in subsequent chapters. There are many other details in RS-232-C operation but no other new concepts.

It is sometimes helpful when focusing on the interface between the DLC module and the modem to view the wires between the modules as a physical channel and to view the DLC and modem as peer processes executing the interface protocol. To avoid confusion between the DLC module's major function as a peer process with the opposite DLC module and its lower-level function of interfacing with the modem, an extra dummy module is sometimes created (see Fig. 1.9) which exercises the interface protocol with the modem.

The X.21 physical layer interface is similar in function to RS-232-C, but it uses a smaller set of wires (eight wires are used, although there is a 15-pin connector) and is intended as an interface to a digital communication link. The idea is to avoid using a separate wire for each possible signal by doubling up on the use of wires by digital logic in the modules. The X.21 physical layer is used as the physical layer for the X.25 protocol, which is discussed in Chapter 2.

It should be clear from the above that there is a great conceptual advantage in removing the question of modem and modem interfaces from the higher-level aspects of networks. Note that this has already been done, in essence, in previous sections in referring to the number of bits per second that could be transmitted over communication links. It should also be noted, however, that modems cannot be totally segregated from network issues. For example, is it better to have a modem that transmits R bits/sec with an error rate of 10^{-6} or a modem that transmits $2R$ bits/sec with an error rate of 10^{-4}? This cannot be answered without some knowledge of how errors are eliminated at higher

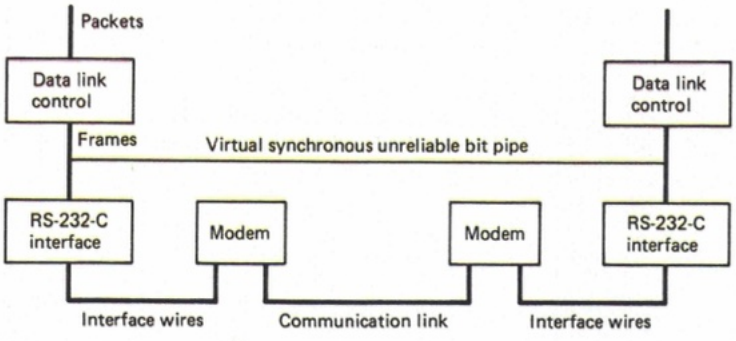


Figure 1.9 Layering with the interface between the DLC and the modem viewed as an interface over a physical medium consisting of a set of wires.

IPR2025-00756, OnePlus Ex. 1013
Page 9 of 29

layers of the architecture. Conversely, decisions on how and where to eliminate errors at higher layers should depend on the error rate and error characteristics at the physical layer.

1.3.2 The Data Link Control Layer

The second layer in Fig. 1.8 is the *data link control (DLC) layer*. Each point-to-point communication link (*i.e.*, the two-way virtual bit pipe provided by layer 1) has data link control modules (as peer processes) at each end of the link. The customary purpose of data link control is to convert the unreliable bit pipe at layer 1 into a higher-level, virtual communication link for sending packets asynchronously but error-free in both directions over the link. From the standpoint of the DLC layer, a packet is simply a string of bits that comes from the next higher layer.

The communication at this layer is asynchronous in two ways. First, there is a variable delay between the entrance of a packet into the DLC module at one end of the link and its exit from the other end. This variability is due both to the need to correct the errors that occur at the physical layer and to the variable length of the packets. Second, the time between subsequent entries of packets into the DLC module at one end of the link is also variable. The latter variability is caused both because higher layers might have no packets to send at a given time and also because the DLC is unable to accept new packets when too many old packets are being retransmitted due to transmission errors.

Data link control is discussed in detail in Chapter 2. In essence, the sending DLC module places some overhead control bits called a *header* at the beginning of each packet and some more overhead bits called a *trailer* at the end of each packet, resulting in a longer string of bits called a *frame*. Some of these overhead bits determine if errors have occurred in the transmitted frames, some request retransmissions when errors occur, and some delineate the beginning and ending of frames. The algorithms (or protocols) for accomplishing these tasks are distributed between the peer DLC modules at the two ends of each link and are somewhat complex because the control bits themselves are subject to transmission errors.

The DLC layers in some networks do not retransmit packets in the presence of errors. In these networks, packets in error are simply dropped and retransmission is attempted on an end-to-end basis at the transport layer. The relative merits of this are discussed in Section 2.8.2. Typically, the DLC layer ensures that packets leave the receiving DLC in the same order in which they enter the transmitting DLC, but not all data link control strategies ensure this feature; the relative merits of ordering are also discussed in Section 2.8.2.

Our previous description of the physical layer and DLC was based on point-to-point communication links for which the received waveform at one end of the link is a noisy replica of the signal transmitted at the other end. In some networks, particularly local area networks, some or all of the communication takes place over multiaccess links. For these links, the signal received at one node is a function of the signals from a multiplicity of transmitting nodes, and the signal transmitted from one node might be

IPR2025-00756, OnePlus Ex. 1013
Page 10 of 29

heard at a multiplicity of other nodes. This situation arises in satellite communication, radio communication, and communication over cables, optical fibers, and telephone lines with multiple taps. Multiaccess communication is treated in Chapter 4.

The MAC sublayer The appropriate layers for multiaccess communication are somewhat different from those in networks of point-to-point links. There is still the need for a DLC layer to provide a virtual error-free packet link to higher layers, and there is still the need for a physical layer to provide a bit pipe. However, there is also a need for an intermediate layer to manage the multiaccess link so that frames can be sent by each node without constant interference from the other nodes. This is called *medium access control* (MAC). It is usually considered as the lower sublayer of layer 2 with the conventional DLC considered as the higher sublayer. Figure 1.10 illustrates the relationship between these layers. The service provided by the MAC to the DLC is that of an intermittent synchronous bit pipe. The function of the MAC sublayer is to allocate the multiaccess channel so that each node can successfully transmit its frames without undue interference from the other nodes; see Chapter 4 for various ways of accomplishing this function.

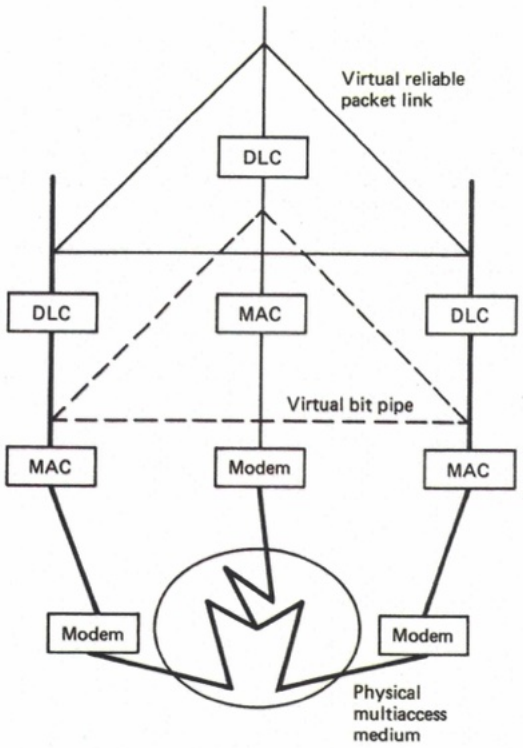


Figure 1.10 Layering for a multiaccess channel. The physical medium is accessed by all three users, each of whom hears the transmitted signals of the others. The DLC sublayer sees virtual point-to-point bit pipes below it. The MAC sublayer sees a multiaccess bit pipe, and the modems access the actual channel.

IPR2025-00756, OnePlus Ex. 1013
Page 11 of 29

1.3.3 The Network Layer

The third layer in Fig. 1.8 is the *network layer*. There is one network layer process associated with each node and with each external site of the network. All these processes are peer processes and all work together in implementing routing and flow control for the network. When a frame enters a node or site from a communication link, the bits in that frame pass through the physical layer to the DLC layer. The DLC layer determines where the frame begins and ends, and if the frame is accepted as correct, the DLC strips off the DLC header and trailer from the frame and passes the resulting packet up to the network layer module (see Fig. 1.11). A packet consists of two parts, a packet header followed by the packet body (and thus a frame at the DLC layer contains first the DLC header, next the packet header, next the packet body, and then the DLC trailer). The network layer module uses the packet header of an incoming packet, along with stored information at the module, to accomplish its routing and flow control functions. Part of the principle of layering is that the DLC layer does not look at the packet header or packet body in performing its service function, which is to deliver the packet reliably to the network layer at the next node. Similarly, the network layer does not use any of the information in the DLC header or trailer in performing its functions of routing and flow control. The reason for this separation is to allow improvements, modifications, and replacements in the internal operation of one layer without forcing the other to be modified.

Newly generated messages from users at an external site are processed by the higher layers, broken into packet-sized pieces if need be, and passed down from the transport layer module to the network module. These packet-sized pieces constitute the packet body at the network layer. The transport layer also provides additional information about how to handle the packet (such as where the packet is supposed to go), but this information is passed to the network layer as a set of parameters in accordance with the interfacing protocol between transport and network layer. The network layer module uses

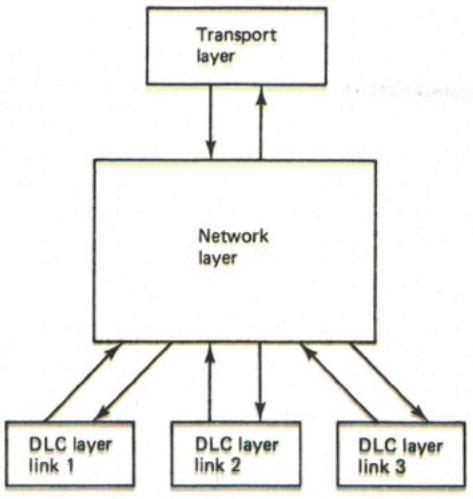


Figure 1.11 The network layer at a node or site can receive packets from a DLC layer for each incoming link and (in the case of a site) from the transport layer. It can send these packets out to the same set of modules.

IPR2025-00756, OnePlus Ex. 1013
Page 12 of 29

these parameters, along with its own stored information, to generate the packet header in accordance with the protocol between peer network layer modules.

Along with the transit packets arriving at the network layer module from the lower layer, and the new packets arriving from the higher layer, the network layer can generate its own control packets. These control packets might be specific to a given session, dealing with initiating or tearing down the session, or might have a global function, informing other nodes of link congestion, link failures, and so on.

For networks using virtual circuit routing (*i.e.*, in which the route for a session is fixed over the life of the session), the routing function at the network layer module consists of two parts. The first is to select a route when the virtual circuit is being initiated, and the second is to ensure that each packet of the session follows the assigned route. The selection of a route could be carried out in a distributed way by all the nodes, or could be carried out by the source node or by some central node entrusted with this task. No matter how the job is allocated between the different nodes, however, there is need for considerable communication, via network control packets, concerning the operating characteristics and level of traffic and delay throughout the network. This subject is treated in considerable depth in Chapter 5. Ensuring that each packet follows the assigned route is accomplished by placing enough information in the packet header for the network layer module at each node of the route to be able to forward the packet on the correct outgoing link (or to pass the packet body up to the transport layer when the destination is reached). Ways of doing this are discussed in Section 2.8.

Datagram networks, on the other hand, do not have a connection phase in which a route for a session is determined. Rather, each packet is routed individually. This appears to be a very natural and simple approach, but as Chapter 5 shows, the dynamics of the traffic patterns in a network and the lack of timely knowledge about those patterns at the nodes make this much more difficult than one would think. Most wide area networks use virtual circuits for this reason.

It is necessary here to make a distinction between virtual circuit or datagram *operation* at the network layer and virtual circuit or datagram *service*. The discussion above concerned the *operation* of the network layer; the user of the network layer (usually the transport layer) is concerned only with the *service* offered. Since successive packets of a session, using datagram operation, might travel on different routes, they might appear at the destination out of order. Thus (assuming that the network layer module at the destination does not reorder the packets), the service offered by such a network layer allows packets to get out of order. Typically, with datagram operation, packets are sometimes dropped also. As a result, datagram *service* is usually taken to mean that the network layer can deliver packets out of order, can occasionally fail to deliver packets, and requires no connection phase at the initiation of a session. Conversely, virtual circuit *service* is taken to mean that all packets are delivered once, only once, and in order, but that a connection phase is required on session initiation. We will often use the term *connectionless service* in place of *datagram service* and *connection-oriented service* in place of *virtual circuit service*. We shall see that the difference between connectionless and connection-oriented service has as much to do with quality of service, flow control, and error recovery as it does with routing.

IPR2025-00756, OnePlus Ex. 1013
Page 13 of 29

The other major function of the network layer, along with routing, is flow control, or congestion control. Some authors make a distinction between flow control and congestion control, viewing the first as avoiding sending data faster than the final destination can absorb it, and the second as avoiding congestion within the subnet. Actually, if the destination cannot absorb packets as fast as they are sent, those packets will remain in the subnet and cause congestion there. Similarly, if a link in the subnet is congested (*i.e.*, many packets are buffered in an adjacent node waiting for transmission on the link), then there are a number of mechanisms that cause the congestion to spread. Thus congestion is a global issue that involves both the subnet and the external sites, and at least at a conceptual level, it is preferable to treat it as a single problem.

Fundamentally, congestion occurs when the users of the network collectively demand more resources than the network (including the destination sites) has to offer. Good routing can help to alleviate this problem by spreading the sessions out over the available subnet resources. Good buffer management at the nodes can also help. Ultimately, however, the network layer must be able to control the flow of packets into the network, and this is what is meant by flow control (and why we use the term *flow control* in place of *congestion control*).

The control of packet flow into the network must be done in such a way as to prevent congestion and also to provide equitable service to the various sessions. Note that with connection-oriented service, it is possible for a session to negotiate its requirements from the network as a compromise between user desires and network utilization. Thus in some sense the network can guarantee the service as negotiated. With connectionless service, there is no such opportunity for negotiation, and equitable service between users does not have much meaning. This is another reason for the prevalence of connection-oriented service in wide area networks. In Chapter 6 we develop various distributed algorithms for performing the flow control function. As with routing, flow control requires considerable exchange of information between the nodes. Some of this exchange occurs through the packet headers, and some through control packets.

One might hope that the high link capacities that will be available in the future will make it possible to operate networks economically with low utilization, thus making flow control unnecessary. Unfortunately, this view appears overly simplistic. As link capacities increase, access rates into networks will also increase. Thus, even if the aggregate requirements for network service are small relative to the available capacity, a single malfunctioning user could dump enough data into the network quickly to cause serious congestion; if the network plays no regulatory role, this could easily lead to very chaotic service for other users.

The discussion of routing and flow control above has been oriented primarily toward wide area networks. Most local area networks can be viewed as using a single multiaccess channel for communication, and consequently any node is capable of receiving any packet. Thus routing is not a major problem for local area networks. There is a possibility of congestion in local area networks, but this must be dealt with in the MAC sublayer. Thus, in a sense, the major functions of the network layer are accomplished in the MAC sublayer, and the network layer is not of great importance in local area networks. For

IPR2025-00756, OnePlus Ex. 1013
Page 14 of 29

this reason, the arguments for virtual circuit operation and connection oriented service in the network layer do not apply to local area networks, and connectionless service is common there.

The network layer is conceptually the most complex of the layered hierarchy since *all* the peer processes at this layer must work together. For the lower layers (except for the MAC sublayer for multiaccess), the peer processes are paired, one at each side of a communication link. For the higher layers, the peer processes are again paired, one at each end of a session. Thus, the network layer and the MAC sublayer are the only layers in which the overall algorithms are distributed between many geographically separated processes.

Acquiring the ability to design and understand such distributed algorithms is one of the basic objectives of this book. Chapter 2 covers the simpler forms of distributed algorithms involving just two peer processes at opposite ends of a link. In Chapter 4 we treat distributed algorithms involving many peer processes in the context of the MAC sublayer, and Chapters 5 and 6 deal with distributed algorithms involving many peer processes at the network layer.

When the network layer and lower layers at all nodes and sites are regarded as one black box, a packet entering the network layer from the next higher layer at a site reappears at some later time at the interface between the network layer and the next higher layer at the destination site. Thus, the network layer appears as a virtual, packet-carrying, end-to-end link from origin site to destination site. Depending on the design of the network layer, this virtual link might be reliable, delivering every packet, once and only once, without errors, or might be unreliable, failing to deliver some packets and delivering some packets with errors. The higher layers then might have to recover from these errors. The network layer might also deliver all packets for each session in order or might deliver them out of order. The relative merits of these alternatives are discussed further in Section 2.8.

The internet sublayer Despite all efforts at standardization, different networks use different algorithms for routing and flow control at the network layer. We have seen some of the reasons for this variety in our discussion of wide area versus local area networks. Since these network layer algorithms are distributed and require close coordination between the various nodes, it is not surprising that one cannot simply connect different subnetworks together. The accepted solution to this problem is to create a new sublayer called the *internet sublayer*. This is usually regarded as being the top part of the network layer. Several subnets can be combined by creating special nodes called gateways between them. A gateway connecting two subnets will interface with each subnet through a network layer module appropriate for that subnet. From the standpoint of the subnet, then, a gateway looks like an external site.

Each gateway will have an internet sublayer module sitting on top of the network layer modules for the individual subnets. When a packet arrives at a gateway from one subnet, the corresponding network layer module passes the packet body and subsidiary information about the packet to the internet module (which thus acts like a transport layer module to the network layer module). This packet body and subsidiary information is

IPR2025-00756, OnePlus Ex. 1013
Page 15 of 29

then passed down to the other network layer module for forwarding on through the other subnet.

The internet modules also must play a role in routing and flow control. There is not a great deal of understanding in the field yet as to the appropriate ways for the internet sublayer and the various network layers to work together on routing and flow control. From a practical standpoint, the problem is exacerbated by the fact that the network layers for the subnets are usually in place, designed without the intention of later being used in a network of networks. Thus the internet layer must of necessity be somewhat ad hoc.

When combining local area networks, where routing and flow control are exercised at the MAC sublayer, it is often possible to replace the gateway between subnets with a bridge. Bridges interface different subnets at the DLC layer rather than at the network layer; for local area networks, this is possible because the routing and flow control are done in the MAC sublayer. In Chapter 5 we discuss gateways and bridges in greater detail, particularly with respect to routing.

1.3.4 The Transport Layer

The fourth layer in Fig. 1.8 is the *transport layer*. Here, for each virtual end-to-end link provided by the network layer (or internet sublayer), there is a pair of peer processes, one at each end of the virtual end-to-end link. The transport layer has a number of functions, not all of which are necessarily required in any given network.

First, the transport layer breaks messages into packets at the transmitting end and reassembles packets into messages at the receiving end. This reassembly function is relatively simple if the transport layer process has plenty of buffer space available, but can be quite tricky if there is limited buffer space that must be shared between many virtual end-to-end links. If the network layer delivers packets out of order, this reassembly problem becomes even more difficult.

Second, the transport layer might multiplex several low-rate sessions, all from the same source site and all going to the same destination site, into one session at the network layer. Since the subnet sees only one session in this case, the number of sessions in the subnet and the attendant overhead is reduced. Often this is carried to the extreme in which all sessions with a common source site and common destination site are multiplexed into the same session. In this case, the addressing at the network layer need only specify the source and destination sites; the process within the source site and destination site are then specified in a transport layer header.

Third, the transport layer might split one high-rate session into multiple sessions at the network layer. This might be desirable if the flow control at the network layer is incapable of providing higher-rate service to some sessions than others, but clearly a better solution to this problem would be for the network layer to adjust the rate to the session requirement.

Fourth, if the network layer is unreliable, the transport layer might be required to achieve reliable end-to-end communication for those sessions requiring it. Even when the network layer is designed to provide reliable communication, the transport layer has to be

IPR2025-00756, OnePlus Ex. 1013
Page 16 of 29

involved when one or the other end site fails or when the network becomes disconnected due to communication link failures. These failure issues are discussed further in Section 2.8 and in Chapters 5 and 6.

Fifth, end-to-end flow control is often done at the transport layer. There is little difference between end-to-end flow control at the transport layer and network layer (or internet sublayer if it exists). End-to-end flow control at the transport layer is common in practice but makes an integrated approach to avoiding congestion somewhat difficult. This is discussed further in Section 2.9.4 and in Chapter 6.

A header is usually required at the transport layer; this transport header, combined with the data being transported, serves as the packet body passed on to the network layer. Thus the actual body of data is encapsulated in a sequence of headers with the lowest layers on the outside (see Fig. 1.12). At the destination, these layer headers are peeled off in passing up through the various layers. In ISO terminology, the body of data shown in the figure is referred to as a transport service data unit (T-SDU). This data unit, along with the transport header, is referred to as a transport protocol data unit (T-PDU). This unit is also the body of the packet at the network layer, which is sometimes referred to as a network service data unit (N-SDU). Similarly, the packet body plus packet header is referred to as a network protocol data unit (N-PDU). Similarly, each layer in the hierarchy has an SDU, as the unit coming in from the higher layer, and a PDU as the unit going down to the next-lower layer. It is difficult to know where to take a stand against acronymitis in the network field, but we will continue to use the more descriptive terminology of messages, packets, and frames.

1.3.5 The Session Layer

The *session layer* is the next layer above the transport layer in the OSI hierarchy of Fig. 1.8. One function of the session layer is akin to the directory assistance service in the telephone network. That is, if a user wants an available service in the network

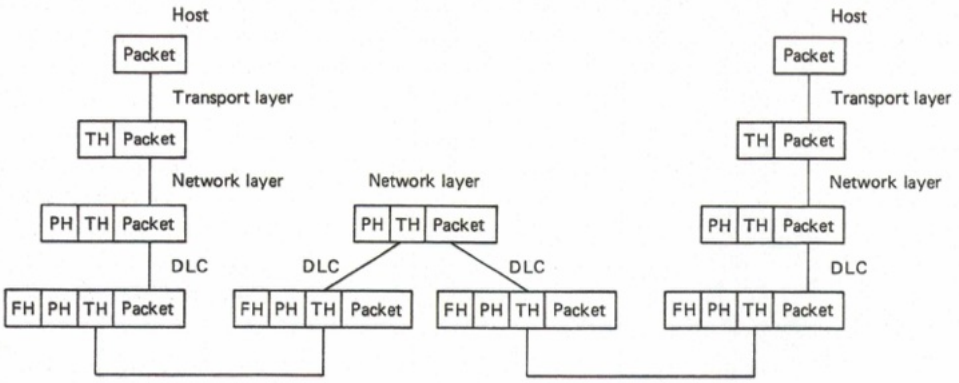


Figure 1.12 Illustration of various headers on a frame. Note that each layer looks only at its own header.

IPR2025-00756, OnePlus Ex. 1013
Page 17 of 29

but does not know where to access that service, this layer provides the transport layer with the information needed to establish the session. For example, this layer would be an appropriate place to achieve load sharing between many processors that are sharing computational tasks within a network.

The session layer also deals with *access rights* in setting up sessions. For example, if a corporation uses a public network to exchange records between branch offices, those records should not be accessible to unauthorized users. Similarly, when a user accesses a service, the session layer helps deal with the question of who pays for the service.

In essence, the session layer handles the interactions between the two end points in setting up a session, whereas the network layer handles the subnet aspects of setting up a session. The way that session initiation is divided between session layer, transport layer, and network layer varies from network to network, and many networks do not have these three layers as separate entities.

1.3.6 The Presentation Layer

The major functions of the *presentation layer* are data encryption, data compression, and code conversion. The need for encryption in military organizations is obvious, but in addition, corporations and individual users often must send messages that should only be read by the intended recipient. Although data networks should be designed to prevent messages from getting to the wrong recipients, one must expect occasional malfunctions both at the external sites and within the subnet; this leads to the need for encryption of critical messages.

The desirability of data compression in reducing the number of bits to be communicated has already been mentioned. This function could be performed at any of the layers, but there is some advantage in compressing the data for each session separately, in the sense that different sessions have different types of redundancy in their messages. In particular, data compression must be done (if at all) before encryption, since encrypted data will not have any easily detectable redundancy.

Finally, code conversion is sometimes necessary because of incompatible terminals, printers, graphics terminals, file systems, and so on. For example, some terminals use the ASCII code to represent characters as 8-bit bytes, whereas other terminals use the EBCDIC code. Messages using one code must be converted to the other code to be readable by a terminal using the other code.

1.3.7 The Application Layer

The *application layer* is simply what is left over after the other layers have performed their functions. Each application requires its own software (*i.e.*, peer processes) to perform the desired application. The lower layers perform those parts of the overall task that are required for many different applications, while the application layer does that part of the task specific to the particular application.

At this point, the merits of a layered approach should be clear, but there is some question about which functions should be performed at each layer. Many networks omit

IPR2025-00756, OnePlus Ex. 1013
Page 18 of 29

the session and presentation layers, and as we have seen, the lower layers are now divided into sublayers. An even more serious issue is that in an effort to achieve agreement on the standards, a number of alternatives have been introduced which allow major functions to be either performed or not at various layers. For example, error recovery is sometimes done at the DLC layer and sometimes not, and because of this, the higher layers cannot necessarily count on reliable packet transfer. Thus, even within the class of networks that conform to the OSI reference model, there is considerable variation in the services offered by the various layers. Many of the existing networks described later do not conform to the OSI reference model, and thus introduce even more variation in layer services. Broadband ISDN networks, for example, do routing and flow control at the physical layer (in a desire to simplify switch design), thus making the network look like an end-to-end bit pipe from the origin to destination.

Even with all these problems, there is a strong desire for standardization of the interfaces and functions provided by each layer, even if the standard is slightly inappropriate. This desire is particularly strong for international networks and particularly strong among smaller equipment manufacturers who must design equipment to operate correctly with other manufacturers' equipment. On the other hand, standardization is an impediment to innovation, and this impediment is particularly important in a new field, such as data networks, that is not yet well understood. Fortunately, there is a constant evolution of network standards. For example, the asynchronous transfer node (ATM) protocol of broadband ISDN circumvents the ISO network layer standard by performing the function of the network layer at the physical layer (see Section 2.10).

One particular difficulty with the seven layers is that each message must pass through seven processes before even entering the subnet, and all of this might generate considerable delay. This text neither argues for this particular set of layers nor proposes another set. Rather, the objective is to create an increased level of understanding of the functions that must be accomplished, with the hope that this will contribute to standardization. The existing networks examined in subsequent chapters do not, in fact, have layers that quite correspond to the OSI layers.

1.4 A SIMPLE DISTRIBUTED ALGORITHM PROBLEM

All of the layers discussed in Section 1.3 have much in common. All contain peer processes, one at each of two or more geographically separated points, and the peer processes communicate via the communication facility provided by the next lower layer. The peer processes in a given layer have a common objective (*i.e.*, task or function) to be performed jointly, and that objective is achieved by some combination of processing and interchanging information. The algorithm to achieve the objective is a *distributed algorithm* or a *protocol*. The distributed algorithm is broken down into a set of *local algorithms*, one of which is performed by each peer process. The local algorithm performed by one process in a set of peers consists of carrying out various operations on the available data, and at various points in the algorithm, either sending data to one or more other peer processes or reading (or waiting for) data sent by another peer process.

IPR2025-00756, OnePlus Ex. 1013
Page 19 of 29

variable delays, subject to a given minimum. As will be seen in the next section, this is used only when high data rates are not an important consideration.

In Sections 2.3 to 2.7 we treat the DLC layer, which is the primary focus of this chapter. For each point-to-point link, there are two DLC peer modules, one module at each end of the link. For traffic in a given direction, the sending DLC module receives packets from the network layer module at that node. The peer DLC modules employ a distributed algorithm, or protocol, to transfer these packets to the receiving DLC and thence to the network layer module at that node. In most cases, the objective is to deliver the packets in the order of arrival, with neither repetitions nor errors. In accomplishing this task, the DLC modules make use of the virtual bit pipe (with errors) provided by the physical layer.

One major problem in accomplishing the objective above is that of correcting the bit errors that occur on the virtual bit pipe. This objective is generally accomplished by a technique called ARQ (*a*utomatic *r*epeat re*q*uest). In this technique, errors are first detected at the receiving DLC module and then repetitions are requested from the transmitting DLC module. Both the detection of errors and the requests for retransmission require a certain amount of control overhead to be communicated on the bit pipe. This overhead is provided by adding a given number of bits (called a *header*) to the front of each packet and adding an additional number of bits (called a *trailer*) to the rear (see Fig. 2.2). A packet, extended by this header and trailer, is called a *frame*. From the standpoint of the DLC, a packet is simply a string of bits provided as a unit by the network layer; examples arise later where single "packets," in this sense, from the network layer contain pieces of information from many different sessions, but this makes no difference to the DLC layer.

Section 2.3 treats the problem of error detection and shows how a set of redundant bits in the trailer can be used for error detection. Section 2.4 then deals with retransmission requests. This is not as easy as it appears, first because the requests must be embedded into the data traveling in the opposite direction, and second because the opposite direction is also subject to errors. This provides our first exposure to a real distributed algorithm, which is of particular conceptual interest since it must operate in the presence of errors.

In Section 2.5 we discuss framing. The issue here is for the receiving DLC module to detect the beginning and end of each successive frame. For a synchronous bit pipe, the bits within a frame must contain the information to distinguish the end of the frame; also the idle fill between frames must be uniquely recognizable. This problem becomes even more interesting in the presence of errors.

A widely accepted standard of data link control is the HDLC protocol. This is discussed in Section 2.6, followed in Section 2.7 by a general discussion of how to initialize and to disconnect DLC protocols. This topic might appear to be trivial, but on closer inspection, it requires careful thought.

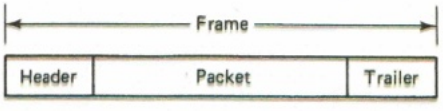


Figure 2.2 Frame structure. A packet from the network layer is extended in the DLC layer with control bits in front and in back of the packet.

IPR2025-00756, OnePlus Ex. 1013
Page 20 of 29

2.4.2 Go Back *n* ARQ

Go back n ARQ is the most widely used type of ARQ protocol; it appears in the various standard DLC protocols, such as HDLC, SDLC, ADCCP, and LAPB. It is not elucidating to know the meaning of these acronyms, but in fact, these standards are almost the same. They are discussed in Section 2.6, and some of their differences are mentioned there. Go back n is also the basis of most error recovery procedures at the transport layer.

The basic idea of go back n is very simple. Incoming packets to a transmitting DLC module for a link from A to B are numbered sequentially, and this sequence number SN is sent in the header of the frame containing the packet. In contrast to stop-and-wait ARQ, several successive packets can be sent without waiting for the next packet to be requested. The receiving DLC at B operates in essentially the same way as stop-and-wait ARQ. It accepts packets only in the correct order and sends request numbers RN back to A; the effect of a given request RN is to acknowledge all packets prior to RN and to request transmission of packet RN.

The go back number $n \geq 1$ in a go back n protocol is a parameter that determines how many successive packets can be sent in the absence of a request for a new packet. Specifically, node A is not allowed to send packet $i+n$ before i has been acknowledged (*i.e.*, before $i+1$ has been requested). Thus, if i is the most recently received request from node B, there is a "window" of n packets, from i to $i+n-1$, that the transmitter is allowed to send. As successively higher-numbered requests are received from B, this window slides upward; thus go back n protocols are often called *sliding window* ARQ protocols.

Figure 2.24 illustrates the operation of go back 7 ARQ when piggybacking of request numbers is being used and when there are no errors and a constant supply of traffic. Although the figure portrays data traffic in both directions, the flow of sequence numbers is shown in one direction and request numbers in the other. Note that when the first frame from A (containing packet 0) is received at B, node B is already in the middle of transmitting its second frame. The piggybacked request number at node B is traditionally in the frame header, and the frame is traditionally completely assembled before transmission starts. Thus when packet 0 is received from A, it is too late for node B to request packet 1 in the second frame, so that $RN = 1$ does not appear until the third frame from B. When this frame is completely received at A, the window at A "slides up" from $[0,6]$ to $[1,7]$.

Note that even in the absence of transmission errors, there are several sources of delay between the time that a packet is first assembled into a frame at A and the time when A receives an acknowledgment of the packet. First there is the transmission time of the frame, then the propagation delay, then the wait until the frame in transmission at B is completed, then the transmission time of the frame carrying the acknowledgment, and finally, the B-to-A propagation delay; the effect of these delays is discussed later.

Figure 2.25 shows the effect of error frames on go back 4 ARQ. The second frame from A, carrying packet 1, is received in error at node B. Node B continues to look for packet 1 and to transmit $RN = 1$ in frames from B to A. Packets 2, 3, and 4 from A

IPR2025-00756, OnePlus Ex. 1013
Page 21 of 29

packet. In many situations, the probability of one or more errors in a frame is 10^{-4} or less, and in this case, retransmitting many packets for each frame in error has little effect on efficiency. There are some communication links, however, for which small error probabilities per frame are very difficult to achieve, even with error correction in the modems. For other links (*e.g.*, high-speed links and satellite links), the number of frames transmitted in a round- trip delay time is very large. In both these cases, selective repeat ARQ can be used to increase efficiency.

The basic idea of selective repeat ARQ for data on a link from A to B is to accept out-of-order packets and to request retransmissions from A only for those packets that are not correctly received. There is still a go back number, or window size, n, specifying how far A can get ahead of RN, the lowest-numbered packet not yet correctly received at B.

Note that whatever ARQ protocol is used, only error-free frames can deliver packets to B, and thus, if p is the probability of frame error, the expected number η of packets delivered to B per frame from A to B is bounded by

$$\eta \leq 1 - p \tag{2.29}$$

As discussed later, this bound can be approached (in principle) by selective repeat ARQ; thus $1 - p$ is sometimes called the throughput of ideal selective repeat. Ideal go back n ARQ can similarly be defined as a protocol that retransmits the packets in one round-trip delay each time that a frame carrying the packet awaited by the receiving DLC is corrupted by errors. The throughput of this ideal is shown in Problem 2.26 to be

$$\eta \leq \frac{1 - p}{1 + p\beta} \tag{2.30}$$

where β is the expected number of frames in a round-trip delay interval. This indicates that the increase in throughput available with selective repeat is significant only when $p\beta$ is appreciable relative to 1.

The selective repeat algorithm at node A, for traffic from A to B, is the same as the go back n algorithm, except that (for reasons soon to be made clear) the modulus m must satisfy $m \geq 2n$. At node B, the variable RN has the same significance as in go back n; namely, it is the lowest-numbered packet (or the lowest-numbered packet modulo m) not yet correctly received. In the selective repeat algorithm, node B accepts packets anywhere in the range RN to $RN + n - 1$. The value of RN must be sent back to A as in go back n, either piggybacked on data frames or sent in separate frames. Usually, as discussed later, the feedback from node B to A includes not only the value of RN but also additional information about which packets beyond RN have been correctly received. In principle, the DLC layer still releases packets to the higher layer in order, so that the accepted out-of-order packets are saved until the earlier packets are accepted and released.

We now assume again that frames do not get out of order on the links and proceed to see why a larger modulus is required for selective repeat than for go back n. Assume that a frame is received at node B at some given time t_2 and that the frame was generated at node A at time t_1. If SN and RN are considered as integers, Eqs. (2.24) and (2.25)

IPR2025-00756, OnePlus Ex. 1013
Page 22 of 29

packet size. Finally, there is the effect of transmission errors. Large frames have a somewhat higher probability of error than small frames (although since errors are usually correlated, this effect is not as pronounced as one would think). For most links in use (with the notable exception of radio links), the probability of error on reasonable-sized frames is on the order of 10^{-4} or less, so that this effect is typically less important than the other effects discussed. Unfortunately, there are many analyses of optimal maximum frame length in the literature that focus only on this effect. Thus, these analyses are relevant only in those special cases where error probabilities are very high.

In practice, typical maximum frame lengths for wide area networks are on the order of 1 to a few thousand bits. Local area networks usually have much longer maximum frame lengths, since the path usually consists of a single multiaccess link. Also, delay and congestion are typically less important there and long frames allow most messages to be sent as a single packet.

Fixed frame length When all frames (and thus all packets) are required to be the same length, message lengths are not necessarily an integer multiple of the packet length, and the last packet of a message will have to contain some extra bits, called fill, to bring it up to the required length. Distinguishing fill from data at the end of a packet is conceptually the same problem as determining the end of a frame for variable-length frames. One effect of the fill in the last packet is an additional loss of efficiency, especially if the fixed packet length is much longer than many messages. Problem 2.39 uses this effect to repeat the optimization of Eq. (2.44), and as expected, the resulting optimal packet length is somewhat smaller with fixed frame lengths than with variable frame lengths. A considerably more important practical effect comes from the need to achieve a small delay for stream-type traffic. As was pointed out earlier, 64 kbps voice traffic must use packets on the order or 500 bits or less, and this requirement, for fixed frame length, forces *all* packets to have such short lengths. This is the primary reason why ATM (see Section 2.10) uses 53 byte frames even though very much longer frames would be desirable for most of the other traffic.

2.6 STANDARD DLCs

There are a number of similar standards for data link control, namely HDLC, ADCCP, LAPB, and SDLC. These standards (like most standards in the network field) are universally known by their acronyms, and the words for which the acronyms stand are virtually as cryptic as the acronyms themselves. HDLC was developed by the International Standards Organization (ISO), ADCCP by the American National Standards Institute (ANSI), LAPB by the International Consultative Committee on Telegraphy and Telephony (CCITT), and SDLC by IBM. HDLC and ADCCP are virtually identical and are described here. They have a wide variety of different options and modes of operation. LAPB is the DLC layer for X.25, which is the primary standard for connecting an external site to a subnet; LAPB is, in essence, a restricted subset of the HDLC options and modes, as will be described later. Similarly, SDLC, which was the precursor of HDLC and ADCCP, is essentially another subset of options and modes.

IPR2025-00756, OnePlus Ex. 1013
Page 23 of 29

HDLC and ADCCP are designed for a variety of link types, including both multiaccess or point-to-point links and both full-duplex links (*i.e.*, both stations can transmit at once) or half-duplex links (*i.e.*, only one station can transmit at a time). There are three possible *modes* of operation, one of which is selected when the link is initialized for use.

The first mode, the *normal response mode* (NRM), is for a master–slave type of link use such as is typical between a computer and one or more peripherals (the word "normal" is of historical significance only). There is a primary or master station (the computer) and one or more secondary or slave stations. The secondary stations can send frames only in response to polling commands from the primary station. This mode is reasonable in multiaccess situations where the polling commands ensure that only one station transmits at a time (see Chapter 4 for a general discussion of multiaccess communication). NRM is also reasonable for half-duplex links or for communication with peripherals incapable of acting in other modes.

The second mode, the *asynchronous response mode* (ARM), is also a master–slave mode in which the secondary nodes are not so tightly restricted. It is not widely used and will not be discussed further.

The third mode, the *asynchronous balanced mode* (ABM), is for full-duplex point-to-point links between stations that take equal responsibility for link control. This is the mode of greatest interest for point-to-point links in data networks. It is also used, in slightly modified form, for multiaccess communication in local area networks. LAPB uses only this mode, whereas SDLC uses only the first two modes.

We first describe the frame structure common to all three modes and all four standards. Then the operation of the normal response mode (NRM) and the asynchronous balanced mode (ABM) are described in more detail. Although ABM is our major interest, the peculiarities of ABM will be understood more easily once NRM is understood.

Figure 2.38 shows the frame structure. The flag, 01^60, is as described in Section 2.5.2, with bit stuffing used within the frame (the flag is not considered as part of the frame proper). One or more flags must separate each pair of frames. The CRC uses the CRC-CCITT generator polynomial $g(D) = D^{16} + D^{12} + D^5 + 1$ as explained in Section 2.3.4; it checks the entire frame (not including the flag). One modification is that the first 16 bits of the frame are inverted in the calculation of the CRC (although not inverted in transmission). Also, the remainder is inverted to form the CRC. At the receiving DLC, the first 16 bits of the frame are similarly inverted for recalculation of the remainder, which is then inverted for comparison with the received CRC. One reason for this is to avoid having an all-zero string (which is common if the link or modem is malfunctioning) satisfy the CRC. Another reason is to avoid satisfying the CRC if a few

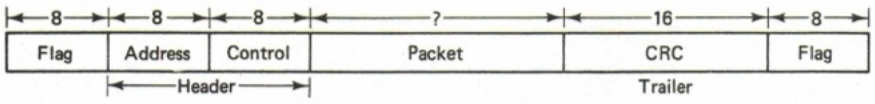


Figure 2.38 Frame structure of standard DLCs. The address, control, and CRC fields can each be optionally extended in length.

IPR2025-00756, OnePlus Ex. 1013
Page 24 of 29

zeros are added or deleted from the beginning or end of the frame. As an option, the 32-bit CRC discussed in Section 2.3.4 can be used. This optional 32-bit CRC is used in the IEEE 802 standard for local area networks.

The address field normally consists of one byte (eight bits). For the normal response mode (NRM), the address is always that of the secondary station; that is, when the primary sends a frame to a secondary, it uses the secondary's address, and when the secondary responds, it uses its own address. Note that this address is not intended as a destination for a session, but simply to distinguish stations for multiaccess use. For point-to-point links, the address field has no natural function, although we shall see later that it has some rather peculiar uses in ABM. There is an option in which the address field is extendable to an arbitrary number of 1 or more bytes; the first bit in a byte is 1 if that byte is the last address byte, and otherwise it is 0.

The layout of the control field is shown in Fig. 2.39. There are three different frame formats: *information*, *supervisory*, and *unnumbered*. The information frames are the actual packet-carrying frames; they normally use go back n ARQ with a modulus of 8. As shown in the figure, the sequence and request numbers SN and RN are part of the control field. The supervisory frames are used to return ARQ information (*i.e.*, RN) either when there are no data packets to send or when a speedy ack or nak is needed. Finally, the unnumbered frames are used for the initiation and termination of a link and for sending various kinds of supplementary control information.

There is an option under which the control field is two bytes long and in which SN and RN are each seven bits long and each use a modulus of 128. Aside from this lengthening of SN and RN, this option is essentially the same as the normal single-byte control field, which is assumed in the following description.

The first bit of the control field distinguishes the information format from the supervisory and unnumbered formats. Similarly, for noninformation formats, the second bit distinguishes supervisory from unnumbered formats. The fifth bit of the control field is the *poll final* (P/F) bit. For the normal response mode, the primary sends a 1 in this bit position to poll the secondary. The secondary must then respond with one or more frames, and sends a 1 in this bit position to denote the final frame of the response. This polling and response function of the P/F bit in NRM is used for supervisory and unnumbered frames as well as information frames. The P/F bit is used in a rather strange way for the asynchronous balanced mode (ABM), as described later.

There are four types of supervisory frames: *receive-ready* (RR), *receive-not-ready* (RNR), *reject* (REJ), and *selective-reject* (SREJ). The type is encoded in the third and

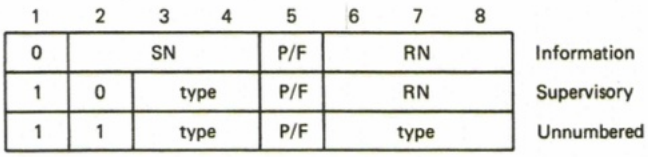


Figure 2.39 Structure of the control field for the information, supervisory, and unnumbered frame formats.

IPR2025-00756, OnePlus Ex. 1013
Page 25 of 29

fourth bits of the control field. All of these have the basic function of returning ARQ information (*i.e.*, all packets with numbers less than RN are acknowledged) and consequently they contain neither a sequence number nor an information field. RR is the normal response if there is no data packet on which to piggyback RN. RNR means in addition that the station is temporarily unable to accept further packets; this allows a rudimentary form of link flow control—that is, if the buffers at one end of the link are filling up with received packets, the other end of the link can be inhibited from further transmission.

The REJ supervisory frame is sent to indicate either a received frame in error or a received frame with other than the awaited sequence number. Its purpose is to improve the efficiency of go back n ARQ, as explained in Section 2.4.2. Similarly, SREJ provides a primitive form of selective repeat. It acks all packets before RN and requests retransmission of RN itself followed by new packets; one cannot provide supplementary information about multiple required retransmissions. This facility can be useful, however, on satellite links, where there is a very small probability of more than one frame error in a round-trip delay. The implementation of REJ and SREJ is optional, whereas RR and RNR are required.

The unnumbered frames carry no sequence numbers and are used for initializing the link for communication, disconnecting the link, and special control purposes. There are five bits in the control field available for distinguishing different types of unnumbered frames, but not all are used. Six of these types will be referred to generically as *set mode* commands. There is one such command for each mode (NRM, ARM, and ABM), and one for each mode using the extended control field. This command is sent by a primary (for NRM or ARM) and by either node (for ABM) to initialize communication on a link. Upon sending or receiving this command, a station sets its sequence and request numbers to zero. The recipient of the command must acknowledge it with an unnumbered frame, either an *unnumbered ack* which agrees to start using the link, or a *disconnected mode* which refuses to start using the link. Similarly, there is a disconnect command which is used to stop using the link; this also requires an unnumbered ack in response.

Figure 2.40 shows the operation of the NRM mode for a primary and two secondaries. Note that the primary initiates communication with each separately and disconnects from each separately. Note also that the P/F bit is set to 1 on each unnumbered frame that requires a response. After a given unnumbered frame with $P = 1$ is sent to a particular secondary, no new unnumbered frame with $P = 1$ may be sent to that secondary until the given frame is acknowledged by a frame with $F = 1$; naturally the given frame may be repeated after a time-out waiting for the ack. This alternation of unnumbered frames with acknowledgments is similar to the stop-and-wait protocol. Section 2.7 explains this similarity carefully and shows that the acknowledgment strategy here contains the same defect as stop-and-wait strategies without numbered acknowledgments.

In the asynchronous balanced mode, the decision was made, in designing these protocols, to have each station act as both a primary and a secondary (thus replacing a simple problem with a more familiar but more difficult problem). When a station sends a primary frame (in its role as a primary station), it uses the other station's address in the address field, and the P/F bit is interpreted as a poll bit. When a station sends

IPR2025-00756, OnePlus Ex. 1013
Page 26 of 29

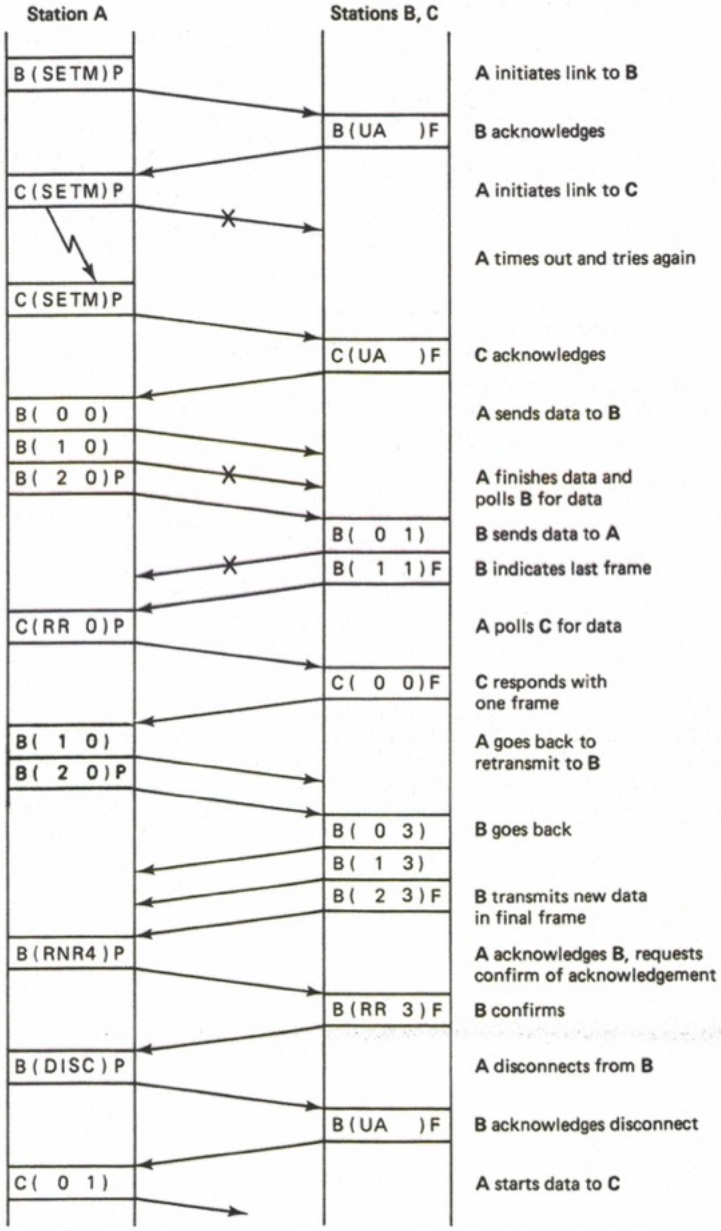


Figure 2.40 Normal response mode (NRM) operation of primary A with two secondaries, B and C. SETM refers to the command setting, the NRM mode with $SN = 0$ and $RN = 0$ at both stations. The address of each frame is given first, followed by (SN,RN) for information frames, (type,RN) for supervisory frames, and (type) for unnumbered frames; this is followed by a P/F bit if set to 1. Note that A uses different SN and RN values for B and C.

IPR2025-00756, OnePlus Ex. 1013
Page 27 of 29

a secondary frame (in its role as a secondary), it uses its own address, and the P/F bit is interpreted as a final bit. In customary operation, all information frames and all unnumbered frames requiring acks are sent as primary frames; all unnumbered frames carrying acks or responses, and most supervisory frames (which ack primary data frames), are sent as secondary frames.

What this means in practice is that the P/F bit is either P or F, depending on the address field. This bit, in conjunction with the address field, has three possible values: $P = 1$, requiring a response from the other station acting in a secondary role; $F = 1$, providing the secondary response, and neither. Given the somewhat inadequate performance of this poll/final alternation, this modified use of the P/F bit was an unfortunate choice.

Figure 2.41 shows a typical operation of ABM. The most interesting part of this is the next-to-last frame sent by station A; this is a supervisory frame sent with $P = 1$, thus requiring an acknowledgment. Since A has signified unwillingness to accept any more packets, a response from B signifies that B knows that packets up to three inclusive have been accepted and that no more will be accepted. Thus, upon receiving the acknowledgment of this, A can disconnect, knowing that B knows exactly which packets from B have been accepted; A also knows which of its packets have been accepted, so the disconnect leaves no uncertainty about the final condition of any packets at either A

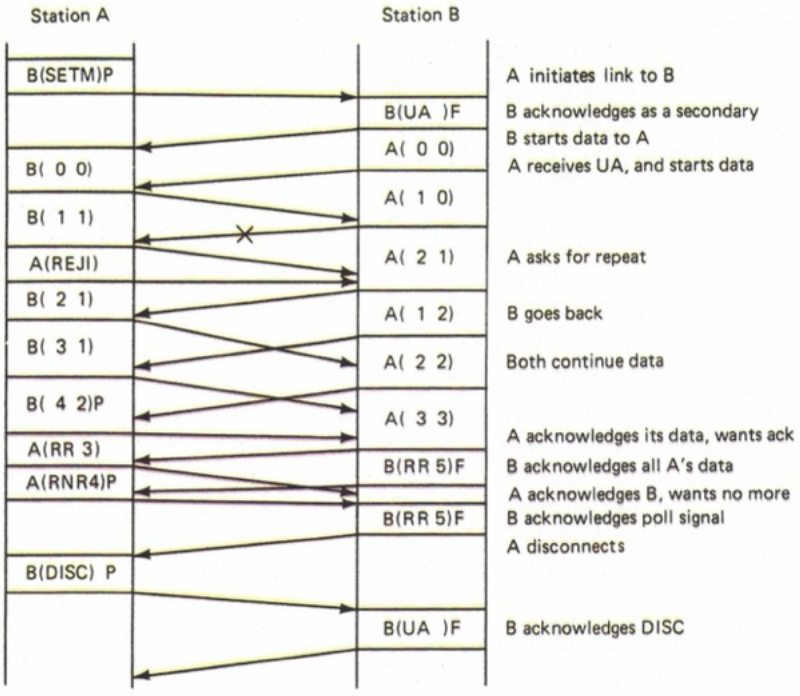


Figure 2.41 Asynchronous balanced mode (ABM) of operation. SETM refers to the set ABM command, which initiates the link with SN and RN at both sides equal to 0. Note the use of addresses in different types of frames.

IPR2025-00756, OnePlus Ex. 1013
Page 28 of 29

or B. Note that this "ideal" disconnect requires some previous agreement (not specified by the standard) between A and B; in particular, A must send RNR, with $P = 1$ and get an ack before disconnecting—this lets B know that A did not go back to accepting packets again temporarily before the disconnect. This "ideal" disconnect does not work, of course, if the line is being disconnected because of a malfunction; in this case, the idea of an orderly sequence of RNR, ack, disconnect, and ack is unrealistic.

Another type of unnumbered frame is "frame reject." This is sent in response to a received frame with a valid CRC but otherwise impossible value (such as exceeding the maximum frame length, being shorter than the minimum frame length, having an RN value not between SN_{min} and $SN_{\text{max}} - 1$, or having an impossible control field). Thus, either an undetectable error has occurred or one of the stations has malfunctioned. Such an error is outside the realm of what the ARQ is designed for, and it should be reported at each station. The usual response to a "frame reject" is for the opposite station either to reinitialize the link with a set mode command or send an unnumbered "reset" command. The first resets RN and SN at each station to 0, and the second sets SN to 0 at the station that sent the faulty frame and RN to 0 at the receiving station. Both stations will again start to send packets (starting from the first packet not acked), but because of the resetting, some packets might arrive more than once and out of order. This is not a fault of the protocol, but a recognition that node malfunctions and undetectable errors cannot be dealt with.

2.7 INITIALIZATION AND DISCONNECT FOR ARQ PROTOCOLS

The discussion of ARQ in Section 2.4 assumed throughout that the protocol was correctly initialized with no frames traveling on the link and with sequence and request numbers equal to 0 at both ends of the link. Initialization is a general problem for protocols involving communication between two nodes, and arises, for example, in transport protocols and many special-purpose protocols. We shall soon see that initialization and disconnection are intimately related.

One might think that the problem of initializing an ARQ protocol is trivial—one might imagine simply initializing the DLC at each node when the link is first physically connected and then using the ARQ protocol forever. Unfortunately, if the link fails or if one or both nodes fail, initialization is more complicated. We first discuss the problem in the presence of link failures alone and then consider the case in which nodes can fail also.

2.7.1 Initialization in the Presence of Link Failures

It was shown in Section 2.4 that standard ARQ protocols operate correctly (assuming correct initialization) in the presence of arbitrary link delays and errors. Thus, even if the link fails for some period of time, the ARQ protocol could continue attempting packet transmission, and those packets not received correctly before the failure would be received after the failure. The difficulty with this point of view is that when a link fails for a protracted period of time, it is necessary for the transport and/or network layers to

IPR2025-00756, OnePlus Ex. 1013
Page 29 of 29