

1 Study Guide

Module 2 Study Guide and Deliverables

Readings: Online lecture material and Chapter 4 - Data Link Layer, Chapter 5 - Network and Transport Layers

Discussions: Chapter 4 discussion ends March 22, 6:00 AM ET
Chapter 5 discussion ends March 22, 6:00 AM ET

Assignments: Assignment 2 due March 22, 6:00 AM ET

Assessments: Quiz 2 due March 22, 6:00 AM ET

2 Chapter 4 - Data Link Layer

Introduction

In Chapter 3 you learned about the physical layer in the five-layer network model. In this chapter you will learn about the data link layer, the next layer up in the model. The physical layer deals with the representation and movement of individual bits from a computer and/or device to another directly connected computer and/or device in the network. The functions of the data link layer are:

- packaging bits into a frame
- flow control
- media access control
- error detection and correction
- addressing

This online material should be read in conjunction with the textbook as it provides clarification of material in the textbook as well as additional information.

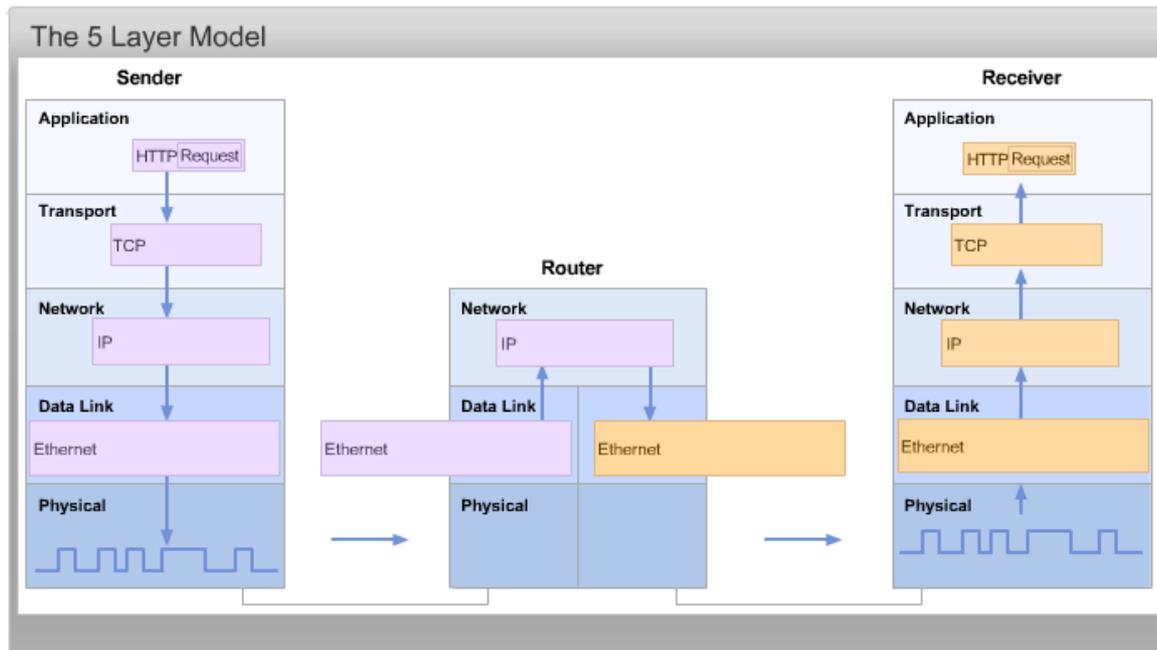
Objectives of this Chapter

In Chapter 3 you learned about the physical layer in the five-layer network model. In this chapter we will move up one layer in the protocol stack and you will learn about the data link layer. The functions of the data link layer are:

- packaging bits into a frame
- flow control
- media access control
- error detection and correction
- addressing

3 The Big Picture

In Chapter 1 you learned about the layering concept and the five-layer model used in networks. The following picture depicts the layers, their relationships, and the information that flows between the layers. This diagram shows a specific example of sending a HTTP request from the client to the server. The protocols being used at each layer are: HTTP (application), TCP (transport), IP (network), and Ethernet (data link).



As you learned in the online material for Chapter 1, each layer deals with a unit of information that is generally called a protocol data unit (PDU). These PDUs have specific names based on the layer. Unfortunately the author of our text does not use these names, but rather uses the term "packet" to refer to a PDU regardless of the layer. To be more exact, let us use the name specific to the layer according to the following:

Layer Name

- Application Message
- Transport Segment
- Network Packet
- Data Link Frame
- Physical just bits-no name

The data link layer sits between the network layer and the physical layer. On the sending side the network layer passes a packet down to the data link layer, which creates a frame by putting a data link header and trailer on the front and back respectively of the packet and passes the bits to the physical layer for transmission. On the receiving side the physical layer receives the individual bits and passes them to the data link to reconstruct the frame. The data link layer strips off the data link header and trailer and passes the data (packet) up to the network layer. At the physical layer the unit of transmission is the bit, so there is just a continuous stream of bits flowing over the media. At the data link layer these bits are turned into a frame, with a beginning and ending. Why do we need a frame?

The data link layer is basically responsible for getting information from a source to a destination without error. In order to accomplish this it must have a unit of data to transfer so that it can do the following:

- manage access to the media so that two sources don't transmit at the same time
- govern the flow of data to ensure the receiver is ready and able to receive data (flow control)
- ensure that errors are handled properly, which involves detection and recovery
- ensure the data reaches the proper destination based on its address

The frame, a PDU at the data link layer, is that unit of data.

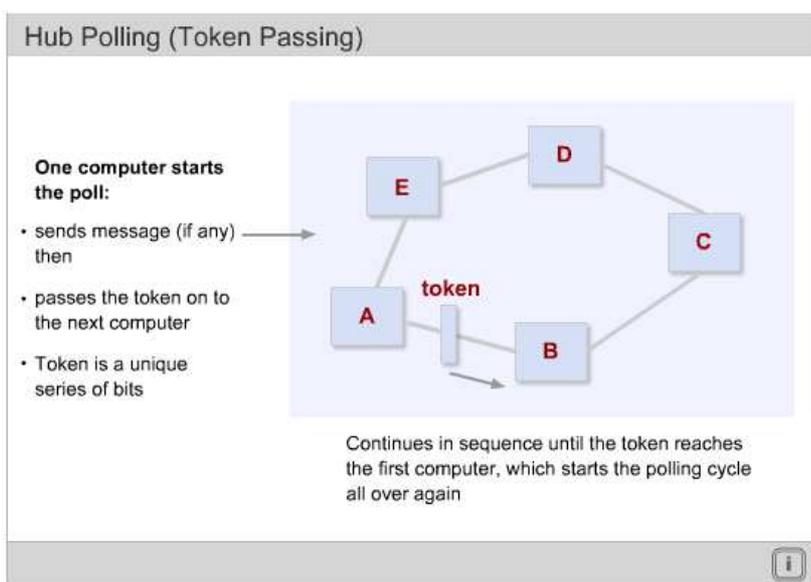
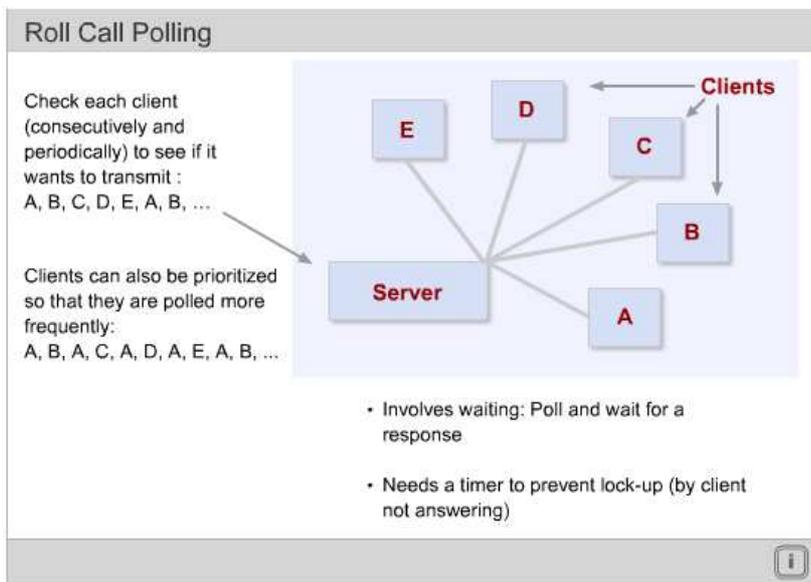
Another important concept is that of a protocol. A protocol is a set of rules to ensure successful communications within a layer across subsystems. In other words, the data link layer software on one computer, or data link layer device, will communicate with the data link software on another computer, or data link layer device, assuming that they are on the same network. When they communicate they exchange messages conforming to an agreed-upon format (syntax), have a specific meaning (semantics) and are sent

at the correct time or in the correct sequence. We will be discussing a number of functions performed in the data link layer protocol.

4 Media Access Control

As stated in our text, Media Access Control refers to the need to control *when* computers transmit. With full-duplex point-to-point connections this becomes a moot issue since the sender can send anytime because the circuit will always be free. However, with half-duplex point-to-point and multipoint connections the circuit may not be free and access to the media must be managed. We must ensure that two or more computers do not try to simultaneously transmit over the same circuit. There are two types of access control techniques- controlled access and contention access.

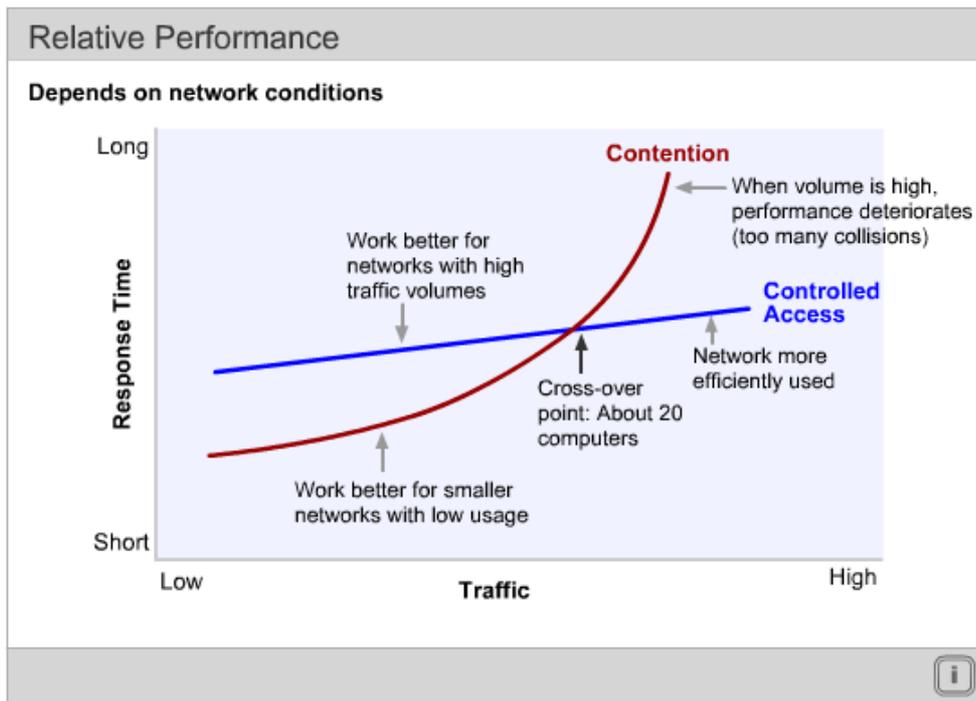
Controlled access is generally used by mainframes and by some LAN protocols such as token ring and FDDI. Polling, the process of sending a signal to a client that gives it permission to transmit if it wants to, is used. Polling falls into two categories: roll call and hub (token passing). They are illustrated by the following two diagrams.



Contention-based access is based on distributing the access control logic to every participant. Each computer will determine when it can transmit by listening to see if anyone else is transmitting. Since it takes a finite amount of time for a signal to travel from source to destination, and everyone listening to the circuit, it is possible to erroneously conclude that the circuit is free when in fact a transmission is in progress. In this case two transmissions will be done simultaneously and a collision will occur. Collisions require retransmission of all frames, which is wasteful. Contention-based access systems work very well on small networks that have low usage and controlled access better on high demand networks.

Ethernet is a contention-based protocol.

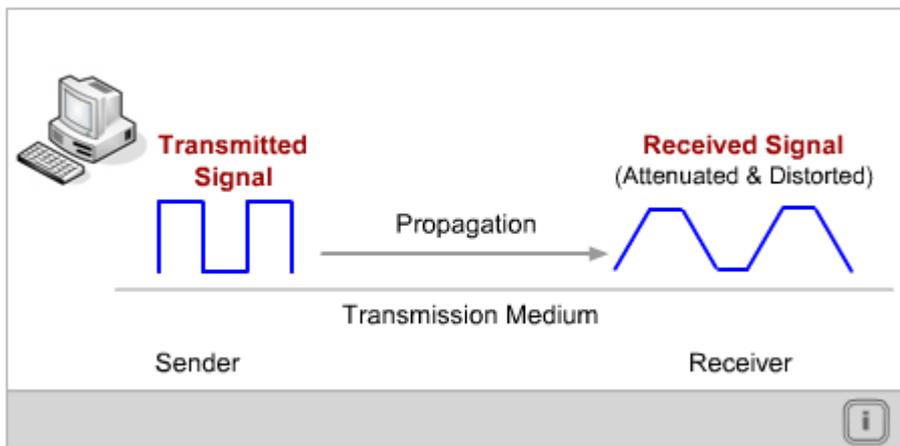
The relative performance of the two approaches is illustrated below.



5 Errors

Error Types

As discussed in the chapter on the physical layer, signals containing information content are transmitted via media from sender to receiver. The signals may be electricity, light waves, or radio waves. The media may be guided or unguided. The goal is for the receiver to recover from the signal the information as originally transmitted. Unfortunately, as a result of the propagation of these signals through the media, the signal is likely to be changed, as shown below.



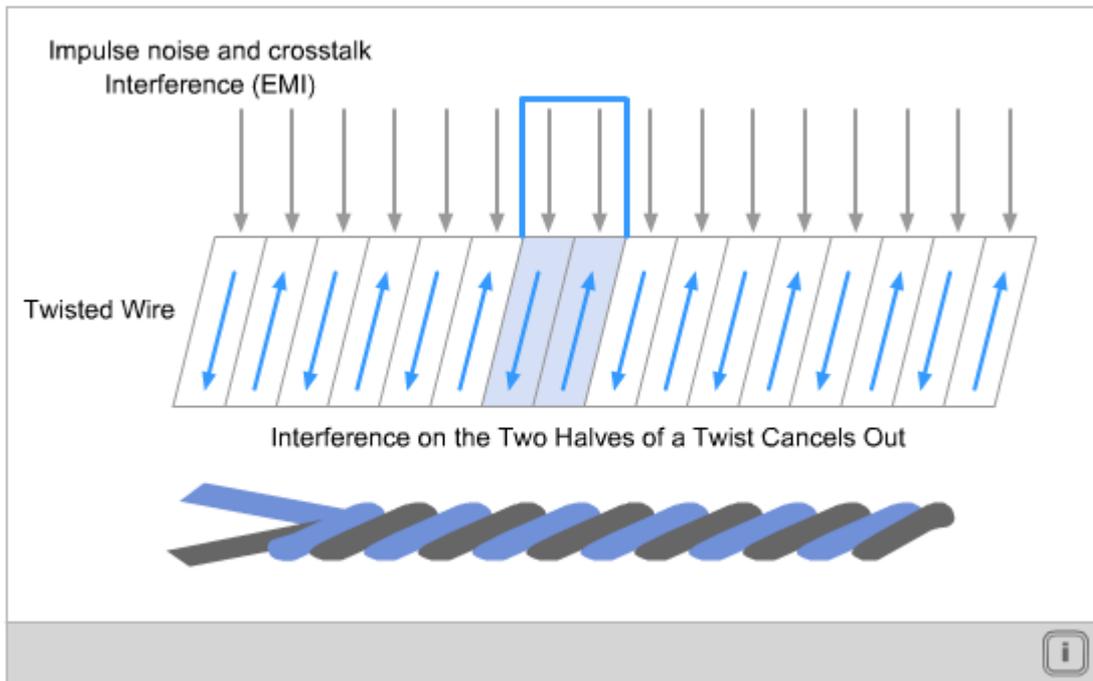
Signals are subject to both attenuation (a reduction in amplitude) and distortion (a change in waveform). These effects result from the transmission characteristics of the medium and cannot be eliminated, however they can be managed. This is part of the physical layer's job.

There are other impediments to recovering the data from the signal. Bits can change value during transmission/propagation. Bit errors fall into two categories-single-bit errors and burst errors. Single-bit errors are typically a result of white noise and are infrequent. Burst errors are more common and involve the loss of integrity of a group of adjacent bits, sometimes in the hundreds or thousands. Burst errors are caused by impulse noise. The types of errors, their causes, and their prevention are summarized in the following chart.

	Source of Error	What causes it	How to prevent it
More important	Line Outages		
	White Noise		
	Impulse Noise		
	Cross-talk		
	Echo		
mostly on analog	Attenuation		
	Intermodulation Noise		
	Jitter		
	Harmonic Distortion		

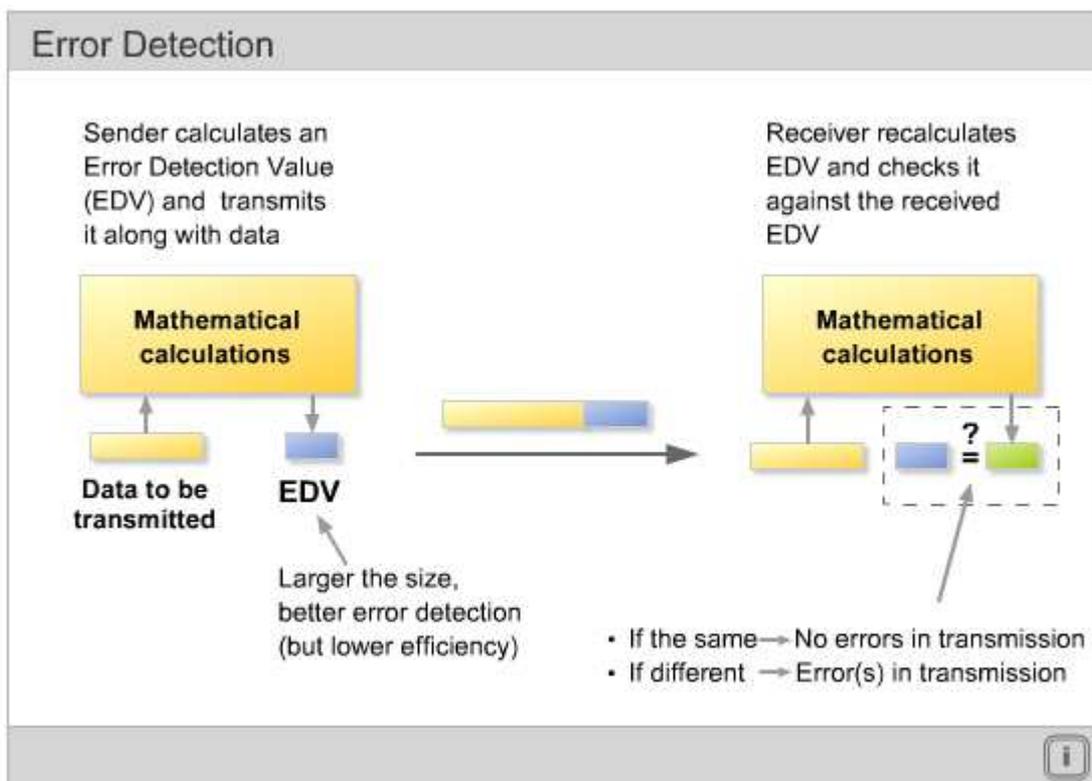
Please click each source of error for more information.

One very common technique, omitted from the textbook, for dealing with electromagnetic interference (impulse noise, crosstalk) is to twist the wires used for transmission. As a result of twisting the wires, the current flows from the offending signals induced in the media cancel each other out, and the interference is eliminated.



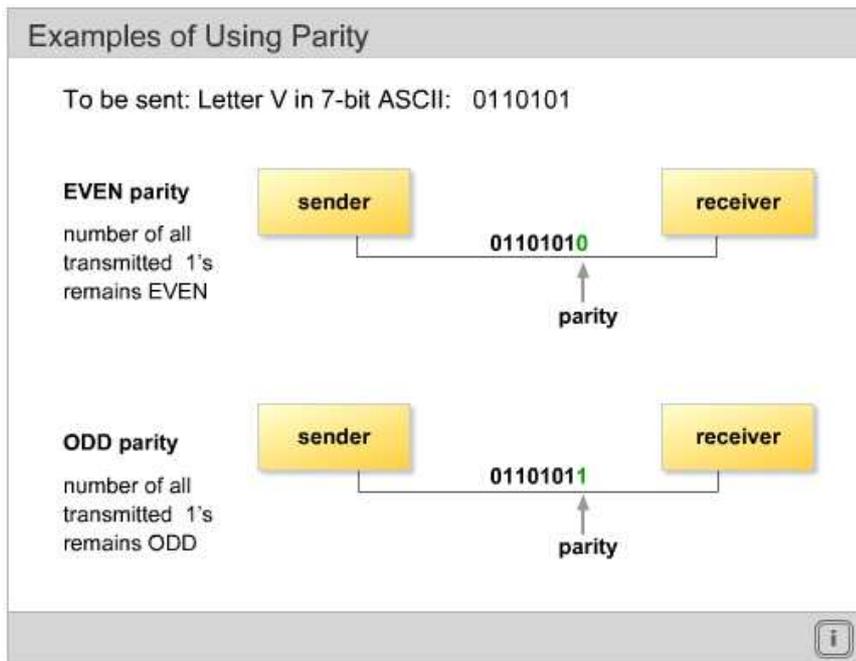
5.1 Error Detection

Since it is impossible to prevent all errors, we must be able to detect them when they do occur. All error detection algorithms are based on adding redundant information, called the error detection value, to the data stream. The error detection value is then used as depicted in the following diagram.



The algorithms for calculating the error detection value vary, but the basic approach is always used. Our text discusses parity checking, checksum, and cyclical redundancy check (CRC).

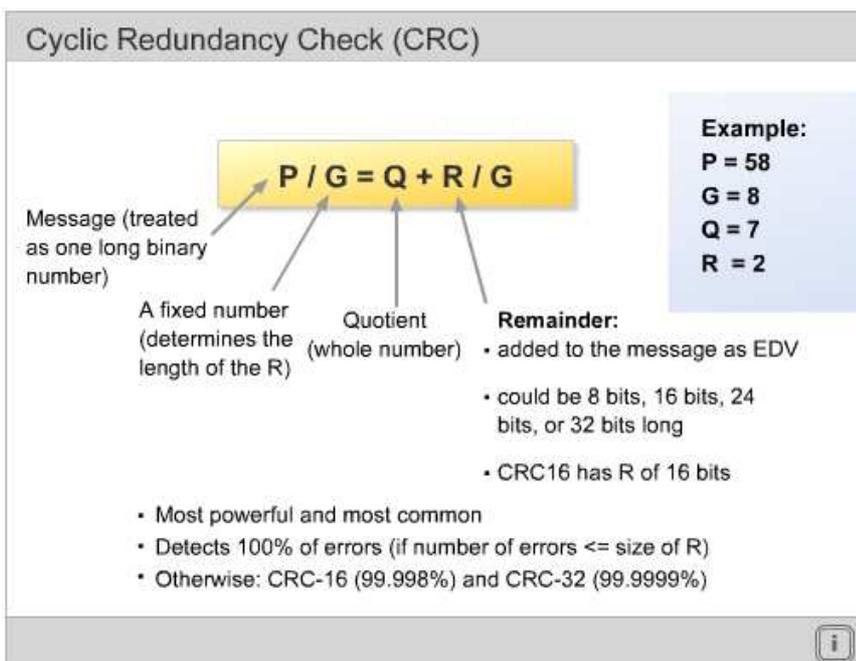
Simple parity checking is an old, simple technique that only has a 50% of catching errors. It involves adding a single bit whose value depends on the number of 1's in the data being transmitted.



It is particularly unsuitable for detecting error burst, the most common type of error in networking. Parity checking can be improved if done two dimensionally. One dimension is across a character, as described in the textbook. The second dimension is the same bit position across a number of characters. This approach increases the likelihood of detecting burst errors.

Checksums have a 95% change of detecting burst errors. Checksums are typically used at the transport layer since they are easy to calculate in software.

The error detection algorithm generally used at the data link layer is called the cyclical redundancy check (CRC). A slightly simplified description of the algorithm is depicted below.



The CRC algorithm is superior to the other algorithms because it is generally able to detect:

- All single- and double-bit errors.
- All odd numbers of errors.
- All burst errors less than or equal to the degree of the polynomial used.
- Most burst errors greater than the degree of the polynomial used.

The degree of the polynomial used relates to the version of the CRC used, i.e., 16 bit or 32 bit. For example, CRC-32 will detect all burst errors longer than 32 bits 99.99999998% of the time.

5.2 Error Correction

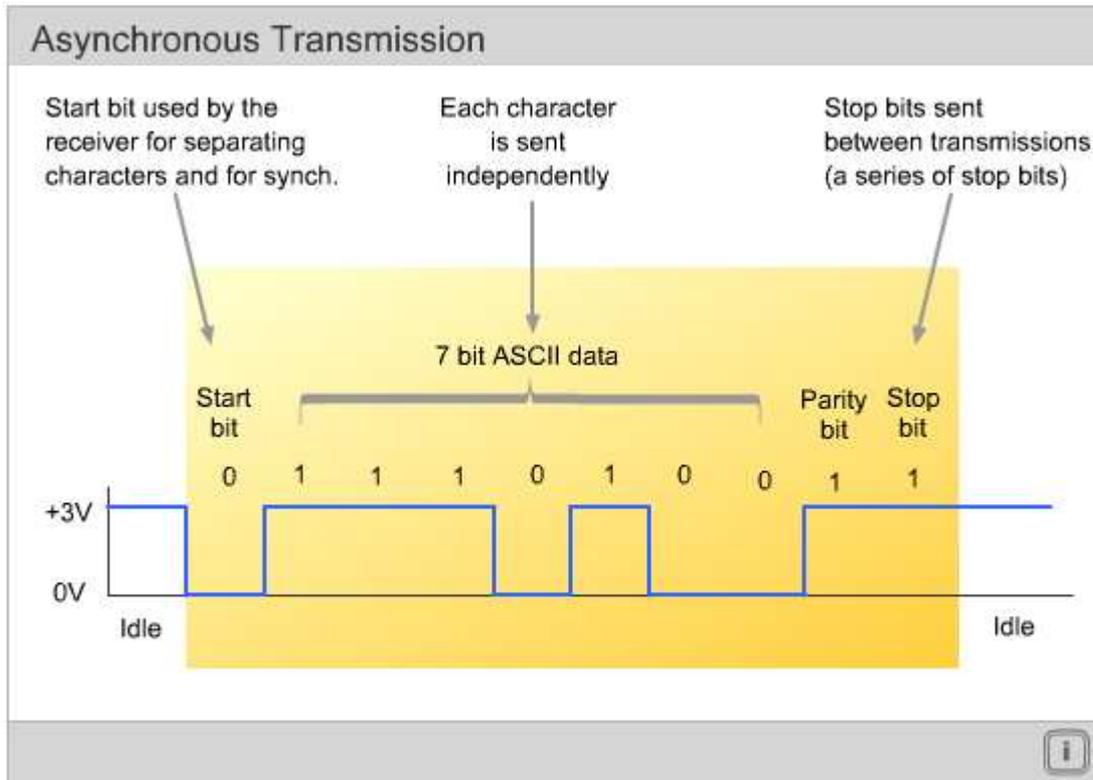
If an error is detected the data link layer has two choices—ignore the frame (throw it away) or correct the error. Interestingly, the most commonly used form of Ethernet II (a data link layer protocol) will, upon detecting an error, ignore the frame and throw it away. How can this be, you ask? In the chapter on the transport layer, you will learn that error correction, done end-to-end by the transport layer, causes the TCP segment in the frame in error to be retransmitted. As mentioned in chapter 1, the purpose of the data link layer is to get data to the next switch in the network, with a low probability of error. We don't need to be perfect--just good enough so that the host error checking is cost-effective. Protocols that ignore transmission error are called unreliable protocols.

Alternatively, a protocol that upon detecting an error then tries to correct the error is called a reliable protocol. Ethernet can be a reliable protocol, as discussed in the text, however this is not the most common way Ethernet is used. Normally error correction is accomplished by retransmission of the frame until it is received without error. The specific approach to accomplishing this is specified as part of the protocol being used. The two protocols discussed in the text, stop-and wait ARQ and continuous ARQ, require either a half-duplex or full-duplex circuit and involve the receiver sending an ACK frame back to the sender to confirm receipt of the frame sent. If the receiver does not receive the frame properly the sender can respond with a NAK frame, causing the sender to retransmit the frame. In the event the frame is never received or the ACK/NAK is lost, the sender will time out (let its time limit expire waiting for an ACK) and automatically retransmit the frame. In the event that the original ACK was lost, the receiver will receive a duplicate copy of the frame and must recognize this. Usually frames are numbered so duplicates can be detected by the receiver.

7 Data Link Protocols

Overview

In the previous section we looked at various techniques for error detection and correction. These techniques are embodied as part of a protocol specification. It is time to consider some specific data link protocols. Our textbook presents a fair amount of detail about asynchronous transmission. This is historically interesting, as it describes how dumb terminals, and other similar devices, communicated serially. I wouldn't really call these data link protocols, as a network really wasn't involved. However, asynchronous transmission does illustrate some important concepts that apply to data link protocols. First, it introduces the concept of synchronization. A function of asynchronous transmission is to parse a continuous bit stream into characters. Synchronization with the beginning and end of a character is done via start and stop bits respectively.



Second, asynchronous transmission error control is done via a parity bit. Third, although not mentioned in the text, flow control is accomplished by the receiver sending a XOFF character to the sender to force him to cease sending and then an XON character to resume sending.

The file transfer protocols mentioned are not data link protocols, but rather application layer protocols. They were originally popular for sending files over dial-up connections in the early PC days before networks as we know them today existed.

The synchronous protocols SDLC, HDLC, Ethernet, and PPP are important data link protocols in use today. They are much more efficient than asynchronous protocols since they package a number of bytes of data into one frame and then synchronize and do error control and flow control on the entire frame. These protocols also include source and destination addresses in each frame, which permits topologies other than point-to-. The most commonly used LAN data link protocol is Ethernet. It will be discussed in detail here as well as in the LAN chapter.

7.1 Ethernet

As mentioned in our text, Ethernet was conceived by Bob Metcalfe in 1973 and developed by Digital, Intel and Xerox later in the 1970s.

Four different Ethernet frame types exist today. They are:

- Ethernet II (derived from the DIX “standard” - the original Ethernet before it became standardized by the IEEE. This is the simplest form of Ethernet and is an unreliable protocol. This still commonly used and will be described in detail
- Non-standard Ethernet 802.3 (raw Ethernet) - originally created by Novell for use with the IPX protocol and latter standardized by the IEEE.
- Ethernet 802.2/802.3 LLC/SNAP - allows a high degree of flexibility for proprietary protocols
- Ethernet 802.2/802.3 LLC (SAP) Frame includes 3 types:
 - Type 1 - unreliable packet exchange
 - Type 2 - reliable, stateful, connection oriented
 - Type 3 - reliable, connectionless

The textbook discusses Ethernet 802.3 type 2 (standard 802.3ac) in great detail including how it uses the control field in the LLC header to achieve reliable communication. Since Ethernet II is the most commonly used form of Ethernet the description of the frame header for Ethernet will follow this standard. See the textbook for a description of the 802.3ac frame.

Ethernet II is an unreliable protocol used primarily in local area networks (LANs), although in more recent times it has been extended to Metropolitan Area Networks (MANs). Some aspects of the protocol will be discussed here, while others, primarily media access control, will be discussed in Chapter 6.

Frame Format - Ethernet II

Ethernet frames are variable in length and include a header, data, and a trailer. The header consists of:

- Preamble - A 7-byte pattern consisting of alternating 0s and 1s (1010101....) This field is used for synchronization purposes, so the receiver can know when a bit starts.
- Start of Frame Delimiter - A single byte containing 10101011. Note the receiver will know when the byte ends, and the remainder of the frame starts, by detecting the two 1s in sequence. This field is used for frame synchronization.
- Destination Address - 48 bits (6 bytes) known as a MAC address. This address is unique to an Ethernet NIC.
- Source Address - 48 bit (6 bytes) known as a MAC address.
- Data Length Field - 2 bytes, the length of the data field.

The data portion of the frame is

- Data/Pad field - 46-1500 bytes. The data would be the packet passed down from the network layer. The data field must be at least 46 bytes long. If it isn't, it is padded to be at least 46 bytes.

The trailer portion of the frame is

- Frame check Sequence - Error checking using CRC-32.

Protocol Characteristics

As previously mentioned, Ethernet II is an unreliable protocol. No ACK or NAKs are sent; the sender assumes its frames have been received properly. However, the receiver does check for errors using the CRC included in the trailer. If an error is detected the frame is thrown away. Addressing is done via the use of 48-bit MAC (media access control) addresses. They are unique to a NIC, and identify both the sender and the receiver. These addresses are used to switch the frame to the proper destination. They are also used by the receiver to selectively accept frames destined for the receiver and ignore all others that it might receive. There are also specific addresses that allow frames to be received by a specific group of machines (multicasting) or all machines (broadcasting).

In recent times the Ethernet protocol has been extended to include virtual LANs (VLANs). A virtual LAN is a scheme for partitioning a number of machines in a LAN into groups of machines, called VLANs. Some transmission of frames within the LAN is restricted to go only to machines within the VLAN of origin, rather than to all machines. This helps reduce congestion in the network.

8 Transmission Efficiency

We often describe the speed at which our networks communicate in terms of data rate, or bits transferred per second. This is a rather gross metric in that it does not accurately portray the transference of information bits, which is really what we are interested in. It does not distinguish between information bits versus overhead bits, nor does it take into account the lost transmission time due to error control. The section on transmission efficiency in our textbook does a very reasonable job in describing transmission efficiency. To illustrate the concepts presented in that section a problem and its solution is presented.

Problem

What is the efficiency if a 100-byte file is transmitted using Ethernet? A 10,000-byte file?

Solution

A 100-byte file would be transmitted in one frame, since a frame can contain up to 1500 bytes of data. Thus,

transmission efficiency = number of information bits / total number of bits transmitted

number of information bits = 100 bytes * 8 bits/byte = 800 bits
total number of bits transmitted = total number of information bits +
total number of overhead bits

total number of overhead bits = bits in header + bits in trailer

= 22 bytes * 8 bits/byte + 4 bytes * 8 bits/ byte
= 208 bits

transmission efficiency = 800 / (800 + 208)
= 79.4%

A 10,000 byte file is a bit (no pun intended) more complicated. Since the maximum amount of data that a single frame can carry is 1500 bytes, more than one frame will have to be transmitted.

the number of frames = amount of data / size of frame
= 10,000 / 1500
= 6.666 frames
rounding up, this gives us
= 7 frames

transmission efficiency = number of information bits / total number of bits transmitted

number of information bits = 10,000 bytes * 8 bits/byte = 80,000 bits
total number of overhead bits per frame = bits in header + bits in trailer
= 22 bytes * 8 bits/byte + 4 bytes * 8 bits/ byte
= 208 bits

Since there are 7 frames transmitted, the total number of overhead bits = 7 * 208 = 1456
transmission efficiency = 80,000 / (80,000 + 1456)

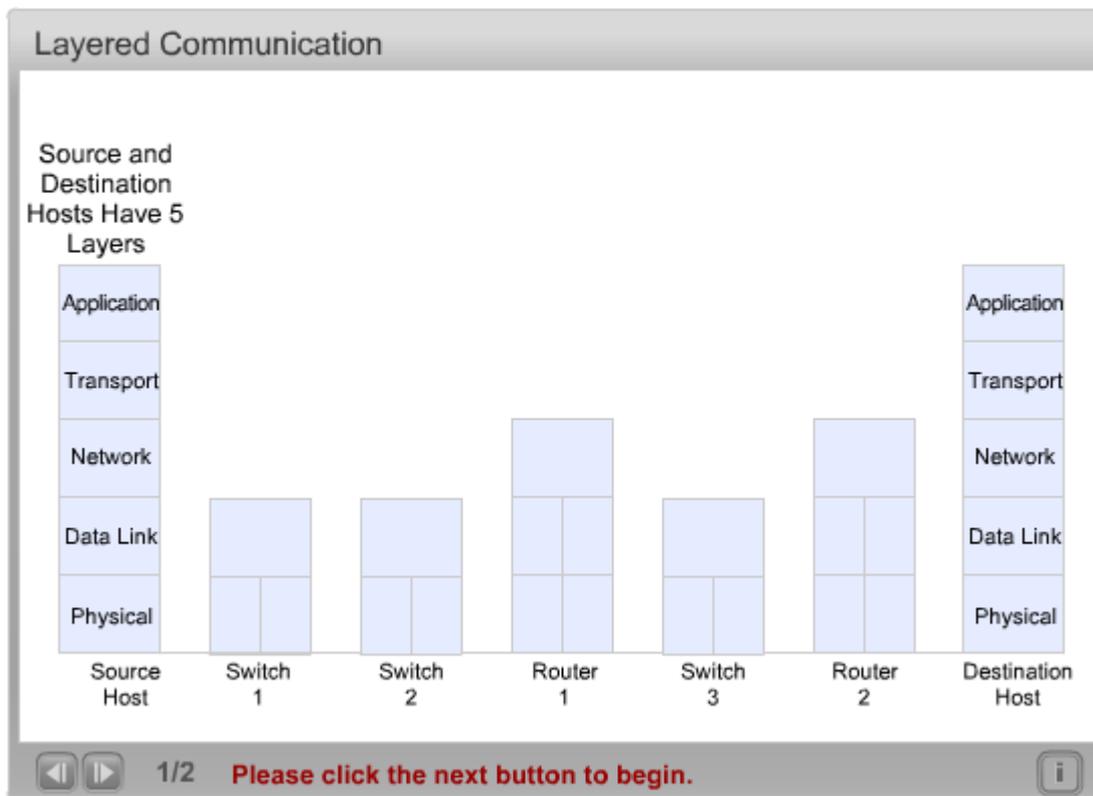
= 98.2 %

9 Data Link Layer Devices and Scope

In this chapter we considered the role of the data link layer in ensuring successful communication within a network. We saw that the data link layer is responsible for constructing frames, flow control, media access control, error detection and error correction and addressing. Two important questions to address are what devices work at the data link layer and what the scope of the data link layer is.

9.1 Data Link Layer Devices

In the chapter on the physical layer we learned that all computers connected to a network have at least one network interface card (NIC) and that this card deals with the physical layer. This card typically also implements a part of, or the entire, data link layer. So, for instance, an Ethernet NIC would implement the Ethernet data link layer protocol, as well as a specific physical layer standard. So all computers include data link layer processing. We also learned that repeaters and hubs are physical layer devices, which means that the highest level at which they operate is the physical layer (that is, they work with individual bits). At the data link layer, the primary data link layer device is the switch. A switch will receive bits on one port, assemble the bits into a frame, examine the contents of the frame header for a destination address, determine what port to forward the frame to via its switching table, and finally send the out frame. That is as far as a switch processes. It doesn't look any deeper into the frame. It doesn't look at the IP header, TCP header or application message. The following figure depicts this:



This figure assumes that hosts have only a single NIC, which is the norm. In the chapter on the transport and network layers you will learn about routers. As you can see from the picture, they operate at the network layer.

9.2 Data Link Layer Scope

Once a specific data link layer is chosen how far does it extend in the network? What is its scope? The layered communication picture helps us answer this question; look at the way it depicts the layers. A single rectangle on a host or device at a given layer means that there is only a single protocol used at that layer. For instance the DL on the source and destination hosts applies to all communication from or to that host. Also, the data link layer for the switches applies to all ports on that switch. The router, since it has two rectangles at

the data link layer could have two different data link layers, one for each interface. So, starting with the source host, the same data link layer protocol must be used from the source host through switch 1 and 2 and up to and including the interface on router 1. A different data link layer protocol can be used from router 1 through switch 3 to router 2. And yet another data link protocol can be used between router 2 and the destination host.

Another way to look at it is that the scope of a data link layer refers to everything connected to a router interface, either directly or via one or more switches.

10 Chapter 5 - Network and Transport Layers

Introduction

In the Data Link Layer, we were concerned about reliably moving data between two systems. If every computer were directly connected to every other computer, that would be all we would need. But it is not only expensive but not really feasible to connect that many wires to a single system. So we introduce the Network Layer and systems dedicated to relaying (routers). This allows us to have fewer wires coming into one box, and allows us to use fewer lines and make better use of them. BUT, we also have to have a way to figure out which packets go where, and how to get them from one host to another through this network of relays.

As we learned in the first chapter, the seminal innovation in networking was realizing that because the hosts would never completely trust the network and would always check to make sure all of the data arrived safely, the network didn't have to be perfectly reliable. The Transport Layer provides this ultimate checking to ensure that all of the data arrives safely. It turns out that no matter how much memory we put in routers, there will always be conditions that cause sufficient queuing delay to force retransmissions and generate congestion. No matter what we do, we can't make the network perfectly reliable, so it is a good thing that hosts don't trust it!

For the applications, the Transport Layer is the distributed Inter-Process Communication, IPC, facility that it uses to communicate with other applications elsewhere in the network. The purpose of the Transport Layer is to make IPC look as much like local IPC as it can.

Objectives:

In this chapter, we will look in greater detail at how the Network and Transport Layers work.

- Understand how TCP (Transmission Control Protocol) handles loss and duplicate detection, flow control and congestion control, and how it is similar to and different from HDLC (High Level Data Link Control)
- Understand the role IP (Internet Protocol) plays and why it may not be a protocol
- Understand the critical role of naming and addressing in networks.
- Understand the basics of how routing is accomplished in networks.
- Understand the role played by DNS (Domain Name Service) and how it works.

11 The Transport Layer

The Transport Layer is the key layer in the architecture in that it provides the distributed IPC mechanism to applications. Its primary purpose is to ensure the end-to-end reliability requested by the application. We will see many of the same mechanisms in transport protocols like TCP that we saw in the Data Link Protocols that performed error and flow control. The Transport Layer differs from the Data Link Layer in that it operates over a different scope (host-to-host as opposed to device-to-device). Look again at the figure on scoping in Chapter 1 Section 1.8. Consequently it encounters different error characteristics and different traffic characteristics. So while the mechanisms are similar, the policies associated with the mechanisms are different. For example, a data link protocol encounters bursts of errors and so uses a CRC polynomial to detect them. A transport protocol encounters single bit errors and so uses a checksum.

The Transport layer facilitates exchange of data between applications. It uses TCP and UDP protocols for transmission. Most TCP/IP applications use TCP for transport layer. TCP provides a connection (a logical association) between two entities to regulate flow and check errors. On the other hand, UDP (User Datagram Protocol) does not maintain a connection, and therefore does not guarantee delivery, preserve sequences, or protect against duplication.

The TCP/IP architecture concentrates on what has traditionally been called the Network and Transport Layers, and says very little about what is above or below those. The primary applications developed for it by the traditional Internet community are Telnet, FTP, SMTP, and SNMP. The world wide web protocols were really developed outside the traditional community.

TCP carries the source and destination port ids. Together these constitute a connection identifier so that different TCP connections can be distinguished. They are assigned locally by the host as unique identifiers between the application and the operating system. They are only unambiguous within the host that assigns them. As an expedient (kludge) in the early days of the 'Net, specific low numbered port ids were assigned to specific application protocols, known as well-known sockets or well-known ports. For the set of application protocols listed above, the application and the application protocol are one and the same; also, there is one per system. For the web, this is not the case.

11.1 TCP Segment header (TCP Protocol Data Unit)

In the following figure, a TCP header is shown which has the following fields:

- Source port (16 bits)
- Destination port (16 bits)
- Sequence number (32 bits)
- Acknowledgment number (32 bits)
- Header Length (4 bits) (in 32 bit words)
- Reserved (6 bits)
- Flags (6 bits) : URG, ACK, PSH, RST, SYN, FIN
- Credit (16 bits)
- Checksum (16 bits)*
- Urgent Pointer (16 bits)
- Options (variable)

** note figure 5.2 in the text labels this field "CRC-16" in error*

TCP is unique among most protocols in having a single header format. This was done initially to save code in the implementation. Because routing in the Network Layer does not guarantee that all packets will take the same path, out of order arrival is possible.

The sequence number is used to detect lost or duplicate messages and to put messages in order. The protocol uses a sliding window similar to that found in Data Link protocols except here the window size is not fixed but can vary. The acknowledgement (ACK) field sets the sender's left window edge. The Credit field added to the ACK field is the amount of data that can be sent (right window edge). A retransmission queue is maintained of all PDUs sent, and if an ACK does not arrive in time, PDUs are re-transmitted. An ACK means that all data before the ACK number has been received and the sender can delete them from its retransmission queue.

The Urgent Pointer is a rudimentary means of out-of-band signaling no longer necessary. However, there are rumors that some TCP implementations use them to mark boundaries important to the application.

The Flags have the following meaning:

URG - means the value in the Urgent field is valid.

ACK - means that the value of the ACK field is valid.

PSH - tells the receiver to deliver immediately to the application

RST - reset the connection

SYN - used in connection establishment

FIN - used in connection termination.

Because TCP is using feedback mechanisms, it must use a 3-way handshake to ensure that initial state is properly synchronized. In the case of TCP, the synchronization is symmetrical.

Since the late-1980s TCP has also done congestion control. In 1986, the Internet experienced congestion collapses as the number of directly connected hosts whose access to the Internet was flow-controlled were supplanted by LAN based systems with no such access control, congestion became a daily occurrence. It can be shown that no matter how much memory a router has, queue lengths will have a tendency to grow indefinitely once traffic exceeds about a 30% load factor. This means that some PDUs will be sufficiently delayed that TCP retransmissions will be triggered causing queue lengths to become even greater.

As a stop gap measure, congestion control was introduced into TCP. TCP was designed to do flow control to keep the sender from over running the receiver. However, congestion is a problem somewhere between the two ends of the TCP connection. This stop-gap will suffice until a killer application arises that does not use TCP.

The TCP sender starts sending slowly doubling its send window as long as it does not detect congestion. When it does, it backs off, cutting its congestion window and then starts increasing again. Notice that TCP congestion avoidance works by causing congestion.

This is a simple solution and has worked well with the "best-effort" policy of the Internet. However, as demands have increased for other than best-effort, the TCP congestion control solution presents problems.

12 IP Addresses

IP addresses are used as the addresses, while DNS names and URLs are as close to application names as we get.

IP addresses indicate where things are. URLs name applications that one wants to access. DNS maintains the mapping between URLs and IP addresses. That is the easy part. The whole topic of network naming and addressing is a very subtle and difficult topic, which is really only understood by a handful of people on the Internet. In this course, we will only have to scratch the surface of this topic. Each computer on a network requires a unique address. Likewise, each application requires a unique address within the computer to allow support for multiple applications (service access points, or SAP).

13 The Network Layer

The primary purpose of the network layer is routing and resource allocation. Traditionally in an IP network, resource allocation has followed a “best-effort” policy for all traffic. However, with new applications coming into the network that require different levels of service, this is changing. We will take up this problem below.

The major prerequisite to routing is naming and addressing. This is one of the most difficult aspects of networks and one that it is crucial to get correct. Very early on, it was understood that three independent concepts were involved:

1. Application names, which indicate *what* we want to communicate with,
2. Addresses, which indicate *where* it is, and
3. Routes, which indicate *how* to get there.

This view was extended in 1982 by Jerry Saltzer, who correctly pointed out that a full network architecture requires:

1. application names,
2. node addresses, and
3. point of attachment addresses.

Application names are location-independent (so that they can move without changing their names), while node addresses are location-dependent. Points-of-attachment name the interfaces the node has with the network. (Point-of-attachment addresses are sometimes Data Link Layer addresses.)

13.1 Routing

In theory, routes are sequences of node addresses. From the route, we determine the next hop, i.e. where the router is to forward packets next. Once a route is calculated, a forwarding table is constructed for each router indicating the next hop for each destination address. The mapping of node addresses to point of attachments is used to distinguish multiple paths to the same next hop. (Sometimes for redundancy or because there may be more bandwidth between two points than can be carried by a single line.) The routers also must maintain the mapping of all point of attachment addresses to their nearest neighbors. From this, the router can pick the path to the next hop (when there is more than one line between the two).

That is how it is suppose to work.

However, for historical reasons the Internet only has point of attachment addresses. IP addresses name the interface, not the node. The reason IP addresses name the interface is because host addresses in the original ARPANet were the port numbers of the switches. The Internet doesn't really have application names. Instead it has well-known sockets that indicate what protocol is being requested. (Both of these are holdovers from short-cuts made in the early 70s.) Hence, routes are sequences of point of attachment addresses. Since for all early Internet applications (Telnet, FTP, mail) there was only one per host; there was not an immediate need for application naming and well-known sockets were created as a short cut. None of this was ever fixed. Consequently, multi-path routing is difficult, and true multihoming is impossible. Mobility is cumbersome, and a multitude of other addressing problems plague the Internet today. So much so, that several years ago the Internet Architecture Board created a special Name Space Research Group to look at the problem. Politics, inertia, and lack of understanding lead to essentially no conclusions. Today this is at the center of a crisis that is causing router table size to grow beyond the limits of hardware for the foreseeable future.

Basically, the Internet has half an addressing architecture.

This means that on the Internet, routes are sequences of points-of-attachment. This makes multi-path routing cumbersome. (In large networks, it is not uncommon for the bandwidth demand between two routers to exceed the capacity of a single fiber or wire. In this case, multiple paths will be necessary between the two routers. Consequently, various workarounds are necessary to do the required load leveling.)

Router table size increased because addresses were not location-dependent. In the flood of requests, addresses were assigned without regard to where they were. Two addresses numerically close to each other could be thousands of miles apart. As an analogy consider the post office: If you mail a letter in Newton MA

USA to 40 Rue de Rivoli, Paris France. The post office in Newton merely looks at the country and puts the letter in a bag going east. They don't have to know where rue de Rivoli is or even where Paris is. But in the Internet, one must trace a route from 40 rue de Rivoli all the way back to where the sender is, just to determine what bag to put it in! Consequently, every router must calculate such a route for every address it may see. For routers in the backbone of the Internet, this can be a very large number and as the Internet grows it gets bigger.

This means that it is very difficult to support multi-homing, i.e. multiple connections to the network. Since the address names an *interface*, the routing algorithms can't know that two interfaces go to the same host. Until recently, IP addresses were not location-dependent but were assigned arbitrarily. Beginning in the early 90s, this was changed so that IP addresses were assigned by providers. This allows some efficiency in routing, but also implies that when a user changes providers she will have to re-number her network with new IP-addresses. While URLs look like application names, they only work with HTTP.

14 IP Addresses

IP addresses are used as the addresses, while DNS names and URLs are as close to application names as we get.

IP addresses indicate where things are. URLs name applications that one wants to access. DNS maintains the mapping between URLs and IP addresses. That is the easy part. The whole topic of network naming and addressing is a very subtle and difficult topic, which is really only understood by a handful of people on the Internet. In this course, we will only have to scratch the surface of this topic. Each computer on a network requires a unique address. Likewise, each application requires a unique address within the computer to allow support for multiple applications.

14.1 IPv4 Header

An IPv4 header has the following fields:

- Version (4 bits)
- Internet header length (4 bits)
- Type of Service (8 bits)
- Total Length (16 bits)
- Identification (16 bits)
- Flags (3 bits)
- Fragment Offset (13 bits)
- Time to Live (8 bits)
- Protocol (8 bits)
- Header Checksum (16 bits)*
- Source Address (32 bits)
- Destination Address (32 bits)
- Options (variable)
- Padding (variable)

** note described in figures 5.3 and 5.4 in the textbook as a CRC, this is a mistake.*

The Internet Protocol is more a common header than a protocol. Its primary purpose is to carry the source and destination addresses of the PDU. Besides the addresses, the most important field in the IP header is the Time To Live field. Anomalies in routing can cause loops to develop that can cause a PDU to remain in the network for long periods (hours or even days). The TTL field is a hop count. At each router it is decremented, when the count reaches zero, the PDU is discarded whether or not it has reached its destination. Other important fields deal with fragmentation. Different networks have different maximum PDU sizes (MTU). When a PDU encounters a network with a smaller MTU, the fragment offset and last fragment fields are set. The PDU is not reassembled until it reaches the destination. Some PDUs should not be fragmented (such as boot records), these have the "do not fragment" (DNF) flag set.

15 Internet Addressing

An IP address consists of a network ID and a host ID. Different hosts belonging to the same network share the same network ID but different host ID's. IP addresses identify interfaces, *not* hosts. If the host has a single interface to the network, the interface and the host are the same. Since this is often the case, it is easy to think of them identifying the host. Be careful, it is not true all the time. Hosts with different network ID's belong to different subnets on the network and can communicate with each other remotely through a router or default gateway. The IPv4 address is 32-bits long composed of network and host identifiers. The network addresses are classified into different classes allowing a mix of network sizes.

Network and Host Ranges

Class	Network Address Range	No. of Hosts
A	1.0.0.0 to 126.255.255.255	16,777,216
B	128.0.0.0 to 191.255.255.255	65,536
C	192.0.0.0 to 223.255.255.255	256
D	224.0.0.0 to 239.255.255.255	Multicast address

The range of the network and host identifiers depends on the class type. For example, the total number of addresses and hosts in class C is 2,097,152 (2^{21}) and 256 (2^8) respectively.

The addresses are written in both binary and decimal format. An example of an IP address in binary form would be: 11000001.00101010.00111111.00000110. To convert the IP address from binary to decimal form, each of the four 8-bit numbers (octet) is converted as follows:

$11000001 = 128 + 64 + 1 = 193$
 $00101010 = 32 + 8 + 2 = 42$
 $00011111 = 16 + 8 + 4 + 2 + 1 = 31$
 $00000110 = 4 + 2 = 6$

So in decimal form, the above IP address is: 193.42.31.6

16 Subnetting

Subnetting is used to divide the address of an IP host into smaller subnets (subnetworks) within an organization. The subnets are defined within the host ID. For example, Class C would have 256 possible IP addresses (0-256 hosts). Two addresses are used for broadcast. That leaves 254 possible hosts (193.10.30.1 - 193.10.30.254). These hosts can be divided into smaller subnets using subnet masks. Subnetting is used for sizing the number of networks and the associated hosts. The number of subnets depends on the subnet mask.

The following diagram shows an example of class C subnetting, dividing the last octet equally into subnets and hosts (2 to the 4th power (16) - $2 = 14$ subnets and $16 - 2 = 14$ hosts each). The subnet mask is 255.255.255.240.

17 Internet Routing

Internet routing is the process of moving a packet from source to destination. The device that performs this function is called a router. The packet might get routed through several intermediate devices (routers) before it reaches the targeted destination. As part of the process each device analyzes a routing table to determine the best path.

In routing there are two key concepts that need to be distinguished:

- **Routing information** - which provides information about the topology and delays on the Internet
- **Routing algorithm** - which determines the decision mechanisms for routing packets

Routing is a distributed computation. Routers exchange information about the connectivity with each other as well as information on the load they are experiencing (usually measured as queue length). This information is used by each router to compute routes through the network. From this information the router constructs a forwarding table. The forwarding table maps the IP destination address of an incoming packet to the next hop it is suppose to be sent to. So in fact, routers calculate routes to determine the next hop. If all has gone well, the next hop will have reached the same conclusion and its forwarding table will cause the packet to be forwarded along the same route. If not (the equivalent of each router thinking it knows a better way to get to the destination), the packet may take a roundabout route to the destination or even loop.

An Autonomous System (AS) is characterized by a set of routers and networks managed by single organization. These routers are exchanging information via a common routing protocol. They are interconnected and there is a path between any pair of nodes. The concept of an Autonomous System has been grafted on to the routing algorithm to provide a basis for aggregating addresses and making the routing calculation tractable.

17.1 Routing Protocols

The routing protocol that passes routing information between routers within an AS is called

Interior Routing Protocol; the routing protocol that exchanges routing information between routers in different ASes is called **Exterior Routing Protocol**.

Border Gateway Protocol (BGP)

BGP is the most common exterior routing protocol. It establishes BGP sessions using TCP connections between pairs of gateway routers for exchanging routing information.

Open Shortest Path First (OSPF)

OSPF is the most common interior routing protocol. It stores information about each link within the AS. Link information may include: link delay, throughput, cost, etc. All these factors determine the routing cost function. Each router has a database that has information about all the links. Whenever there is an update on the status of any router's link, the router updates its table and passes this information to all the other routers. Routing is calculated and decided based on the predetermined cost function.

In the early 'Net, a class of routing protocols were used based on what are called the distance-vector routing algorithm. With these algorithms, information is only exchanged with a router's nearest neighbors. This means that the algorithm scales well but it does tend to converge slowly when there is a change in the network (congestion or a temporary line outage or router crash). Another class of routing algorithms called link state which converge much more quickly, but require all routers to have complete information on the network connectivity. Clearly, this will not scale and would be impractical for the whole Internet. So the strategy adopted was to use link state algorithms within Autonomous Systems (AS) (provider or corporate subnets), i.e. interior gateway protocols and distance vector between AS, i.e. exterior gateway protocols.

In both cases, the routing algorithm executed by each router calculates a least cost route (based on some metric) all the way from the destination of this packet to this router. It then uses this information to create a forwarding table, which tells it which router to forward packets to next, i.e. the next hop. Yes, the entire route is calculated just to figure out the next hop. At the next hop, the whole process is repeated. Just because one router calculates a route to the destination, this does not mean the packet will follow that route. The next router may have different information in its routing tables. However, in practice most major ISPs use static routes, because they find that it is too difficult to achieve stable behavior with OSPF.

Enhanced Interior Gateway Routing Protocol (EIGRP)

EIGRP is a proprietary interior routing protocol developed by CISCO. Contrary to what is said in our textbook, EIGRP is an advanced [distance-vector routing protocol](#) (not a dynamic link state protocol). EIGRP is an enhanced version of CISCO's original interior gateway routing protocol (IGRP).

18 QoS

As we have said earlier, the Internet was designed as a “best-effort” network. The whole concept of connectionless datagrams is counter to the idea that some packets will be treated differently. As the popularity of the Internet grew, the desire to support applications that required something other than best-effort also grew. This created quite a quandary that goes on to this day. Early on in networking, the telephone companies developed networks that reflected their tradition of connections, e.g. X.25, Frame Relay, etc., while the research community adopted connectionless, and there have been some horrific arguments between them. Flame war doesn't come close to characterizing it! This tended to create two well-defined and well-separated camps: each certain that the other was completely wrong, simply didn't get it, and would eventually die out. When the need for QoS began to arise, the connection camp saw their chance to achieve the upper hand. However, their proposals were still too tightly tied to the perceived requirements of traditional voice and too tightly tied to a deterministic paradigm and yielded expensive, complex solutions: ATM and more recently MPLS. Meanwhile, the connectionless camp really had no new ideas but Moore's Law and the plummeting cost of bandwidth meant that it was easier to throw bandwidth at the problem than engineering...and less dangerous. (By throwing bandwidth at the problem, it is certain there will be some improvement. Traffic engineering these networks is still enough of an “art” that there is much less certainty in the outcome. There is a real chance that things could get worse. Hence, many organizations opt for the easy, more certain course of action.) Hence, neither the connection nor connectionless camp achieved the kind of decisive “victory” they wanted.

DiffServ is the most widely adopted QoS method in the Internet. But as you can tell from the description, DiffServ is fairly loose and the classes of service are based on existing applications and determined by the provider. The biggest problems with DiffServ as well as other proposals are scaling it to operate in large networks and achieving QoS agreements across different providers. (The flexibility and lack of specificity in the definition of classes of service that allows DiffServ to be reasonably implemented also makes providing a given level of QoS across providers difficult. But there is another factor as well. In the traditional phone system, different phone companies served different regions of a country or different countries. They were more willing (not necessarily much more) to share information about ensuring consistent quality because they weren't really in direct competition. Providing the customer with a good experience benefited all of them. But in the networking business today, everyone serves every area (for the most part). They are all in competition and all believe that they will eventually displace the other contenders. Hence, there is little or no incentive to share information that would allow creating good QoS across providers. In fact, if it is bad enough (and it is always the other guy's problem) perhaps the customers will consolidate their business on a single provider!)

With DiffServ, it may be difficult to support new applications and new service classes. Furthermore, the case remains that bandwidth is sufficiently cheap, simply over-provisioning the networks so that network utilization remains very low (< 25%) is still quite common. However, providers feel that if the only thing they can do is to order more bandwidth, they have little control over their own destiny. And, as we have pointed out earlier, over-provisioning a best-effort service is a commodity business and commodity businesses have low margins. Profits and growth and “new products” are hard to come by in a commodity business, especially for companies with 100 years of experience at being a monopoly.

19 DNS - Domain Name System

There are four main components of a DNS system: Domain name space, DNS databases, Name Servers, and Resolvers.

- Domain name space is defined as a tree-structured space that identifies all Internet resources. It is strictly hierarchical, and it contains the names of every accessible interface over the Internet.
- DNS databases are stored as distributed databases. This means that not all the data about the host is held at one location; instead, there is a hierarchical correlation between various databases that together form the globally connected DNS database

There are two programs that are used in database searches: Name servers and Resolvers.

- Name servers are server programs that hold information about specific portions of the domain name tree.
- Resolvers are programs that extract information from name servers based on client requests.

19.1 DNS Server Hierarchy

The DNS servers are organized in hierarchical structures. Each name server is configured for a specific local zone, and it includes sub-domains.

There is an authoritative source for each portion of the hierarchy. At the top of the hierarchy are the root servers. There are different root servers for different top level domains. However, there is some redundancy within domain spaces to prevent bottlenecks.

19.2 DNS Operation

Common operation of the DNS database is initiated by a user program request for IP address of a given domain name.

The resolver module in the local host or ISP formulates a query for the local name server, which is at the same domain as the resolver. After this, the local name server checks the local database/cache and, if the matching address is found, returns the IP address to the requestor.

If the matching address is not found, the local name server will contact other available name servers, starting down from the root of the DNS tree or as high up the tree as possible. When a response is received, the local name server stores the name/address mapping in the local cache and the user program receives an IP address or an error message if the address can't be matched.

19.3 DNS Name Resolution

A DNS query begins with the name resolver located in the user host system. The first location to be checked is the system cache memory, which contains the recently obtained server name/IP address mappings.

If the requested name is not stored in the cache, a query is sent to the local DNS server.

19.4 IPv6

The major solution to having enough address space was supposed to be more addresses with IPv6. IPv6 has been under development for over a decade. From early on, it has run into trouble and the IETF has resorted to pushing for adoption. Contrary to what you may read, the only advantage IPv6 has is longer addresses.

All of the other “advantages” you may read about (QoS, Security, etc.) work equally well with IPv4. Furthermore, IPv6 exacerbates many problems, e.g. calculation of a forwarding table becomes an exponential of exponential.

The policy of “small incremental change” in the case of IPv6 was so small that there is insufficient economic advantage to those who have to pay for the conversion to make it worthwhile. Furthermore, the transition requires the use of a NAT (Network Address Translation). Adoption is further stymied because once an organization has a NAT, it no longer needs IPv6. For this reason, many strongly object to the use of NATs and private address space. These people point to a long list things that NATs break in the current architecture. However, with careful inspection one will find that all of these are either due to the fact that the Internet lacks a full addressing architecture or are simply bad things to do, such as applications passing IP addresses as parameters. (Application passing IP addresses is a bit like trying to pass absolute memory addresses as parameters in a Java program.) Subnets and Subnet Masks.

IPv6 also eliminates private address space. There is no equivalent capability in IPv6. Recently, it has also come to light that the lack of a solution to multihoming in IPv6 is causing router table growth to rise alarmingly. (This is exacerbated by the fact that during the transition both IPv4 and IPv6 must be supported.) At this point, it appears that Moore’s Law will not come to the rescue. This problem is currently under intense scrutiny in the IETF, but the current approach does not hold much promise.

19.5 Private Network Addresses and Network Address Translation

A series of IP ranges have been defined by the IETF to be used by any user/organization on their private network. These addresses may not be used on the public internet. This means the same private IP addresses may be reused locally by many organizations saving millions of IPv4 addresses. When the traffic is sent out of the private network to the public network, it must be translated to a public network address through a mechanism called NAT (Network Address Translation). On large user networks many private IP addresses may well pass through routers while still inside a private network; the traffic is mapped onto a smaller range of public IP addresses when it’s sent out.

The Range of IP Private Addresses:

- A 10.0.0.0 to 10.255.255.255
- B 172.16.0.0 to 172.31.255.255
- C 192.168.0.0 to 192.168.255.255

Network Address Translation (NAT) refers to the process by which a private IP address may be mapped (or translated) to one or more public IP addresses which can be routed across the internet. The NAT function is usually located in a Router or Proxy function at the edge of the network. A large number of private IP addresses can be mapped (or translated) to a single or a limited number of public IP address as needed.

Private address space is controversial. Some are very attached to the idea that every device on the network has a global address, even if the owner of the device would prefer that it not be globally visible. Furthermore, since the Internet does not have a full addressing architecture, this has lead to a number of kludges becoming accepted practice. Some of these are broken by NATs. However, owners of networks like private address for the added level of security they provide.

IP addresses may be either statically or dynamically assigned by a service provider. A dynamic IP address is normally assigned to the host when the user connects to the network. The network provider usually changes the IP addresses periodically. On the other hand, a static IP address is fixed, incurring more cost. Static IP addresses are normally used for host sites, email services, servers, etc., while a limited number of static IP addresses are typically used in conjunction with NAT to map a large pool of dynamic private IP addresses onto a limited number of static public internet addresses. A static assignment of addresses is a historical holdover from the early days of the Internet. As our understanding of addressing has improved, it became clear that addresses must be assigned by the network. Because only the network knows where in the network the system is.

19.6 Growth Causes Addressing Problems

As the Internet was expanding (or exploding) in 1990's many people started to fear that we would run out of IPv4 addresses. The total number of IP addresses was 2³² and that seemed too low. But this was the most visible of the problems. There was also great concern about the growth of routing tables. The organization that hands out addresses had not been too careful about how it was done. This fear initiated a number of "IP address saving" techniques, one of which was private network addressing. In addition, a policy was adopted to further limit the blocks of addresses allocated and force most users to get their addresses from their provider. Five actions were taken:

1. Classless Interdomain Routing (CIDR, pronounced "cider") was adopted to make IP addresses more aggregatable.
2. Most requests were advised to get their addresses from their provider. This improves the ability to aggregate routes.
3. When blocks of addresses were handed out, fewer were given out and some large Class A blocks were re-claimed from organizations that didn't really need them.
4. Private address space and Network Address Translation (NAT) was instituted to further relieve the demand for addresses.
5. Development of a new version of IP began.