THE ROUTE LESS TAKEN

DANIEL W. ENGELS, PHD

THE ROUTE LESS TAKEN

(NEARLY) ALL THE LAYERS

VERSION 0.1

First edition: September 2016

We reject kings, presidents and voting. We believe in rough consensus and running code.

David D. Clark, MIT, talk given to IETF 1992

The Net interprets censorship as damage and routes around it.

John Gilmore, EFF

On the Internet, nobody knows you're a dog.

Peter Steiner, The New Yorker, page 61 of July 5, 1993 issue

Contents

List of Figures

List of Tables

Dedicated to those who endeavor to comprehend and improve the connections in our lives.

Preface

I teach Computer Networks, like all other subject matter, in a very personal manner that reflects my personality, interests and inclinations to both challenge my students (and myself) intellectually and support them as they discover the beauty in the subject. I have a strongly held belief, driven into me by many professors, mentors, and my personal experiences, that all learning begins with the foundational, or first, principles for the discipline under study. These principles must be learned, explored, and applied before mastery of the subject matter can be achieved. This is true wether the subject matter is fundamental to a broad range of disciplines, such as Calculus, or is more limited in its scope, such as Cryptography. First principles are the foundation of every discipline, and Computer Networks is no exception.

Computer Networks, as a topic area, is easily described in its scope to be computer-to-computer, or machine-to-machine, communications whether done directly (i.e., direct computer-to-computer communications), through a sequence of direct connections (i.e., through a network) or through an indirect means such as through the exchange of removable flash memory drives. Because of this 'simple' description and the immense seemingly error free use of computer networks that students today have experienced throughout their lives, most students assume that Computer Networks is a simple and limited application domain with a small set of narrowly tailored problems that are easily solved.

Computer networks are *neither simple nor do their problems have easy solutions.* The knowledge base relied upon to understand and even design computer networks and communication protocols covers the whole of what a typical undergraduate computer science student should study during the course of obtaining their bachelor's degree and, in fact, goes well beyond what a computer science undergraduate student would normally study. Expertise in a broad range of disciplines, including, but certainly not limited to, the physics of communications, electronic design (both analogue and digital), protocol design, information theory, queueing theory, algorithm design and distributed systems is required to truly master Computer Networks.

My preference for first principles combined with the need to apply knowledge from a broad knowledge base leads me to teach Computer Networks in a holistic fashion that is primarily bottom up but is also top down simultaneously. Computer networks were built beginning with the physical medium, and, even in the layered network model, the Physical Layer forms the foundation of the layers. Medium access control, error recovery, flow control, addressing, indirect communications and the layer concept all came after the Physical Layer was made usable. And, many of the features of higher layer protocols were designed and implemented in response to some design requirement or problem experienced that ultimately was dependent upon the Physical Layer.

Some of the old computer networking problems persist today, particularly with the physical medium, likely because of the continued adoption and deployment of large numbers of wirelessly communicating devices. Advances in materials and electronics have driven significant changes in how we use various media, even the old twisted pair copper cables. However, most computer science students have little to no experience with the actual physical medium and the problems experienced because of its limitations.

In contrast to their limited experience with the Physical Layer, all students, in fact all young people today, have extensive experience with applications and the Applications Layer that resides at the top of the network layer stack. Todays applications hide the network functionality from the user, meaning that students typically have no direct experience with the layers below, or the networks used by, the applications. Students have simply used the network, but they have not been trained in or shown how the network works. This is akin to drivers of automobiles. Most adults can drive a car; however, very few people know how an automobile works with even fewer of these people knowing how to fix it. Ultimately, there are fewer people still capable of designing an automobile.

My goal, and the goal of this book, is to train computer network designers. By following a holistic, first principles approach to teaching, this book provides the foundation for network designers to learn, explore and master the first principles of Computer Networks. Along this journey, students will understand why the network applications that they use function the way that they do (and don't function from time to time) and understand how to write better, more secure network applications. Ultimately, by mastering the first principles, students will be able to design novel communication protocols.

Along the way to their mastery, my hope is that students will gain a love for the computer networking and the problems that must be solved to make them work.

The potential material that can be covered in a Computer Networks book is extensive, and this book does not even attempt to cover all of the material or even all of the details of the material that is covered. In writing this book I have attempted to limit the content to only that material which I consider to be the first principles and the core details required for a semester long introductory course in Computer Networks. This necessarily leaves out many details and concepts that are important to computer networks in practice. So, you may not find your favorite details or concepts covered in this book.

Another tenet of my teaching philosophy is that learning should be fun, or at least not a grind. When faced with a one thousand page book to read over the course of a semester, most students opt to listen to lectures and read only those portions of the book necessary to answer homework questions and, hopefully, test questions. This is not a good learning environment, and when I use these extra large paper weights and door stops, my students tell me that they do not read every word, every page, every section or even every chapter, in part because they are bombarded with too many facts and details and concepts of no consequence to their grade.

By focusing on a limited amount of material and providing it in a more-or-less summary oriented form, my goal is to provide the space, and the incentive, for students to read, study, apply and master the fundamental first principles of Computer Networks and, as you learn, to come to love the beauty that is computer networking. To this end, this book is not a one thousand page tome, and I have worked hard to make it easy to read and easy to use this book as a study reference. This book is created from my notes on computer networks, and, to the greatest extent possible, I have attempted to make the contents of this book simple to understand regardless of the complexity of the material being presented. The numerous and important details beyond those contained in this book may be found elsewhere in other books, through the Internet or through the thorough examination of the computer networks that we use every day. I encourage you to find these details for the topics that you find interesting.

I wish you, my student, excellence in your studies and mastery of Computer Networks. May you come to love this subject as I have.

DWE, 15 August 2016

Introduction

Computer networks are the backbone of the information society. Our lives revolve around the connected devices that put the collective knowledge of the world at our fingertips and allow us to speak with nearly every human in the known galaxy.

Mark Weiser, in his landmark 1991 Scientific American article entitled "The Computer for the 21st Century," noted that *The most profound technologies are those that disappear. They weave themselves into the fabric of our everyday life until they are indistinguishable from it.* The Internet, and computer networks more broadly, have most definitely woven themselves into the fabric of our daily lives. Connectivity, really, connectivity to the Internet, has become the lifeblood of our everyday lives. And, all of the technologies that are making our world smarter are dependent upon this seemingly ubiquitous connectivity in order to further weave themselves into our ever more connectivity dependent every day lives.

Increasingly, the things around us are intelligent and thoughtful and capable of making our lives heaven on earth (or a living hell) by being connected to a network. Our ever-growing smart things and this connected world that is our reality are possible only through the wonders of computer networks. Without computer networks, the collective knowledge of the world would be locked away somewhere in a library in Alexandria, and we would not have the information revolution that created our current Information Age. Computer networks, quite literally, bring the world to our fingertips and bring our local world to all of humanity. With the network making its way into our everyday objects and the very fabric of our lives, increasingly computer networks bring life to our world. As the networks and the devices using them disappear into the fabric of our lives, we become dependent upon these devices and, therefore, the computing networks that make them so useful.

The *Internet of Things* (*IoT*) is made of up billions of network connected devices embedded within the things around us. These things include our heating and air conditioning systems, our lighting systems and even our groceries. These things are primarily connected via wireless communications to each other, and eventually to a server via a gateway. The information support allows for a broad range of application appropriate communication protocols and cost effective devices to be used. The servers and information systems supporting the IoT devices often provide the real functionality of the system with the IoT devices acting as the senses of the network.

Yes, computer networks are ubiquitous and getting even more ubiquitous every day. Cloud computing, big data, social networks, mobile computing, cellular networks and the Internet of Things (IoT), amongst other networked services and computing networks, collect data, store data and process data in huge quantities, all while enabling us to access the data and processed information from anywhere at any time. The diversity of computing networks is expanding too. Large servers and data warehouse systems require high speed connectivity and access from anywhere in the world. Smart devices tend to be more local in their communications, either communicating in a mesh to other nearby devices, communicating directly to a master controller connected to the Internet or communicating directly through a gateway to a server located somewhere over the Internet. Each communication scenario has a unique set of communication speed and data requirements and functional and physical limitations requiring different computer networking protocols and communication hardware. The result is that we are surrounded by a set of heterogeneous computer networks that mimic the heterogeneous computer hardware using those networks.

With our very way of life now completely dependent upon computer networks, it is imperative that we as a society and, in particular, as engineers and scientists, know and understand not just how to use or manage a computer network but also how to design a computer network beginning with first principles. The broad diversity of computer networks that surround us, while each is seemingly completely different and unique, are all designed in accordance with the same set of first principles and the same fundamental architecture. This fundamental architecture and these first principles have been designed and learned over the course of decades of experience designing, using and evaluating computer networks in practice.

Modern computer networks got their start with the ARPANET developed in the late 1960s and early 1970s with significant funding from the U.S. Advanced Research Projects Agency (ARPA). Before ARPANET, networks were proprietary vendor specific connections designed primarily to connect terminals and remote entry stations to a mainframe. ARPANET changed the paradigm from master-slave communications to peer-to-peer communications. With equal peers came distinctly different traffic patterns, and ARPANET relied upon the then novel technique of packet switching to efficiently utilize communication resources.

In their 1996 book, *Where Wizards Stay up Late: The Origins of the Internet*, Katie Hafner and Matthew Lyon provide a detailed history of how the Internet was born of the blood, sweat and tears of a small army of pioneers and visionaries.

ARPANET began in 1969 with a small set of four computers located in California and Utah as shown in Figure 1. By 1977, ARPANET had grown to more than 50 computers spanning the United States. By then, ARPANET had experienced all the early growing pains, and problems, of moving from a small, easily managed set of computers, to a highly distributed mesh-based computer network.

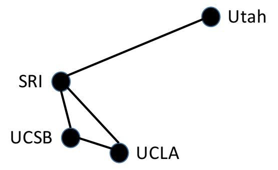

Figure 1: The ARPANET circa 1969.

The initial ARPANET design had a finite structure and introduced the protocol layering design concept. In protocol layering, the total communication functions are divided into several layers, each building upon and isolated from the services provided by the layer below. This layered concept persists in nearly all communication protocols used today.

In the beginning ARPA created the ARPANET.
And the ARPANET *was without form and void.*
And darkness was upon the deep.
And the spirit of ARPA moved upon the face of the network and ARPA said, 'Let there be a protocol,' and there was a protocol.
And ARPA saw that it was good.
And ARPA said, 'Let there be more protocols,' and it was so. And ARPA saw that it was good.
And ARPA said, 'Let there more networks,' and it was so.
– Danny Cohen, speaking at the *Act One* symposium, 1989.

The 1980's saw an explosion in networking performance and the number of computer networks that were supported. By the late 1980's many of these networks could interact, allowing for a user to traverse multiple networks from a single node. The newer networks, such as NSFNET, utilized the latest high speed communication technologies that significantly outperformed the older technologies underlying the ARPANET. By 1990 the ARPANET had been completely decommissioned, providing an end to the beginning of the Internet.

By the early 1990's, the primary backbone networks worldwide, such as NSFNET and UUNET, were interconnected, forming the Internet as we know it today. Businesses had identified the business opportunities in computer networking equipment during the 1980's, resulting in the founding and success of a large number of computer network equipment manufacturers and software companies. In 1991, the introduction of the World Wide Web (WWW) laid the foundation for the land rush that has become the "Internet" as most people know it today. The release in 1993 of the Mosaic browser, written by Marc Andreessen and Eric Bina of the University of Illinois' National Center for Supercomputing Applications, opened the World Wide Web to non-programmers and led directly to the Internet boom, and the boom of online businesses, of the late 1990's.

Metcalfe's Law states that the value of a network is proportional to the square of the number of devices on that network.

The term *cyberspace,* which refers to the virtual realm of electronic communication, interaction, community and accessible information, was introduced in 1984 with the release of the science fiction novel, *Neuromancer,* by William Gibson.

The Internet of the 2010's is surprisingly recognizable as being substantially similar to the Internet of the 1990's. The innovations over the intervening quarter of a century were incremental in performance enhancement, reliability and functionality. The Internet of Things has extended the reach of the Internet to everyday items. But, throughout these decades, the primary innovations were happening within the applications and how the networks were used but not necessarily how computer networks are designed.

The fundamental computer network design of today is substantially the same as the fundamental design of the ARPANET. The network designs, layering concepts, packetized communications and operational decisions created and made in response to the requirements of the late 1960's and early 1970's still dominate and strongly influence the network designs of today.

The layered design of communication protocols, with the concepts of encapsulation, end-to-end communications and black box networks, dominates the division of functionality between the devices in a computer network and strongly influences how the interconnections of and interactions between computers is conceived and provided.

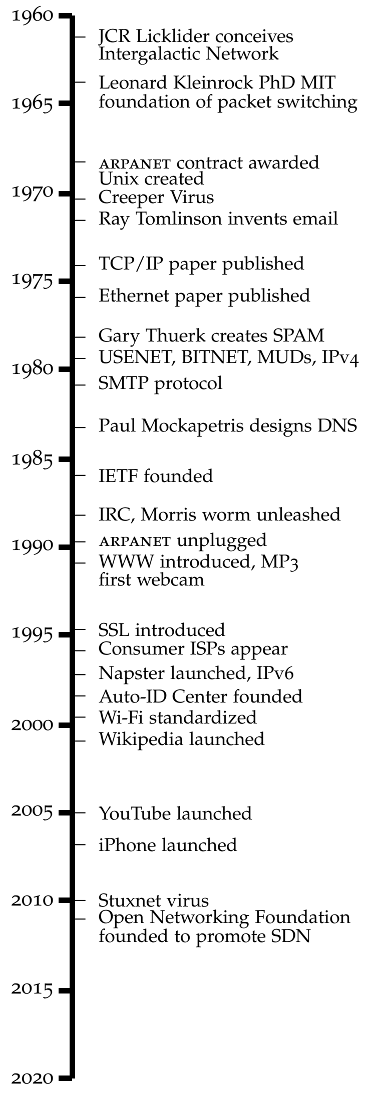

Figure 2: Select events in the history of the Internet.

Ultimately, computer networks, in all their variations and all of their evolutions, provide value to the users and devices connected to the networks. *Metcalfe's Law* states that the value of a network is proportional to the square of the number of devices (users) on the network. This is certainly true for communications networks dominated by human conversations. For machine to machine and automated conversations, the value of the network is limited to those devices that are authorized to communicate with one another. For these types of networks, such as the automated control and monitoring systems for the power grid, devices that are not authorized to communicate with the power grid systems are threats. Consequently, unauthorized users and devices reduce the value of the network for the power grid systems.

Due to the range of attacks that an open network affords to non-authorized devices and users, security is a critical component of computer networks. Security was not designed into the original protocols of either the Internet or computing systems. Consequently, security is layered on top of insecure systems.

Security is provided, in part, by maintaining private networks. Private networks typically do not connect to the public Internet, and they typically require access to a physically secured location in order to access the network. Virtual private networks can be utilized over the public Internet to secure communications, but they do not provide the same level of security as a true private network.

Computer network designs continue to evolve as new technologies, new applications and new approaches to computing are put into practice either for or utilizing computer networks. The focus on security is causing an evolution away from the insecure computer networking protocols and policies developed to launch what became the Internet. The likely result is that the future computer networks will become fundamentally secure systems whose design is inspired by the foundational concepts, first principles and fundamental design philosophy of the original computer networking protocols.

Computer networks will continue to evolve into more secure, more capable and higher performance systems. The resulting future network designs will be based upon the first principles of computer networks.

1 An Introduction to Computer Networks

Primary Learning Objectives:

- Understand the basic role for computer networks.
- Understand the basic topologies, relationships, and types of computer networks.
- Understand the network layer model used in computer network design and communication.

Computer Networks were born from the desire to share resources, to transfer data and files and, more generally, to communicate from one device to another. This desire to share resources and to communicate between devices led inexorably to the computer networks that we have today. Simple peripheral communication busses transformed into timeshare systems with remote terminals that further evolved into local area networks and then to the highly distributed, mobile and global Internet network of networks that we experience today. Simply stated, computer networks connect computers and other computing devices to one another. The widespread adoption of computer networking for every computing device, no matter how small, enabled the information and communication revolution that has transformed, and continues to transform, the very fabric of our lives.

In this chapter, we explore the fundamental concepts and designs of computer networks by taking a top-down approach in our exploration and performing a broad overview of computer networks and their basic layered design approach.

1.1 Overview

Computer networks always require at least two communicating computing devices. Two directly communicating devices is the smallest possible computer network. The largest computer network is the *Internet* that connects billions of computers and devices from across the world.

Computer networks are groups of computers that are able to communicate with one another either directly, i.e., machine to machine, or indirectly, e.g., through massages passed through other devices.

There exists a broad range of different fundamental types of computer networks as shown in Table 1.1. These types, including the ubiquitous Local Area Network (LAN) and Wide Area Network (WAN) networks, are differentiated primarily along the geographic scope of their direct communications. There is considerable overlap in the communication protocols used by each type of network; however, some protocols, such as Bluetooth, are designed for a particular

Fundamental types of computer networks
Sneaker Net
Body Area Network (BAN)
Personal Area Network (PAN)
Local Area Network (LAN)
Metro Area Network (MAN)
Wide Area Network (WAN)
internetwork

Network Type	Brief Description
Sneaker Net	Removable storage device transported by humans from computer to computer
Body Area Network (BAN)	Single etwork restricted to devices contained on, or in, a human body
Personal Area Network (PAN)	Single network restricted to devices on a person or in their immediate surroundings
Local Area Network (LAN)	Single network restricted to a localized area or facility
Metropolitan Area Network (MAN)	Single network restricted to a large geographic area such as a city or a campus
Wide Area Network (WAN)	Single network that spans a large geographic area such as a continent
Internetwork (internet)	Network of networks interconnected by gateways and gateway routers

Table 1.1: General Types of Computer Communication Networks

network type, PAN (Personal Area Network) in the case of Bluetooth. The reason for the protocol overlap is that there is not always a clear line delineating one type of network from another. For example, a network connecting a person's phone, computer and printer using WiFi may be considered to be either a PAN or LAN. However, the rise of the Internet of Things (IoT), that is, smart devices that are connected to the network, is moving the notion of a PAN to be human centric causing the network in our example to be considered more and more as a LAN and not as a PAN. This simply shows that our perception of what is a particular type of computer network changes over time and may even vary from person to person. As such, these fundamental network types operate more as guideposts regarding communication distances and likely communication protocols than as definitive rigid classifications.

The brief descriptions shown in Table 1.1 define all fundamental network types as being a single network. The *internet,* defined as a network of networks, is the lone exception. This means that there is little, if any, hierarchy within a particular fundamental type of network, although their *topology* may be complex. This also means that only the internetwork type of network may contain multiple types of fundamental networks and large hierarchies of networks. We utilize this simplistic view of the fundamental network types. The devices that are part of a particular type of network are considered to exist at the same level, or layer, within the network.

Internet vs internet The *Internet* (capital I) refers to the globally interconnected set of computers and devices that may be accessed from anywhere in the world. The *internet* (lower case i) is simply any set of interconnected networks.

Specialized computer networks exist in addition to the fundamental types of computer networks. A brief list of some of these specialized networks is shown in Table 1.2. The *Internet* is included with this list of specialized networks, since the Internet is the global

Specialized types of computer networks include
- Internet
- Cellular Network
- Storage Area Network (SAN)
- Enterprise Private Network (EPN)
- Virtual Private Network (VPN)

Table 1.2: Examples of Specialized Types of Computer Communication Networks

Specialized Network Type	**Brief Description**
Internet (the Internet)	The internetwork that connects computers and devices globally
Storage Area Network (SAN)	A network that provides access for operating systems to data storage devices
Enterprise Private Network (EPN)	A network or internetwork that is managed and controlled by a single entity
Cellular Network	A network or internetwork of wireless base stations designed to communicate to mobile hand sets
Virtual Private Network (VPN)	A private network that is extended across a public network such that all devices appear to be on the same private network

public network connecting computers and networks worldwide. Other specialized network types, such as EPNs (Enterprise Private Networks), provide for a range of functionality that builds upon the fundamental network types. An EPN, for example, may consist of interconnected WANs and LANs that are all controlled by a single organization or company. An EPN is typically not directly connected to the Internet to prevent network access from anywhere in the world. Such private networks are operated by large companies such as Google and Facebook to connect their servers and data centers. They are also used by governments and corporations to protect internal communications from the range of attacks that exist when public networks are used.

The Internet can be mapped by using `traceroute`, by collecting routing tables from Border Gateway Routers and by using a combination of these two approaches. (Other approaches exist; however, they are much more complex than these basic approaches.) An example of an Internet mapping using these approaches is available at The Opte Project (http://www.opte.org). The Opte Project has performed several Internet mappings with the resulting Internet data displayed in large, visually appealing graphs.

1.2 *Network Communications*

Computing systems and networked devices exchange *messages* with one another. These messages may be anything appropriate for the two systems including a `GET file` command sent to a web server and the `file` itself returned from the web server. Many files, such as the file that is this book, are too large to be sent as a single message, or single sequence of bits, between the communicating systems. Large is a relative term that is determined by many factors including how many systems are using the same communication link, how fast bits are sent over that link, and how much memory each system has for being able to either send or to receive the message. Because of these very practical limitations, messages are broken into discrete *data blocks*, typically having a fixed maximum number of bits, and the data blocks are sent sequentially between the systems until the complete message is communicated.

Packetized communication is the most common form of communication in data networks. **Packets** consist of a *message* or portion of a message that is *encapsulated* by a *header* and a *footer.*

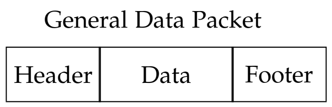

Figure 1.1: Data is encapsulated between a Header and a Footer in packetized communications.

In data networks, it is common for the data blocks to be *framed* as shown in Figure 1.1. Framing involves adding a *header* before the data block and a *footer* after the data block. The framed data block is generically referred to as a *packet.* The header typically contains information that indicates the beginning of the packet, information regarding the data, such as its length and its block number, as well as communication information such as a source address and a destination address to indicate the sender and the intended receiver respectively. The footer typically contains information indicating the end of the frame and information for error detection or correction for any transmission errors that may occur during transmission.

Packets may be communicated either through a *packet switched* network or through a *circuit switched* network. Some circuit switched networks, such as the landline POTS telephone system, allow for the message to be sent directly. POTS circuit switched systems do not use packeted communications, and, therefore, they do not use framing of the data.

POTS, or *plain old telephone service,* is the traditional landline phone service provided to fixed locations. POTS networks provide a physical end-to-end circuit switched communication and do not use framed data blocks, i.e., POTS systems do not use packets for communication between devices.

1.2.1 Packet Switching

In a *packet switched* network, all communications over the network are made by transmitting a packet in a *store and forward* approach. A packet switched network contains multiple devices, such as routers and switches, that connect end systems through the network. In store and forward transmission, the complete packet is sent along a link in a stream of continuous bits. At each router along its journey, the complete packet is received before the first bit of the packet is sent along the next link. The store and forward approach allows for each router to examine the packet to ensure that no errors were experienced in the communication. Packets received with uncorrectable errors may be dropped from the system.

Packet switched networks send packets in a **store and forward** manner. In a store and forward system, packets are sent point-to-point from one device to the next device and received in their entirety before being sent on to the next device.

Routers and other networked devices may queue multiple packets that are received at a faster rate than the router may process them and send them on to their next link. The routers use a forwarding table or routing table to determine on which link a packet is sent. Packets for the same message may follow different paths through the network to arrive at the same destination.

1.2.2 Circuit Switching

In a *circuit switched* network, the resources needed to communicate between two end systems are reserved along a path connecting the hosts. The path must be determined, and the resources along the path reserved, prior to the communication beginning. These resources are reserved for the duration of the communication session between the two hosts. All messages and data blocks sent in the

Circuit switched networks utilize a predefined path through the network for communication between two hosts. The path must be set up and the resources needed for the communication are reserved along the path before communication begins. Data is sent continuously along the circuit.

communications follow the same path.

In the traditional circuit switched networks, such as POTS networks, only messages and other control signals are sent along the path, usually in a predetermined time slot or consuming the entire bandwidth along the path for the duration of the communication session. In these traditional circuit switched networks, an actual physical electrical circuit is created between the communicating hosts. When no data is sent during an allocated time slot for a circuit path, that slot remains unused by any other communications since any data appearing in that slot will necessarily be delivered to the receiving host allocated to that slot.

Circuit switching may be performed over a packet switched network. The circuits that are created over a packet switched network are referred to as *virtual circuits.* Virtual circuits are similar to traditional physical circuits in that all communications follow the same paths; however, all data blocks are framed in the virtual circuit as required by the packet switched network routers and devices and are communicated as packets.

A **virtual circuit** is created over a packet switched network by defining a path through the network and allocating resources to the communication along the path. Packets, i.e., framed data blocks, are sent along the path. When the allocated resources are not being used by the communicating hosts, those resources may be used by other packets transiting through the devices and links on the path.

Communication sessions often have short bursts of high activity followed by large periods of inactivity. When using virtual circuits, the underlying use of packetized store and forward functionality allows for each router to utilize its resources to communicate other packets during the long periods of inactivity over a particular virtual circuit.

1.3 Layered System Design

Layering is the central design philosophy for computer network design. In a layered design approach, a system architect creates modules that provide black box functionality to other modules through well defined interfaces. This is to say that how the functionality of a module is achieved does not matter to any other module, so long as the services provided by that module and the interfaces to access those services are constant.

Layering is a design approach that modularizes the system and component designs so as to both reduce the complexity of any one module and to minimize the dependence of one module upon another module for its design.

The general layered approach is illustrated in Figure 1.2. In computer network designs, lower layers provide services to upper layers through well-defined APIs (Application Programming Interfaces). The layered design approach has been applied to computer network design from its early days. These designs have resulted in three similar primary layered architecture designs for computer networks: the OSI Model, the Network Model and the TCP/IP Model. Each of these models is similar to the other models with differences in the number of layers and the division of functionality between the layers.

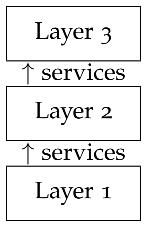

Figure 1.2: Layered designs reduce complexity and improve manageability.

1.4 *Layered Communication Model Overview*

The *OSI Model* is the most general network architecture model. The OSI Model contains seven layers with a relatively clean structure. The seven OSI layers are: Physical Layer, Data Link Layer, Network Layer, Transport Layer, Session Layer, Presentation Layer and Application Layer.

The **OSI Model** has seven layers: Physical, Data Link, Network, Transport, Session, Presentation and Application.

The *Network Model* contains five layers that encompass the same functionality of the seven OSI layers. The five layers are: Physical Layer, Data Link Layer, Network Layer, Transport Layer and Application Layer. The Network Model Application layer encompasses the functionality of the top three OSI layers, the Session layer, the Presentation layer and the Application layer.

The **Network Model** has five layers: Physical, Data Link, Network, Transport and Application. The Network model is the most widely deployed model, and it is essentially a subset of the OSI Model and a superset of the TCP/IP Model.

The *TCP/IP model* contains four layers that encompass the same functionality as the five Network layers. The fours layers are: Network Interface layer, Internet layer, Transport layer and Application layer. The TCP/IP model Network Interface layer encompasses the functionality of the bottom two Network layers, the Physical layer and the Data Link layer. The Internet layer is equivalent to the Network layer of the Network Model, and the Transport layer and the Application layer are equivalent to their same named layers in the Network Model.

The **TCP/IP Model** has four layers: Network Interface, Internet, Transport and Application.

Figure 1.3 illustrates the functional equivalents of the various layers of the three models.

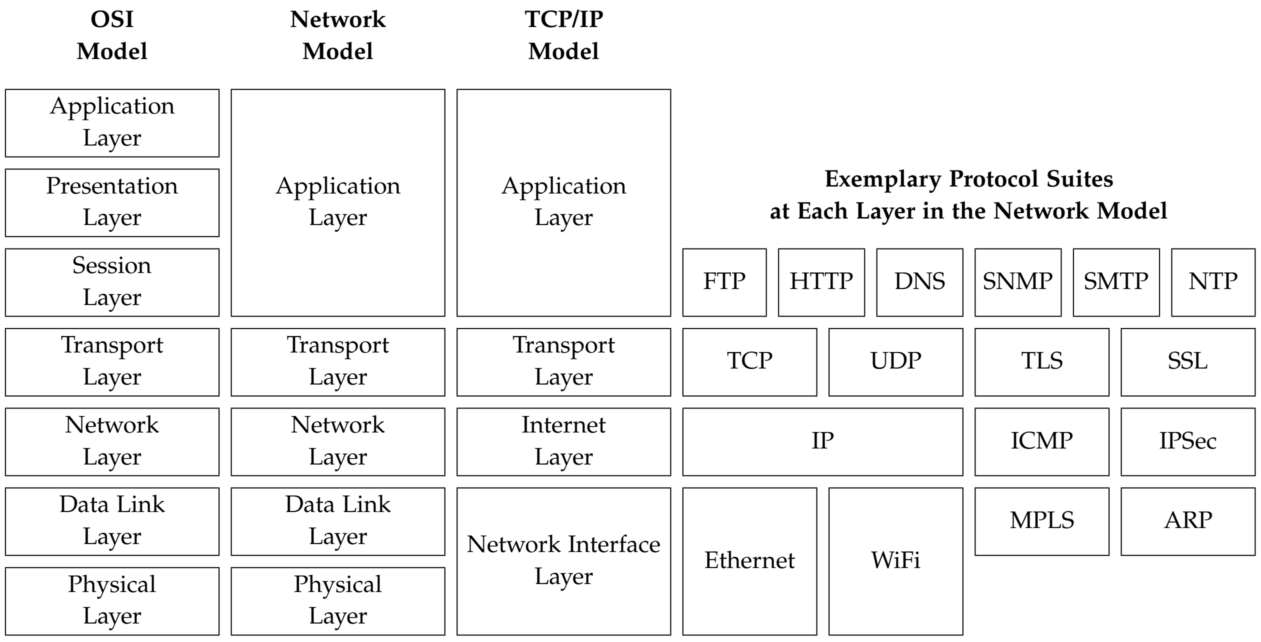

Figure 1.3: Comparison of the layers in the OSI Model, the Network Model and the TCP/IP Model and several protocols defined at each layer.

Each layer of these models contains its own set of protocols for communication within the layers and communication between layers. The protocols of the layers are referred to as a *protocol stack.* Figure 1.3 lists several exemplary protocols that exist at each layer of the Network Model.

A **protocol** is a set of rules and formats to be used from communications. There are two primary portions specified for all protocols:

- the sequence of messages that must be exchanged, and
- the format of the messages and of the data.

We primarily utilize the Network Model in this book. In the following subsections, we briefly examine the Network Model layers.

1.4.1 Physical Layer

The Physical (PHY) Layer is primarily responsible for transmitting bits and symbols over a communication medium from one node to a physically adjacent node. The nodes communicate over a *channel.* Traditionally, the two broad classifications of channels are *digital* and *analogue.*

The **Physical Layer** provides a physical link for transmitting a sequence of bits, or symbols, between any pair of nodes joined by a physical communication channel such as a twisted pair cable.

The protocols used in the Physical Layer are channel dependent. The dependency is driven in part by the channel medium (for example, twisted pair or wireless frequency). Ethernet, for example, is a collection of multiple protocols, each utilizing a different transmission medium, different data coding and different signalling.

1.4.2 Data Link Layer

The Data Link (LINK) Layer is primarily responsible for error control, flow control and access control over the physical channel directly connecting hosts. The net result of these responsibilities, and primary use of the Data Link Layer, is to convert the unreliable, error prone bit pipe of the Physical Layer into a reliable communication link for sending *frames* asynchronously and seemingly error free.

The **Data Link Layer** provides a virtual communication link between physically connected devices, and it provides for error control, flow control and access control of **frames** over that communication link.

The Data Link Layer services to the Network Layer include:

1. unacknowledged connectionless service
2. acknowledged connectionless service
3. acknowledged connection-oriented service
4. protocol specific services

Link Layer packets are known as *frames*. The contents of a frame are defined by the protocol being used. Most protocols include both a source address and a destination address in the frame. Addresses at the Data Link Layer are referred to as *MAC addresses.*

The Data Link Layer offers three basic services to the Network Layer: unacknowledged connectionless service, acknowledged connectionless service and acknowledged connection-oriented service. Additional services may be offered depending upon the protocol.

Medium Access Control (MAC) is a primary responsibility of the Data Link Layer. The medium access control functionality is referred to as being contained in the *MAC Sublayer.* MAC functionality ensures that different devices can access the same medium with limited interference from one another. A broad range of MAC algorithms exist. The most used algorithms are based upon the Aloha protocol.

The **MAC Sublayer** manages all of the medium access control functionality.

Addresses at the Data Link Layer are referred to as **MAC addresses.**

1.4.3 *Network Layer*

The Network Layer is primarily responsible for routing data from sender to receiver through the network. The basic Network Layer design has the Network Layer operate as a black box to the upper layers. Packets go into the network from the message source and arrive at the destination some time later. Due to this lack of visibility into the network, flow control operations undertaken at the upper layers may make network performance worse. Consequently, feedback mechanisms have been integrated into the Network Layer packets to provide some limited visibility into the network operations in an attempt to allow the upper layers to provide for flow control that results in better overall network performance.

The network packets contain *data,* that may be of variable size, but there is a maximum data length referred to as the *Maximum Transfer Unit,* or MTU.

Network Layer packets are known as *datagrams*, or simply *packets.*

The **Network Layer** provides services and functionality to move Network Layer **datagrams**, also referred to simply as **packets**, from a source device to a destination device through the various network routers and other devices, i.e. the Network Layer provides routing services.

The Network Layer services to the Transport Layer:

1. should be independent of router technology
2. should be independent of and hide router topology
3. should use an internet-wide identification scheme

IP addresses are the global addresses used to uniquely identify hosts at the Network Layer.

1.4.4 *Transport Layer*

The Transport Layer is primarily responsible for end-to-end communication services. The Transport Layer operates as if the hosts are communicating directly with one another. Consequently, the Transport Layer provides for error control and flow control just as the Data Link Layer. The algorithms used at both of these layers are similar in their fundamental design, although they typically differ significantly in the details of their operations.

Messages are divided into packets in the Transport Layer. Ideally, the packets are sized such that they may transit across all links without *fragmentation.* At the destination, the packets are reassembled into the message.

A Transport Layer packet is referred to as a *segment.*

The **Transport Layer** provides for end-to-end, or host-to-host, communications. The Transport Layer communicates message **segments** between hosts and performs error control and flow control of messages and segments.

The Transport Layer services to the Application Layer:

1. provides end-to-end packet delivery
2. connection-oriented packet delivery
3. connectionless packet delivery

1.4.5 *Application Layer*

The Application Layer is primarily responsible for the higher level protocols such as the File Transfer Protocol (FTP) and distributed system services such as the Domain Name Service (DNS). Application layer protocols are the most general and flexible networking protocols. They rely upon the functionality of the underlying layers in order to communicate from one machine to another.

Distributed system services provide for a broad range functionality that is accessible over or through the network.

The Application Layer communicates a packet of information referred to as a *message.*

The **Application Layer** provides the primary interface between users and applications and the network functionality. Protocols such as HTTP, FTP and DNS and higher level services operate at the Application Layer. The Application Layer communicates **messages** between hosts.

1.5 *Layered Communication*

Communication between two computers requires the message sent from the source application to travel down through each of the layers prior to being sent through the Physical Layer and, upon receipt by the Physical Layer of a host, to travel up through each of the layers to the Application Layer. Figure 1.4 illustrates both the actual connections (with the thick line) through the layers and between devices at the Physical Layer. Layers in the sending and receiving hosts communicate with one another forming a virtual communication link for each layer.

In **layered communication** each layer communicates with the corresponding layer in other devices. This results in virtual connections between the layers. This is particularly important for the end to end communications at the the Transport Layer and at the Application Layer.

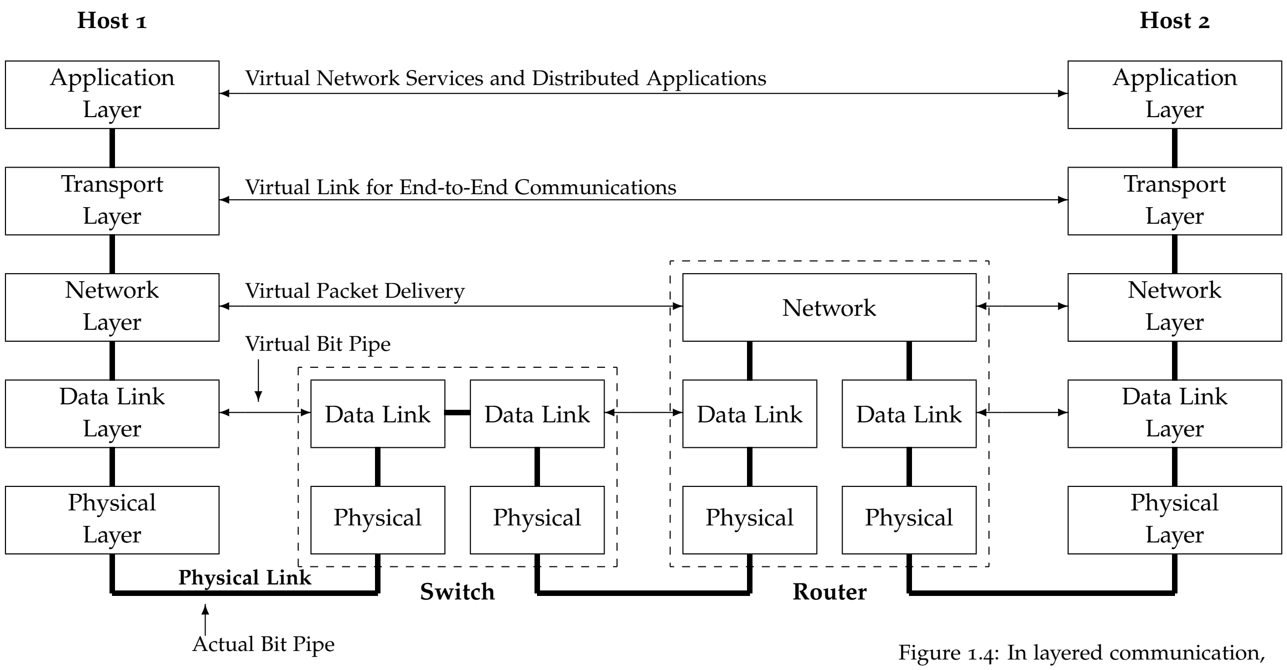

Figure 1.4: In layered communication, each layer forms a virtual communication link between the devices.

Figure 1.4 illustrates the layers examined by various network components. Hosts examine all layers when they are either the source or the destination for a packet. Other network components, such as a *switch* and a *router*, examine only some of the layers as shown in Figure 1.4. A switch connects devices at the Physical Layer and the Data Link Layer, allowing devices on either side of the switch to operate as if they are physically connected. Therefore, the switch examines the communicated packets at these two layers only. A router, in contrast, needs to determine the next device on the path through the network for the packet. The global address of the destination host is the IP Address contained in the Network Layer packet. Therefore, the router examines a packet up to the level of the Network Layer.

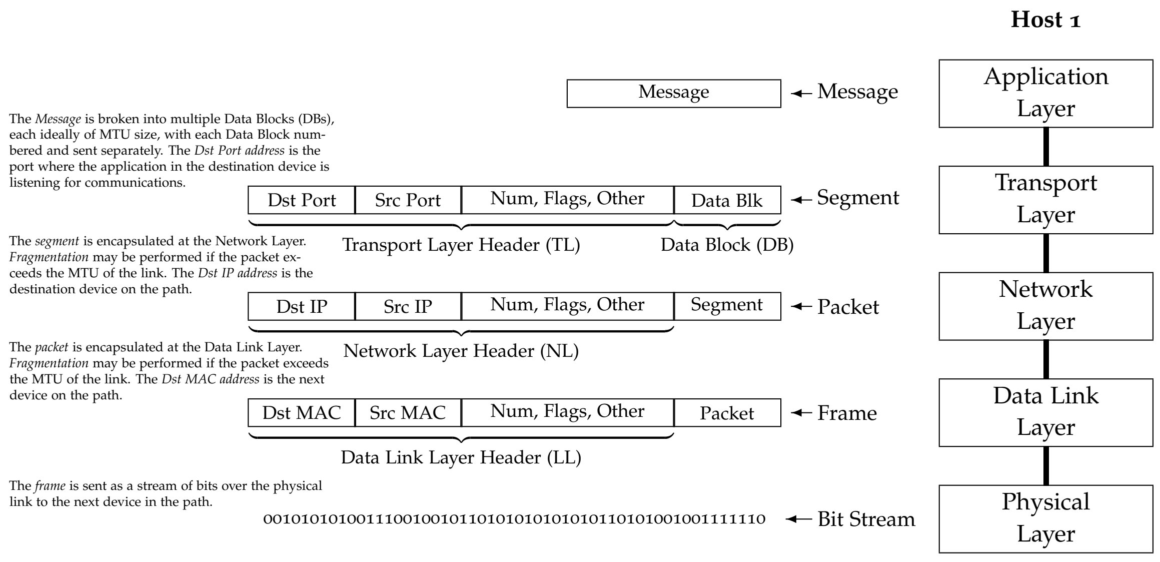
Host 1
Application Layer
Transport Layer
Network Layer
Data Link Layer
Physical Layer
Message
Message
The *Message* is broken into multiple Data Blocks (DBs), each ideally of MTU size, with each Data Block numbered and sent separately. The *Dst Port address* is the port where the application in the destination device is listening for communications.
Dst Port
Src Port
Num, Flags, Other
Data Blk
Segment
Transport Layer Header (TL)
Data Block (DB)
The *segment* is encapsulated at the Network Layer. *Fragmentation* may be performed if the packet exceeds the MTU of the link. The *Dst IP address* is the destination device on the path.
Dst IP
Src IP
Num, Flags, Other
Segment
Packet
Network Layer Header (NL)
The *packet* is encapsulated at the Data Link Layer. *Fragmentation* may be performed if the packet exceeds the MTU of the link. The *Dst MAC address* is the next device on the path.
Dst MAC
Src MAC
Num, Flags, Other
Packet
Frame
Data Link Layer Header (LL)
The *frame* is sent as a stream of bits over the physical link to the next device in the path.
001010101001110010010110101010101010110101001001111110
Bit Stream

Figure 1.5: Each layer has unique framing. Most protocols add only a header at a layer.

The layers examined by each device are more easily seen, and are more readily apparent, when we consider the framing applied at each layer. Figure 1.5 illustrates the framing that is typically applied at each layer in the Network Layer Model. The Application Layer generates a *message* and communicates that message to the Transport Layer using a specific port and identifying a specific IP address. The Transport Layer breaks the message into MTU sized *data blocks* that are then framed to create a *segment.* The segment is passed to the Network Layer along with the source and destination IP addresses, and the segment is framed to create a *packet.* The packet is passed to the Data Link Layer, and the packet is framed to create a *frame.* The frame is passed to the Physical Layer where it is sent along the physical medium as a *bit stream.* Table 1.3 summarizes the names of the framed packets at each layer in the Network Model.

Table 1.3: Packet Name at Each Layer

Packet Name	**Layer**
Message	Application
Segment	Transport
Packet*	Network
Frame	Data Link
Bitstream	Physical

*Packet is also a general term used to refer to the framed data at each layer. *Datagram* refers to the packet at the Network Layer only.

Figure 1.6 illustrates how the layers are used during communications between hosts. Given a message the size of an MTU Data Block (DB) sent from Host 1 to Host 2, framing is added at each layer in Host 1 as the Data Block moves from the Application Layer to the Physical Layer. When the resulting packet bitstream arrives at a router, the router examines the MAC address in the Link Layer header. When the destination MAC address matches the MAC address of the Network Interface Card (NIC) on which the router received the packet, the router removes the Link Layer framing and sends the resulting Network Layer packet to its Network Layer. The Network Layer examines the source and destination IP address in the Network Layer header, and, by consulting its routing table, it determines the NIC on which to send the packet towards its destination.

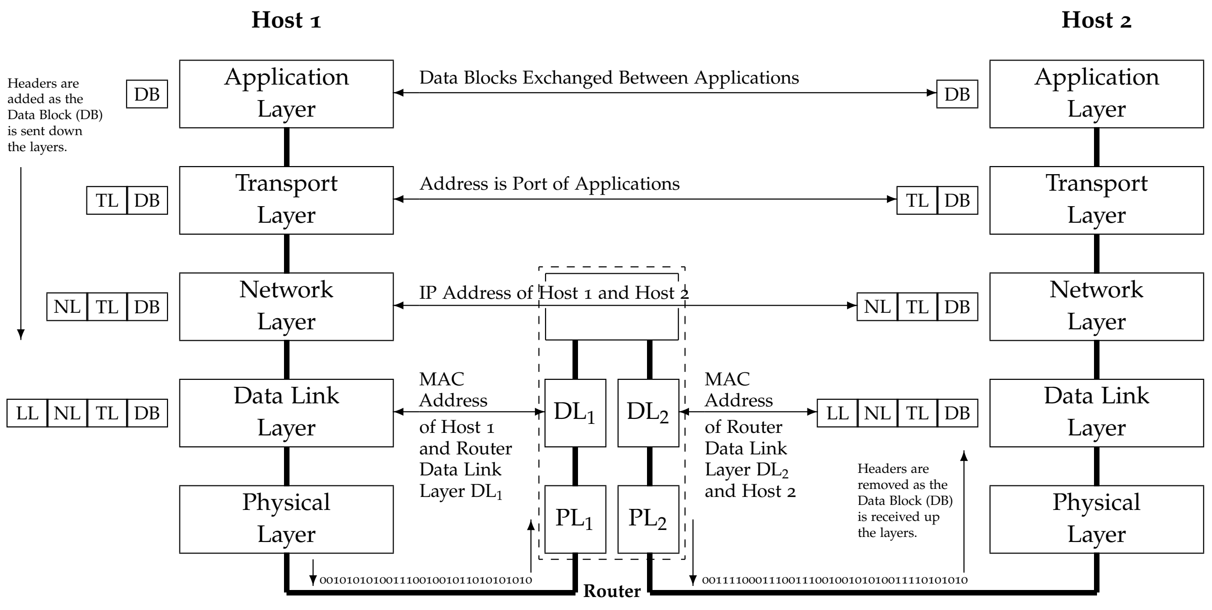
Host 1
Host 2
Headers are added as the Data Block (DB) is sent down the layers.
DB
Application Layer
Data Blocks Exchanged Between Applications
DB
Application Layer
TL DB
Transport Layer
Address is Port of Applications
TL DB
Transport Layer
NL TL DB
Network Layer
IP Address of Host 1 and Host 2
NL TL DB
Network Layer
LL NL TL DB
Data Link Layer
MAC Address of Host 1 and Router Data Link Layer DL1
DL1
DL2
MAC Address of Router Data Link Layer DL2 and Host 2
LL NL TL DB
Data Link Layer
Headers are removed as the Data Block (DB) is received up the layers.
Physical Layer
PL1
PL2
Physical Layer
001010101001110010010110101010
Router
0011110001110011100100101010011110101010

Figure 1.6: When a message is sent, headers are added at each layer as the packet moves down the layers of the Host. Upon receipt, headers are removed as the packet moves up the layers.

The Data Link Layer for the chosen NIC adds its MAC address as the source address in the *frame* header, and it adds the MAC address of the next device (Host 2 in Figure 1.6) on the packet's path as the destination address in the *frame* header. Upon receipt at its destination (Host 2), the packet is received at each layer moving up the layer stack. At each layer, the destination address is confirmed to be that of the host, the framing for that layer is removed, and the remaining packet is sent to the next higher layer.

1.6 Directly Connecting Computers on the Same Network

The simplest network consists of two computing devices that are directly connected to one another. When computers are located at the edge of the network, they are referred to as *hosts* or *end systems.* Note that when we are discussing network topologies and network graphs we refer to the network created by the directly wired and/or wireless communication links. Human centric links, such as a removable flash drive, do not typically create an edge in the network graph. This is because in computer communications we are concerned primarily with the ability of computers to communicate without human intervention.

Hosts, also referred to as **end systems**, are the computing devices at the *edge* of the network. Traditionally, the term 'host' or 'end system' refers to a computer or server; however, the advancement of technology with the resulting rise of the IoT means that even small embedded IoT devices can be hosts in a network.

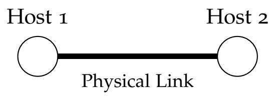

Figure 1.7: Two hosts directly connected in a network through either a wired or a wireless connection.

Figure 1.7 illustrates the simplest network graph with just two computers, or hosts, communicating. Using either a *crossover cable* to physically connect the hosts or a wireless connection, such as a *Bluetooth* connection, this two-host network forms the simplest usable computer-to-computer network, e.g., a wireless keyboard communicating with a computer using Bluetooth.

1.6.1 Wired Connections

A broad range of physical wired connections are used to directly connect computers and other devices to the network. Table 1.4 lists just a few of the common wired technologies in use today for computer networks. Notice that each wired connection type has different physical properties that impact the maximum bandwidth and data rates achievable over that medium. For this reason, communication protocols such as Ethernet (the most widely used set of wired LAN communication protocols) and WiFi (the most widely used set of wireless LAN communication protocols) that are designed for direct computer-to-computer communications are designed for a specific type of physical medium.

Ethernet is the most commonly used wired protocol. The Ethernet family of protocols defines how computers communicate directly with one another over LANs and MANs. Ethernet was initially developed in 1973 and 1974 by Robert Metcalf, David Boggs, Chuck Thacker and Butler Lampson at Xerox PARC. Ethernet was first standardized as IEEE 802.3 in 1983.

Table 1.4: Commonly used wired connection technologies.

Wired Technology	Notes
Twisted Pair	Twisted pair is the most common type of conductor in general wired installations. The cables consist of at least one pair of metal wires twisted together within a single sheath. Twisting mitigates electromagnetic interference from and to the wires. Cat 5 is the most common cable for use with 100Base-T and 1000Base-T Ethernet.
Coaxial Cable	Coaxial cable, or coax, is a general conductor containing an inner conductor surrounded by an insulator that is further surrounded by a conducting shield that operates to protect the inner conductor from electromagnetic interference and an outer insulating sheath. Coax is used as a transmission line for radio frequency signals and is commonly used for television but can be used for computer communications as well.
Power Lines	Power line transmission utilizes the existing wires for conducting power as a communication medium. Power line communications often cause significant interference and are susceptible to electromagnetic interference due to the unprotected nature of power transmission wires.
Fiber Optics	Fiber optic cables consist of a thin glass filament surrounded by a protective casing. Light signals are transmitted through the glass filaments for communication.

Table 1.5: Some Ethernet variants

Ethernet	Speed	Medium
10BASE5 ‘thicknet’	10 Mbit/s	Coax
10BASE2 ‘thinnet’	10 Mbit/s	Coax
10BASET	10 Mbit/s	T Pair
100BASET	100 Mbit/s	T Pair
1000BASET	1 Gbit/s	T Pair
10GBASET	10 Gbit/s	T Pair
10GBASEER	10 Gbit/s	Fiber

Table 1.5 lists some of the Ethernet standards that have been defined for wired device-to-device communications. ‘Classical Ethernet,’ encompassing both 10BASE5 (*thicknet*) and 10BASE2 (*thinnet*), was defined for use over coaxial cable where all devices shared the same physical wire. 10BASE-T defined Ethernet for use with twisted pair copper wires, and it defines both a half-duplex and a full-duplex

Simplex communication is one direction only, e.g., terrestrial radio.

Half-duplex communication occurs in one direction at a time.

Full-duplex communication occurs in both directions at the same time.

mode of operation. These early standards were designed to operate at 10 Mbit/s over the physical medium. Higher speed standards, e.g., 100BASE-T and 1000BASE-T, allow for even higher communication speeds over twisted pair copper wires. Even higher speed standards, such as 10GBASE-ER, allow for 10 Gbit/s communication speeds over several kilometers using fiber optic wires.

A wire medium provides a *guided medium*, meaning simply that the communication signals are constrained within the wire.

Wires provide a **guided medium** for communications.

1.6.2 *Wireless Connections*

All wireless communication technologies utilize the electromagnetic frequency spectrum for communications. While the frequency spectrum is continuous from 0 Hz upwards, the vast majority of wireless communications happens in only a few frequency bands as illustrated in Figure 1.8. The most widely used consumer frequency bands are the ISM (Industrial, Scientific and Medical) frequency bands. In general, the higher the frequency the higher the bandwidth available, but also the more power that is required to transmit that data a fixed distance compared to lower frequencies. Additionally, lower frequencies travel through solid matter much further and with lower *attenuation* than do higher frequencies.

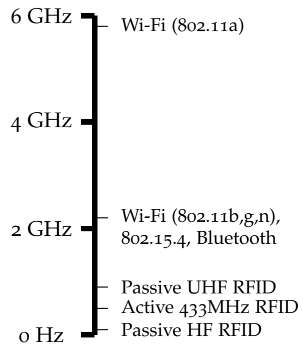

Figure 1.8: Commonly used ISM frequency bands and popular protocols that use them.

Advances in silicon manufacturing technologies and design techniques now allow consumer wireless network systems to operate at several tens to hundreds of gigahertz. Even with advances in silicon manufacturing technologies and design, most consumer wireless networks operate at frequencies less than 6 GHz.

The use of radiating devices is regulated within each country and coordinated globally by the **International Telecommunications Union** (**ITU**). The **Federal Communications Commission** (**FCC**) regulates the frequency spectrum in the US. The **Electronic Communications Committee** (**ECC**) within the **European Conference of Postal and Telecommunications Administrations** (**CEPT**) recommends frequency allocations within the European Union.

Many protocols, such as IEEE 802.11 and IEEE 802.15.4, are families of protocols defined to operate at multiple frequencies. Just as the wired protocols are designed for a particular physical medium, each wireless protocol is designed to operate within a particular frequency range. Variations in protocols for different frequencies are due to both the physics of the frequency range and the regulations that govern the usage of devices within each frequency range.

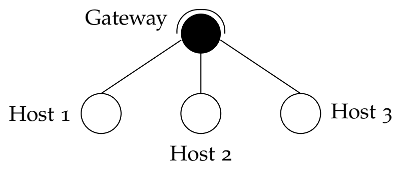

Figure 1.9: Simple star topology common in wireless protocols.

Wireless protocols determine to whom a particular device may communicate. WiFi, for example, defines that computers may communicate with the WiFi base station forming a star topology as shown in Figure 1.9. This communication limitation exists even

Wireless Protocol	Notes
Wi-Fi (WiFi)	WiFi is a local area network wireless communication technology using either the 2.45GHz and 5.8GHz ISM frequency bands. WiFi is a trademark of the Wi-Fi Alliance that, when marked on a product, indicates that the product is interoperable with other WiFi certified products. The WiFi protocol is based upon the IEEE 802.11 family of standards. WiFi has become synonymous with wireless LAN (WLAN).
Li-Fi (LiFi)	LiFi, short for Light Fidelity, is a local area network wireless communication technology using the visible light spectrum. LiFi may use infrared and near-ultraviolet frequencies in addition to visible light frequencies.
Bluetooth	Bluetooth is a personal area network technology designed to allow direct communications between devices using the 2.45GHz ISM frequency band. Bluetooth was originally based upon the IEEE 802.15.1 standard, but it is now managed by the Bluetooth Special Interest Group. Bluetooth Low Energy (BLE) was introduced in version 4.0.
IEEE 802.11	IEEE 802.11 is a family of wireless local area network protocols that define the LINK and PHY layers. IEEE 802.11 is the foundation of the WiFi family of protocols, and it is the primary foundation of most deployed WLANs.
IEEE 802.15.4	IEEE 802.15.4 is a family of personal area network protocols that define the LINK and PHY layers. IEEE 802.15.4 is the foundation of several ad hoc networking protocols including Zigbee and WirelessHART.
IEEE 802.16	IEEE 802.16 is a set of wireless broadband protocols for wireless metropolitan area networks that define the LINK and PHY layers. IEEE 802.16 is the basis of *WiMAX* and is managed by the *WiMAX Forum.*
LTE	LTE (Long-Term Evolution) is a set of wireless cellular telephony standards that allow for wireless data communications between mobile handsets and base stations and provides for an IP-based communication system.

Table 1.6: Commonly used wireless communication protocols.

though most computers talking to the same base station will be able to hear one another.

Table 1.6 introduces several common data communication protocols that use wireless communications for their physical medium. Most of these protocols operate in the 433 MHz, 915 MHz, 2.45 GHz and/or the 5.8 GHz ISM frequency bands.

1.7 *Connecting Computers on Local Networks*

Wired and Wireless connections form direct communication links between computers and networking devices and possibly other devices. In a wired connection, the most common computer connector is a Cat-5 cable that physically connects your computer to either a

Local communication refers to communication between devices connected to the same local network where the devices communicate at the two lowest levels of the Network Model.

bridge, a switch, a hub or a router to which other computers are physically connected. In a wireless connection, the most common connection utilizes Wi-Fi for general data networks and Bluetooth for direct device to computer networks. Wi-Fi networks allow for direct computer-to-computer connections but generally utilize a Wi-Fi gateway. The resulting logical communication configuration forms a star communication topology with the gateway at the center of the star.

The connection of two hosts through either a bridge, a switch or a hub is illustrated in Figure 1.10. Table 1.7 contains a description of bridges, switches and hubs. Simply stated, each of these network devices creates a logically connected local area network where the network device just forwards the communications received from one computer to all other computers directly connected to the network device. In this way, each of the computers connected to the network device appear as if they are on the same physical wire. Consequently, computers connected through these networking devices are able to directly communicate with one another using only the Physical Layer and the Data Link Layer protocols. Gateways and routers may also provide some bridge, switch or hub functionality.

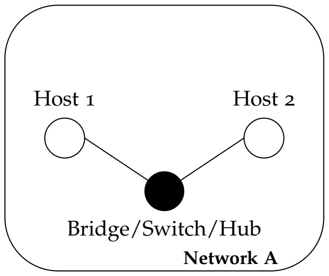

Figure 1.10: Two hosts connected through a bridge, a switch or a hub. The computers communicate as if they are directly connected when a bridge or switch is used.

Table 1.7: Bridges, Switches, Hubs, Gateways and Routers.

Network Device	Notes
Bridge	A bridge is a network device that physically and logically connects two or more network segments and allows devices on them to communicate as if they are directly connected to one another. A bridge may filter communications to minimize traffic on network segments to which a communication is not destined. All network segments utilize the same, or compatible, communication protocols.
Switch	A switch is a network device that physically and logically connects two or more network segments and allows devices on them to communicate as if they are directly connected to one another. A network switch is a multiport network bridge. A switch isolates each network segment, allowing for simultaneous communications on each without interfering with the other network segments. All network segments utilize the same, or compatible, communication protocols.
Hub	A hub is a network device that physically and logically connects two or more network segments and makes them into a single, larger segment. All network segments utilize the same communication protocols.
Gateway	A gateway is a network device that connects two network segments, each utilizing a different communication protocol, e.g., WiFi and Ethernet.
Router	A router is a network device that connects two or more networks, allowing them to communicate through the network by routing communications towards the correct destination network and device.

While bridges, switches and hubs physically and logically connect network segments, they operate in slightly different ways. *Hubs* connect different network segments thereby creating a larger network segment where all devices communicate using the same protocols. *Bridges* connect different network segments where the devices on a particular segment may use a different communication protocol than the devices on another segment. *Switches* are effectively larger more intelligent bridges that isolate the different network segments and may route the communication from one device to another device on a different segment only amongst those two segments instead of broadcasting the communication along all segments. *Gateways* isolate different network segments and operate to translate from one communication protocol into another communication protocol. *Routers* isolate different subnetworks and route communications between them as appropriate. Consequently, routers operate to route the communications from one device to its intended recipient on some other network or subnetwork.

Subnetworks are portions of networks that are independently *addressable* meaning that they contain devices with non-overlapping network, i.e., globally routable, addresses.

1.8 Connecting Computers on Different Networks

Computers on different local networks can communicate if their local networks are connected, either directly or indirectly. The connection between local networks typically occurs through either a router between subnetworks or through border gateway routers between networks. The use of routers and border gateway routers logically connects the (sub)networks, but they do not allow computers and other devices to communicate as if they are physically connected. Instead, communications between computers and devices on different (sub)networks must utilize the Network Layer, including the IP addresses contained therein.

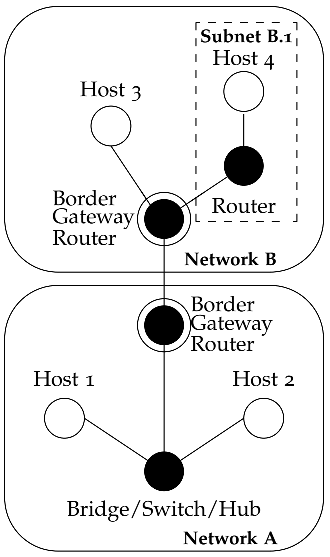

Figure 1.11: Hosts in different networks connected through border gateway routers.

Figure 1.11 illustrates a simple internet graph with two networks connected through border gateway routers with Network B containing a subnetwork, Subnet B.1. Each network has its own border gateway router since one of the functions of the border gateway router is to limit the communications that enter the network to only those communications destined for hosts and other devices in its network. A subnetwork does not require a router to be on each side of the communicating subnetworks. A single router can operate within multiple subnetworks simultaneously.

Border Gateway Routers execute the Border Gateway Protocol, or BGP. The BGP is a set of protocols that allow the administrator of a network to defined the policies regarding the admittance and transmission of all communications into and out of the network.

BGP is a critical component to managing and securing a network as well as allowing for the entire Internet to operate seamlessly and efficiently.

Network B of Figure 1.11 contains a *router.* The purpose of a router is to isolate two networks, much like the border gateway routers, such that the hosts and other devices on one side of the router do not appear to be physically connected to devices on the other side of the router. Thus, the router physically isolates the two *subnetworks,* while logically connecting them and allowing devices within each subnetwork to communicate independently and without interference from other subnetworks. The router performs packet routing functionality to move packets either to, or closer to, their intended destination. In the Network Layer, routers exist within a large domain, such as an Internet Service Provider, and operate to move network packets by communicating that packet between adjacent routers.

1.9 Communication Paradigms and Architectural Models

Computing devices form many different types of communication and functional relationships with one another. Hosts don't simply exist and operate independently of all other hosts. Computer networks are designed to facilitate communications which, in turn, allow for highly distributed functionality. How computing devices communicate and the type of network infrastructure that is established to support that communication is critical to the use of both the network and the devices connected through the network. Table 1.8 lists some basic architectural models commonly used in the design of networks and networked services and applications.

Table 1.8: Architectural Models

Model	Notes
Client-Server	Server performs functionality
Peer-to-Peer	Direct communication
Multi-Hop	Devices route communications
Tiered	Hierarchical network design
Internetwork	Connected networks

Interprocess Communication	**Remote Invocation**	**Indirect Communication**
Message Passing	Remote Procedure Call (RPC)	Distributed Shared Memory (DSM)
Sockets	Remote Method Invocation (RMI)	Group Communication
Multicast	Request-reply	Message Queues
		Publish-Subscribe

Table 1.9: Communication Paradigms

Interprocess communication refers to the low level communication between application processes.

Remote invocation refers to the two-way communication that results in the invocation of remote operations.

Indirect communication refers to communication through broadcast or other indirect techniques.

Table 1.9 summarizes the most common communication paradigms and techniques used over networks. The three main paradigms are *interprocess communication, remote invocation* and *indirect communication.*

There are many architectural models that support these communication paradigms. The most common architectures include *client-server, peer-to-peer, multi-hop networks, tiered networks* and *networks of networks.* We briefly examine each of these basic architectural models and communication paradigms in the following sections.

1.9.1 *Client-Server*

Perhaps the most common architectural model is the *client-server model.* This model was developed and evolved during the mainframe era, and it is the most common model for the centralized communication paradigm in use today. In a client-server system, the server contains resources, such as web pages, storage or computing resources, that are to be shared amongst several clients. In the client-server model, clients communicate indirectly with one another through the server, if at all. Clients invoke functionality in the server. Client-server models may be use with all three of the primary communication paradigms. A simple illustration of a client-server communication *remote invocation,* i.e., the complete process of sending a request and receiving a response, is shown in Figure 1.12. These communications may occur either directly between Client and Server or through a communication network.

A **server** refers to a *process* that accepts requests from programs running on other computers and performs a service or responds accordingly to the requests.

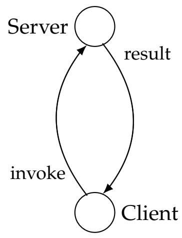

Figure 1.12: Client-Server Model. The client invokes functionality in the server, receiving the result in response.

1.9.2 *Peer-to-Peer*

In the *peer-to-peer* model, computing devices communicate directly with one another. In these direct communications, the peer hosts may operate in multiple roles such as source of and sink for communications. In addition to being a source and/or destination of communications, peer nodes often operate as network devices providing for routing functionality, storage, and other functionality as needed by the peer-to-peer network. Peer-to-peer networks may operate in a standalone fashion, such as an ad hoc sensor network, or they may operate over a broader *multi-hop network* or even an *internet,* such as the original file sharing program Napster and the voice and video application Skype. Peer-to-peer models may be use with all three of the primary communication paradigms. A simple illustration of a peer-to-peer communication is shown in Figure 1.13. Communications may be direct or they may be through a network.

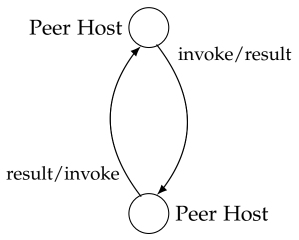

Figure 1.13: Peer-to-peer Model. Hosts operate in a similar manner with no distinction between client and server functionality.

1.9.3 *Multi-Hop Networks*

The *multi-hop network* paradigm allows hosts to communicate with other, non-adjacent, hosts. Hosts communicate through multiple routers (which may be other hosts acting as routers). In these systems, communications will hop from one subnetwork to the next through one or more routers. The routers use a *store-and-forward* paradigm in a *packet switched network.* Multi-hop models may be used with all three of the primary communication paradigms. A simple illustration of a multi-hop communication, with the communication traveling through a host or router, is shown in Figure 1.14.

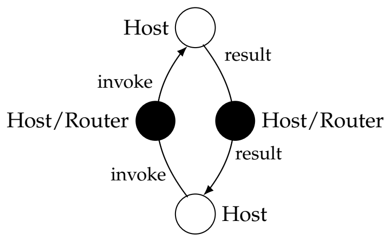

Figure 1.14: Multi-hop Model. Hosts operate in a similar manner with no distinction between client and server functionality. Communications may be routed through intermediate devices.

1.9.4 *Tiered Networks*

Tiered networks are used in the vertical layering of functionality and the physical organization of networks. Network applications, such as the World Wide Web, can have their functionality decomposed into discrete components, a Web server versus a Web browser, for example. In tiered network architectures, there is a mapping of application logic components to hosts. Hosts for an application may exist on the same subnetwork, on different subnetworks or even on different networks. Three or more tier networks are common for applications or components, such as large Websites, utilizing databases.

A common functional decomposition of network applications:

presentation logic: user interaction

application logic: app functionality

data logic: data management

1.9.5 *Networks of Networks*

Networks of networks, or *internets,* are used in the horizontal layering of functionality and the physical organization of networks. Internets are necessary to interconnect networks owned and operated by different entities. Internet Service Providers, or ISPs, provide for local, regional, national and global connectivity. ISPs are arranged in a layered architecture with local networks connecting to the local ISPs as illustrated in Figure 1.15.

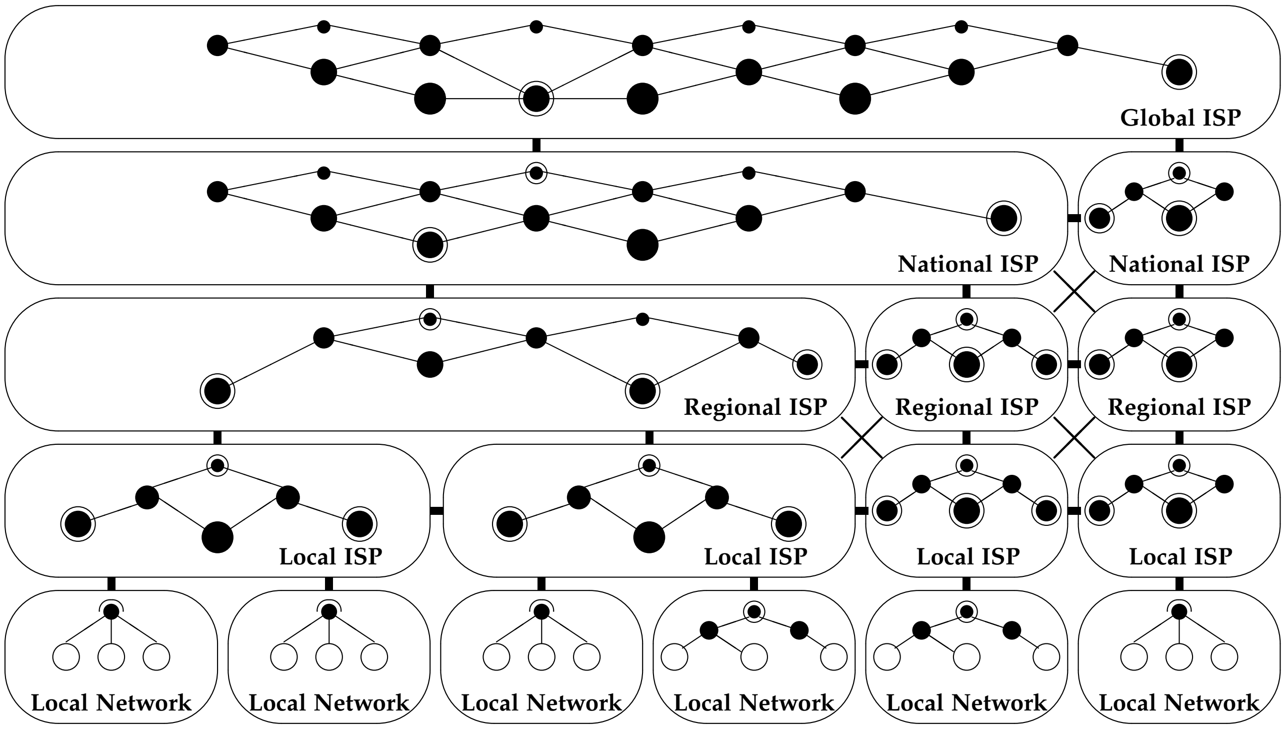

Figure 1.15: The layers of the Internet's Internet Service Providers (ISPs).

Distinct networks communicate through gateway routers. In routing packets from a source host to a destination host on a different network, the packets are routed to the gateway router most appropriate for the destination host. Hosts that are reachable from the same Local ISP will communicate up to the Local ISP, and the Local ISP will then route the packet down to the correct local network. Similarly, Local ISPs will route packets up the ISP hierarchy to the Regional ISP, and Regional ISPs will route packets up to the Global ISP. Within a particular ISP layer, the ISP networks may route to adjacent ISP networks if that adjacent network is able to route the packet down the ISP hierarchy to its destination. In this way, packets are routed from the source host up the ISP hierarchy as far as needed to be able to route down the ISP hierarchy to the destination host as illustrated in Figure 1.16. ISPs at a layer do not pass communications through themselves to other ISPs at the same layer. Communications are always passed up, down or sideways, but never from one side (arriving from a peer ISP) to another side (delivered to a peer ISP).

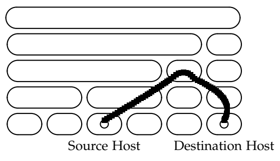

Figure 1.16: The packet is routed up and down the ISP hierarchy to reach its final destination.

1.10 *Packet and Message Network Delays*

The performance of a network is dependent upon a range of network characteristics including the physical characteristics of the network, the communication protocols used, the packet structure and number of bits and the communication traffic distribution and characteristics. The total performance of a network is measured both at the packet level and at the communication level. Each packet experiences a certain level of performance as it travels through the network. Multiple packets are sent for each message, and the total performance experienced by each packet determines the throughput experienced by the message.

Packet delay is the total delay experienced by a packet as it moves from its source to its destination. Some packets may be dropped resulting in an infinite delay.

Message delay is the total delay time from the sending of the first packet of the message to the correct receipt of the last packet of the message. Dropped packets must be resent if they are to be received at the destination.

Each packet experiences various types of delay as it travels through the network. The primary delay types experienced by a packet are listed in Table 1.10. The *nodal processing delay* is experienced at each host and each router due to the need to process each packet. *Queueing delay* is experienced by packets as it waits to be processed at a node. *Transmission delay* is experienced as each bit transmitted, i.e., 'put onto the wire,' from a node on its path. *Propagation delay* is experienced by each packet as it propagates through the medium to the next node on its path. For a specific wire or link, the propagation delay is fixed. The other delays are variable depending upon the number of bits in the packet (nodal processing and transmission) and the number of other packets at the node (queueing).

Table 1.10: Types of Delay for Packets

Delay Type	Time
nodal processing	variable (no. bits)
queueing	variable (no. packets)
transmission	variable (no. bits)
propagation	fixed (medium)

In addition to the various delays that a packet experiences as it travels through a network, packets may encounter transmission errors that render them unreadable and they may exist within the network for such an extended period of time that the network determines them to be undeliverable. In both of these cases, transmission errors and long lived packets, packets are dropped from the network and, therefore, never delivered. In these cases, packets must be resent; thereby, causing the message delay to be increased.

Packets may experience errors and extended delays that result in the packet being dropped from the network.

In addition to the delays experienced by packets that have been sent, messages may experience delays at the sending host. Many protocols, including the MAC protocols in the Data Link Layer and the end-to-end protocols in the Transport Layer, may intentionally delay sending a packet into the network. MAC protocols are designed to allow multiple devices to efficiently share the same medium, typically optimizing either average throughput or average *goodput.*

Throughput is a measure of the bits delivered by the network per unit of time. All delivered bits are counted towards the throughput measure.

Goodput is a measure of the useful, i.e., correctly received without errors or duplications, bits delivered by the network per unit of time. Framing bits, other overhead bits, incorrect or discarded bits and all retransmitted bits are excluded from the calculation of useful bits received.

The maximum throughput achievable along the path from the source host to the destination host is determined by the *minimum possible maximum throughput* along the links in the path. This *bottleneck link* is the link with the minimum transmission rate among all of the links on the path. Transport Layer protocols attempt to determine the throughput of this bottleneck link and transmit packets into the network at the same rate as the bottleneck link.

The **bottleneck link** limits the maximum throughput achievable along a path through the network.

1.11 Standards

A number of standards bodies exist to create standards related to computer networks. Table 1.11 lists the most influential standards bodies for computer networking standards.

Table 1.11: The most influential standards bodies in computer networks.

Standards Body	Notes
Internet Engineering Task Force (IETF)	The primary body for creating network related standards at all layers
Institute for Electrical and Electronic Engineers (IEEE)	Primarily creates standards related to the Data Link Layer and the Physical Layer
International Telecommunications Union (ITU)	Harmonizing frequencies and usage globally.
International Standardization Organization (ISO)	Process and low level standards related to networks, network management and network design.
World Wide Web Consortium (W3C)	Standards related to the World Wide Web and related applications.

1.12 *Summary*

Computer networks have a broad array of performance and capability characteristics. Special application networks, such as storage area networks, and localized networks, such as personal area networks, have been developed and deployed in addition to general purpose computer networks. These deployments include a broad range of architectures and usage scenarios with client-server architectures being the most prevalent.

Modern network designs are based upon packet switched store-and-forward architectures. By utilizing packets, communicating devices are able to quickly send communications when they are ready and to not send communications when no data or control communications are required.

Layering is used for network designs to mitigate complexity and enhance maintainability of the system. Layering occurs in the deployed networks and is common in the subnetwork designs within networks. Layering is the fundamental design approach for network functionality. The Network Layer is necessary for network to network communications. IP is the narrow waist of the Network Model, and it is critical for routing and global identification of hosts and other network devices. The Data Link Layer manages direct host-to-host communications. And, the Transport Layer manages host-to-host communications over the network.

Within the Network Model, each layer must manage the packets for that layer including handling any communication errors or performance limitations that may occur. Packets experience normal delays as they are communicated through the network. The primary variables that impact the overall message delays are the total number of packets in the network, impacting the queueing delay, and the bottleneck link throughput along the communication path. Medium access delays and communication errors further increase experienced delays and decrease the goodput of the network.

2 Network Security

Network Security is concerned with the authorized access to computing systems, network components and the communication channel including the data communicated therein. While network security is similar to computer security, and both are components of *information security,* network security is concerned with communications in addition to the security of the communicating computing systems whereas computer security is concerned primarily with the protection of a single computing device.

Primary Learning Objectives for this Chapter:

- Understand the basic network security concepts.
- Understand the primary security requirements.
- Understand the fundamental attacks experienced in computer networks.
- Understand and know how to use the basic security mechanisms.

In this chapter, we investigate the fundamental cyber aspects of computer and network security and examine the primary attacks experienced in computer networks, the primary mechanisms used to provide security and their application.

2.1 Overview

Computer and network security is concerned with the *confidentiality, integrity* and *availability* of computer and network system resources. These requirements are typically referred to as *CIA,* and they form the foundation of a secure system. In addition to these three security objectives, two additional security objectives are relevant for computer and network security: *authenticity* and *accountability.* We summarize these objectives in Table 2.1. Together, these five security objectives inform the types of *security attacks* that we expect, indicate the types of *security mechanisms* that we will use to mitigate attacks and their consequences, and even indicate the types of *security services* that are made available by the systems and the network.

Computer security: the protection afforded to an automated information system in order to attain the applicable objectives of preserving the integrity, availability and confidentiality of information system resources (including hardware, software, firmware, information/data, and telecommunications). – National Institute of Standards and Technology (NIST), *An Introduction to Computer Security: The NIST Handbook,* Special Publication 800-12, October 1995.

A broad array of both *passive* and *active* attacks are possible. Passive attacks include simple eavesdropping and traffic analysis while active attacks range from the very simple, such as a message replay, to the very complex, such as the intrusion into a system. Security mechanisms, such as *symmetric key cryptography, public key cryptography* and *cryptographically secure hash functions,* and protocols using them have been developed to mitigate vulnerabilities and

Threat: a potential for the exploit of a vulnerability.

Attack: an assault that typically attempts to exploit a vulnerability.

Objective		Notes
Confidentiality		preserving authorized restrictions on data and information access and disclosure
	Data Confidentiality	data is disclosed to authorized individuals and entities only
	Privacy	individual controls or influences what data and information related to them is collected and with whom it is shared
Integrity		guarding against unauthorized data or information modification or destruction
	Data Integrity	data, information and programs are changed in an authorized manner only
	System Integrity	system performs its functionality as intended without unauthorized manipulation of the system
Availability		ensuring timely and reliable access to and use of data, information and resources
Authenticity		property of being genuine of the claimed identity and being able to be verified and trusted
Accountability		actions of an individual or an entity are associated with that individual or entity and traceable to them

Table 2.1: Computer and network security key objectives – CIA-AA

FIPS PUB 200, Minimum Security Requirements for Information and Information Systems, National Institute of Standards and Technology, March 2006.

thereby eliminate a threat or significantly reduce the probability of a successful attack. *Security services* are provided by a system to implement and enforce security *policies* typically by using one or more security mechanisms.

The most secure systems are designed with clear *security requirements* defined from the beginning, with security implemented in each layer of the system architecture beginning with the hardware and including the operating system, communication protocols and all applications and protocols used within the system. These truly secure systems are the exception to the security design rule. Security requirements are typically defined late in the design process resulting in security being 'bolted on' to the system functionality instead of being designed into the system functionality. The more complex the system or its functionality and the later in the design process that security requirements are identified, typically the more vulnerabilities that exist and the more unidentified vulnerabilities that exist.

The vast majority of deployed computing systems and networks were designed with little to no security requirements being available early, if at all, during the initial design process. Network security was not a high priority and was never initially implemented in the design of modern packetized network communication protocols. Secure functionality has been adopted on top of a network architecture that was not designed with security requirements.

2.2 *Trust and Policy*

The foundation of computer and network security is *trust.* Computers, not humans, communicate over a computer network. Computers cannot see, hear, taste, smell or touch one another over the network. The computing devices don't know if the other device they are talking to is right next to them within the same chassis or on the other side of the continent connected over a very large WAN. Computers can only communicate with one another. And, this communication

Trust is the foundation of cyber security.

Over the network, your computer doesn't know if it's talking to you, your friend, a dog or an attacker. Therefore, *authentication mechanisms* are required to authenticate devices to one another both before and during communication.

occurs over a limited bandwidth physical link only some of whose properties the computing device can measure through its communications. Further still, it is quite typical that computers communicate with other computers not owned or controlled by the same entity. Internet communications occur over networks that are owned and controlled by still other (possibly unknown) entities. The Internet, and its many applications including the World Wide Web, are examples of computers, networks and even applications that are communicating or used for communications where the owners and managers of each of the network components are unknown to one, or even both, of the communicating computers.

Internet communications occur over untrusted networks with unknown owners and operators.

Ultimately, this limited communication bandwidth and use of resources controlled by other, unknown, entities, requires that the devices communicating over the network must perform some form of authentication during the communication process to ensure that the devices they are communicating with are who they claim to be and are authorized to be communicated with. The mechanisms used for this authentication, including public key certificates, symmetric key ciphers and secure hash functions, ultimately depend upon some form of trust, such as a trusted third party or trust that the other device or a shared secret has not been compromised.

Communication over untrusted networks requires the end devices that are communicating to authenticate themselves and secure communications in an end-to-end fashion.

While *trust* forms the foundation of cyber security, the definition of what *secure* means is defined by the *cyber security policies* of an organization. Regardless of what you may think *secure* means, the organizations security policy defines security. If an organization's security policy allows users to use any password, then the users are free to choose any password, even the most commonly used and easily guessed passwords. While I don't believe this to be secure (and, you may agree or disagree with me), this use of common or easily guessed passwords is secure for the organization that defines its password policy as 'any password allowed.'

Policy defines *security* for an organization.

The most common passwords used through the 1990's and 2000's were '12345' and '123456' according to SplashData.

In the end, a security policy represents a tradeoff between risks and costs. The likelihood of an attacker determining a user's common or easily guessed password is very high. All typical users are equally likely to be attacked, and all organizations are attacked. Therefore, the risk of a successful attack is very high with an 'any password allowed' policy. The costs are typically very high when an account is compromised. Conversely, the cost is very low to use a mechanism to enforce a password policy that forces users to use a broad array of alphanumeric characters and other special characters. The probability that an attacker obtains or is able to guess a password with this more stringent policy is significantly lower than with the 'any password allowed' policy. Therefore, for a little upfront cost of a password enforcement mechanism, the risk of a successful attack

An organization's *policy* represents its *choices between the risks* of successful attacks *and the costs* associated with both implementing and using the mechanisms that mitigate attacks and the actual costs associated with a successful attack.

Most organizations today have *password policies* that require complex, hard to attack passwords and utilize mechanisms to enforce their policies.

on a user's password can be significantly reduced and the overall expected cost (including losses due to compromised passwords) of the more stringent password policy is lower.

Computational power continues to increase at an exponential rate, primarily with the addition of parallelism. Quantum computing promises to make a revolutionary jump in computational power. Increases in computational power require longer and more complex passwords and longer security keys in order to prevent real-time brute force attacks.

Since I know that the risk of a successful password attack can be significantly reduced for a little up front cost, I believe that the 'any password allowed' policy is insecure. To be more accurate, I believe the 'any password allowed' policy makes a poor tradeoff of risk versus cost.

Security policies should be created to cover a broad range of both cyber and physical activities. Table 2.2 lists several policies that many organizations should, and often do, define.

Table 2.2: Example policies that may be defined for all organizations.

Policy Titles	
Acceptable Encryption Policy	Technology Equipment Disposal Policy
Acceptable Use Policy	Information Logging Standard
Clean Desk Policy	Lab Security Policy
Disaster Recovery Plan Policy	Server Security Policy
Digital Signature Acceptance Policy	Software Installation Policy
Email Policy	Risk Assessment Policy
Ethics Policy	Web Application Security Policy
Pandemic Response Planning Policy	Analog/ISDN Line Security Policy
Password Construction Policy	Anti-Virus Policy
Password Protection Policy	Server Audit Policy
Social Engineering Awareness Policy	Server Malware Protection Policy
Email Retention Policy	Removable Media Policy
Lab Anti Virus Policy	Virtual Private Network Policy
Security Response Plan Policy	Automatically Forwarded Email Policy
Wireless Communication Policy	Communications Equipment Policy
Acquisition Assessment Policy	Dial In Access Policy
Bluetooth Baseline Requirements Policy	Extranet Policy
Remote Access Policy	Internet Equipment Policy
Remote Access Tools Policy	Internet Usage Policy
Router and Switch Security Policy	Mobile Device Encryption Policy
Wireless Communication Policy	Database Credentials Policy
End User Encryption Key Protection Policy	Employee Internet Use Monitoring and Filtering Policy
Personal Communication Devices Policy	Workstation Security (For HIPAA) Policy
Remote Access Mobile Computing Storage	Mobile Employee Endpoint Responsibility Policy

2.3 *Securing the Network*

Communications between computers physically occur over wires and over the air with internet communications utilizing routers that often are neither controlled by nor known to either of the communicating parties. Attackers may exist at any point in the network from the wires and the air along a specific link to controlling the router or routers used for a communication. For this reason, the security model used for network security is one of two secure hosts, Alice and Bob, communicating over an insecure channel that both attackers Eve

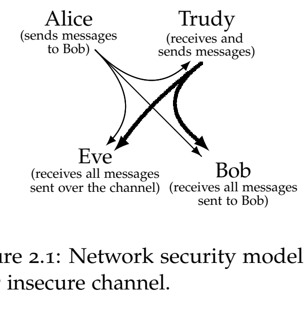
Alice
(sends messages to Bob)
Trudy
(receives and sends messages)
Eve
(receives all messages sent over the channel)
Bob
(receives all messages sent to Bob)

Figure 2.1: Network security model over insecure channel.

and Trudy have access to.

Eve may perform an array of *passive attacks* that all begin with *eavesdropping.* Eavesdropping allows Eve to capture all communications along a channel. She may then perform an array of analysis attacks including *traffic analysis* attacks. Eve may also perform a *message release* attack of all or some of the messages she captured. Table 2.3 lists the basic forms of passive attacks.

Table 2.3: Basic passive attacks

Passive Attack
Eavesdrop
Traffic Analysis
Message Release

Trudy may perform an array of *active attacks* in addition to the same passive attacks that Eve may perform. Active attacks involve either modifying otherwise valid packets in transit, masquerading as someone other than yourself, actively injecting packets into the communication network or preventing the legitimate use of a service or resource. Table 2.4 lists the basic forms of active attacks.

Table 2.4: Basic active attacks

Active Attack
Masquerade
Replay
Spoof
Message Modification
Denial of Service

Active attacks may come in many forms and have the potential to for more damage and cost more money to mitigate or recover from. Very specific active attacks exist to take advantage of specific vulnerabilities within a system. Furthermore, sophisticated active attacks may be difficult to detect as they are occurring.

Alice and Bob can protect themselves against both passive and active attacks by utilizing mechanisms that provide the CIA-AA functionality. The foundation mechanisms for CIA-AA are encryption algorithms, or *ciphers.* Two primary types of ciphers exist: shared symmetric (secret) key ciphers and asymmetric (public) key ciphers. Cryptographic *hash functions* provide secure *digital fingerprint* capabilities.

A **cipher** is an algorithm that is used to turn the plaintext of the message into the seemingly random sequence of bits forming the ciphertext (and back again from the ciphertext to the plaintext).

A **hash function** $H(\cdot)$ is one way function that creates a fixed length **message digest** h when given a variable length input message m ($h = H(m)$).

Symmetric key ciphers utilize a single shared secret key and provide both encryption and decryption functionality using that secret key. Public key ciphers utilize both a public key that is known by everyone and a private key that is known only by the *owner* of the public key. Both the public key and the private key may be used for encryption, but the opposite key used for encryption must be used for decryption.

Cryptography forms the basis of must security services provided by a system. Additional services are based upon cryptographic *hash functions.* Ciphers and hash functions, when used properly, can provide strong security functionality through an array of *security services.* Computing systems may provide these security services to enable secure functionality for applications and for communications. Typical security services flow from the CIA-AA requirements. Consequently, typical security services include *authentication, access control, confidentiality, integrity* and *nonrepudiation.* Table 2.5 lists these common security services provided by systems. All of these services rely upon either ciphers, hash functions or both ciphers and hash functions.

Service	Notes
Authentication	the assurance that the communicating entity is the one that it claims to be, specialized services include *Peer Entity Authentication, Data-Origin Authentication*
Access Control	the prevention of unauthorized use of a resource
Data Confidentiality	the protection of data from unauthorized disclosure, specialized services include *Connection Confidentiality, Connectionless Confidentiality, Selective-Field Confidentiality, Traffic-Flow Confidentiality*
Data Integrity	the assurance that data received contain no modification, insertion, delation or replay, i.e., they are exactly as sent by an authorized entity, specialized services include *Connection Integrity with Recovery, Connection Integrity without Recovery, Selective-Field Connection Integrity, Connectionless Integrity, Selective-Field Connectionless Integrity*
Nonrepudiation	provides protection against denial by one of the entities involved in a communication of having participated in the communication, specialized services include *Nonrepudiation Origin, Nonrepudiation Destination*

Table 2.5: Common Security Services

[ITU-T X.800] CCITT, Security Architecture for Open Systems Interconnection for CCITT Applications, Recommendation X.800, International Telecommunications Union, 1991.

[RFC 4949] R. Shirey, Internet Security Glossary, Version 2, RFC 4949, Internet Engineering Task Force, August 2007.

Authentication: sender and/or receiver confirm the identity of the other

Access Control: only authorized users can gain access to a resource

Confidentiality: only the sender and the intended receiver are able to understand the message contents

Integrity: ensure that the data or message is not altered either in transit or at rest

Nonrepudiation: a sender (receiver) should not be able to falsely deny that he sent (received) a message

2.4 *Symmetric Key Cryptography*

A *cipher* is an algorithm implementing a *trap-door one-way function* that takes as input a message m and a *key value* K and outputs a message m' that is a function of both m and K. As the type of the function suggests, given the output m' of a cipher, it is infeasible to retrieve the original message m without knowledge of the trap-door, which is the value of the key K for a *symmetric key cipher.* Therefore, a symmetric key cipher is a cipher that uses a single key value for both *encryption* and *decryption.*

A symmetric key cipher provides *confidentiality* and a simple form of *authentication.* When Alice and Bob share the same secret key K_S, Alice may encrypt a plaintext message $ct = E_{K_S}(pt)$ and send the ciphertext ct to Bob. Only with knowledge of the shared secret key K_S, Bob is able to decrypt the ciphertext to retrieve the plaintext $pt = D_{K_S}(ct)$. Due to the structure in the plaintext, such as the message being written in correct English, Bob is able to verify that the received message is correct and, therefore, authenticates that the message is from Alice (the only other person with the key K_S). Note that Bob and not the computer determines that the message is received and decrypted correctly. *Hash functions* (see Section 2.6) must be used in order for the computer to determine that the message was received and decrypted correctly. This is because the computer simply 'knows' about bits, and ciphers operate on bits. Hash functions provide the structural check that humans perform by reading the message

A good **symmetric key cipher** implements a *trap-door one-way function* utilizing a shared secret key K.

A **trap-door one-way function** is easy to calculate in one direction, but infeasible to calculate in the other direction without some special knowledge such as the shared secret key K.

Encryption is the process of encoding a plaintext message pt such that only an authorized entity can retrieve the original message from the encoded ciphertext message $ct = E(pt)$.

Decryption is the process of decoding a ciphertext message ct so as to retrieve a plaintext message $pt = D(ct) = D(E(pt))$.

A symmetric key cipher has four main elements:

1. Plaintext
2. Ciphertext
3. Cipher
4. Symmetric key

When used securely, protocols may add one or both of the following for the secure use of the cipher:

5. Initialization Vector (IV)
6. Mode of operation

i.e., hash functions provide integrity protection.

The simplest symmetric key cipher is a substitution cipher such as a *Caesar cipher.* In a simple Caesar cipher used for the English language, each of the 26 letters in the alphabet is assigned a number between 0 and 25 inclusive and sequentially in order with 'a' being assigned 0 and 'z' being assigned 25. Each plaintext letter is to be substituted by a different letter that is at a fixed offset distance from the plaintext letter. The offset distance is the key value K_S. For example, with a key value of 2, the plaintext letter 'a' is substituted by the ciphertext letter 'c'. The encryption function of the Caesar cipher is easily represented as $ct = (pt + K_S) \mod 26$.

The **Caesar cipher** has 26 possible key values ignoring case and 52 possible key values considering case. A simple *brute force* attack will find the correct key in milliseconds on a modern computer.

A **brute force attack** involves trying every possible key value until the key used to encrypt the message is found. On average, half of all possible key values must be searched to find the correct key value.

2.4.1 Stream Ciphers

A *stream cipher* is a cipher that functions by creating a *keystream* that is then combined symbol by symbol with the plain text to create the ciphertext. The same keystream is generated by the receiving device and is then combined with the ciphtertext to create the plain text. The combination is typically performed using the bitwise XOR function.

A **stream cipher** generates a pseudo-random keystream that is combined with text for both encryption and decryption functionality.

The *Vernam cipher* is an example of a simple stream cipher that uses a *one-time pad* as the key material. A one-time pad is a keystream that is as long as the message (in bits), that is a truly random sequence of bits, that is never used in whole or in part and that is kept secret. The use of a one-time pad as the keystream in a Vernam cipher has been shown to provide *perfect secrecy.*

The **Vernam cipher** was patented as U.S. Patent 1,310,719, issued on July 22, 1919 and issued to Gilbert S. Vernam for the XOR operation used to encrypt what became a **one-time pad**.

The *RC4* (*Rivest Cipher 4*) stream cipher, also known as the *ARC4* cipher, was designed by Ron Rivest in 1987. The cipher remained a trade secret of RSA Security until 1994 when a description of it was anonymously published. RC4 makes use of secret internal state that consists of a permutation of 256 bytes, two 8-bit index-pointers and a key of length up to 256 bytes (2048 bits). A *key scheduling* algorithm is used to initialize the original permutation of 256 bytes. RC4 does not use an LFSR allowing it to be implemented efficiently in software. RC4 has been found to be insecure allowing for both theoretical and practical attacks against both the cipher and various implementations and protocols that use RC4.

The first commercial use of RC4 was in Lotus Notes. RC4 was adopted for use in the WiFI WEP and WPA protocols and the SSL and TLS standards, although it is now prohibited for use in all versions of TLS according to RFC 7465. For a time, RC4 was (and may still be) the most widely used stream cipher in the world.

A description of RC4, its design, its weaknesses and a way to overcome them may be found in: *Ron Rivest and Jacob Schuldt, "Spritz – a Spongy RC4-like Stream Cipher and Hash Function," 27 October 2014. http://people.csail.mit.edu/rivest/pubs/RS14.pdf.*

2.4.2 Block Ciphers

A *block cipher* is a cipher that operates on a block of text at one time. The encryption and decryption functionality are distinctly different algorithmic operations, although they are typically the same or similar operations performed in reverse order. The first standardized block cipher is the Data Encryption Standard.

A **block cipher** is a deterministic algorithm that operates on a fixed sized *block* of data, taking as input both the block and a key, and performing a transformation of the block value based upon the key value.

The *Data Encryption Standard* (DES) was standardized by NIST (National Institute of Standards and Technology) on November 23, 1976. DES (pronounced *dez*) is a symmetric key *block cipher* based upon the *Feistel structure.* DES utilizes a 56-bit key (plus 8 parity bits) and operates on a 64-bit block of data. For encryption, DES performs an initial permutation of the plaintext block followed by 16 identical rounds of functions. Each round uses a different set of 48 bits of the key following a *key schedule.* A final permutation is applied after the 16 rounds to create the final ciphertext. Through a series of competitions in the late 1990's, the DES cipher was found to be insecure due to its small key size.

The DES functionality, and the Feistel structure specifically, has proven to be resistant to sub-brute force attacks. There are no known sub-brute force attacks against DES; however, modern computers can brute force the 56-bit key space (consisting of 2^{56} values) in less than six minutes with parallelization reducing this time by a factor equal to the number of processing cores.

3DES (pronounce *triple dez*) increases the effective key size of the DES cipher by utilizing DES three times in succession as shown in Figure 2.2. 3DES performs an encryption, a decryption and, finally, an encryption. By using a different key for each of the three DES operations, the resulting output utilizes 168 bits of keying material. It is also possible to utilize the same key for both encryption operations ($k_1 = k_3$) and a different key for the decryption step ($k_2 \neq k_1$). This results in 3DES having an effective key size of 112 bits. Backward compatibility with the basic DES cipher is achieved by setting all keys equal to one another ($k_1 = k_2 = k_3$). 3DES has been found to be insecure due to its 64-bit block size.

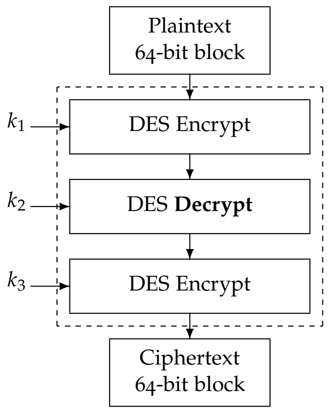

Figure 2.2: The 3DES cipher encryption process.

The *Advanced Encryption Standard* (AES) was established as the latest NIST cipher standard on November 26, 2001 with the publication of FIPS PUB 197 (FIPS 197). AES is a symmetric key block cipher based upon a *substitution-permutation network* that utilizes either a 128-bit, 192-bit or 256-bit key and operates on a 128-bit block of data. The two primary differences between these variants of AES are the key scheduler used to create the 128-bit round keys and the total number of rounds used in the encryption process. AES-128 uses 10 rounds. AES-192 uses 12 rounds. And, AES-256 uses 14 rounds.

AES has been extensively analyzed since its standardization in 2001. To date, only theoretical attacks have been found to determine the key value in sub-brute force time. The best such attack on AES-128 reduces the expected number of keys to be examined to $2^{126.0}$ versus 2^{127} for brute force attacks.

2.4.3 *Modes of Operation*

A block cipher must be utilized in a secure fashion in order to maintain security during its use. The most basic mode of operation is *Electronic Codebook* (*ECB*) *mode.* ECB mode is a straight application of a block cipher using the encryption function to encrypt plaintext and the decryption function to decrypt ciphertext as shown in Figure 2.3.

A **mode of operation** for a block cipher refers to how a block cipher is used.

ECB mode is not recommended for encrypting large quantities of data with the same key. A single block of data encrypted once with the same key will always yield the same ciphertext. Therefore, any structure or repeated values within a large block of data will become evident due to the repeated ciphertext.

ECB mode is the simplest mode of operation, and also the most easily used insecurely.

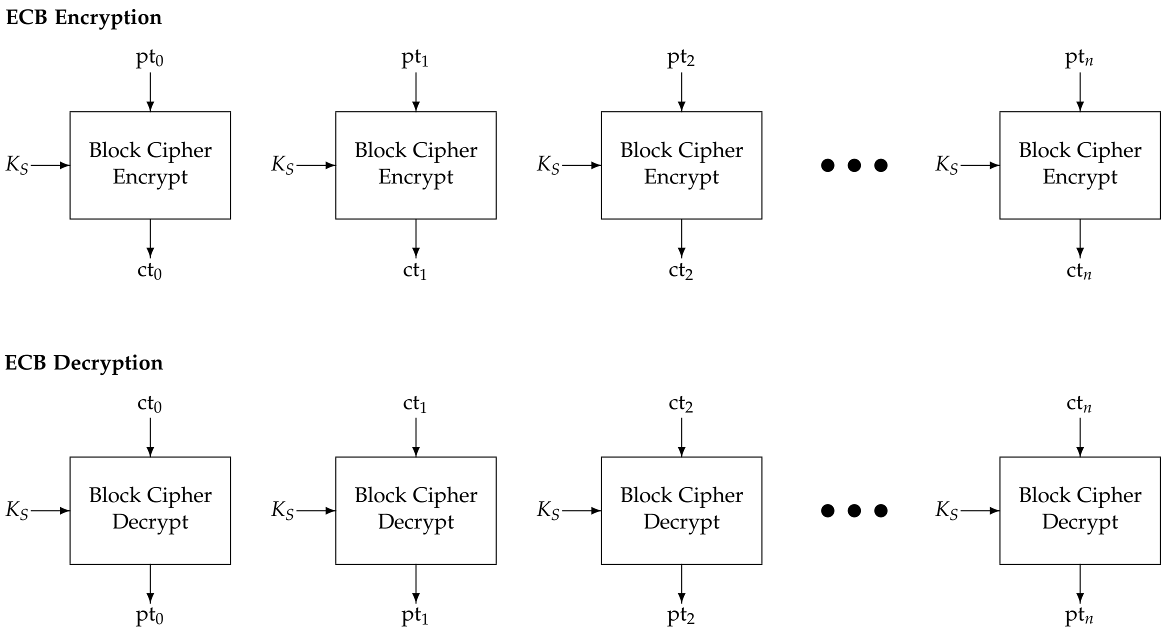

Figure 2.3: Electronic Codebook (ECB) mode of operation.
$\mathrm{ct}_i = E_{K_S}(\mathrm{pt}_i)$
$\mathrm{pt}_i = D_{K_S}(\mathrm{ct}_i)$

ECB mode might be used securely for small quantities of data, for non-repeated plaintext values, when secret keys are changed sufficiently often and for applications where it is secure for an attacker to understand that the same value is being repeated.

Cipher Block Chaining (*CBC*) *mode* entangles the ciphertext output from a first block with the ciphertext output of all subsequent blocks by XORing the ciphertext output of the first block with the plaintext input of the second block prior to encryption as shown in Figure 2.4. A unique *initialization vector* (*IV*) is XORed with the first plaintext block of the message to ensure that the same plaintext message yields different ciphertext even if the same key is used for encryption.

CBC mode was patented with U.S. Patent No. 4,074,066 being issued on February 14, 1978 to inventors Ehrsam, Meyer, Smith and Tuchman and assigned to IBM.

CBC mode is perhaps the most commonly used mode of operation for block ciphers. It is inherently sequential for encryption operations, but decryption may be performed in parallel. Due to the structure of the CBC mode, an error or modification of one block of ciphertext affects at most two blocks of plaintext when decrypted.

Cipher Feedback (*CFB*) *mode* is a *self-synchronizing* stream cipher mode for block ciphers. In CFB mode, the plaintext is not directly encrypted by the block cipher. Instead, the block cipher is used to create a *keystream* that is XORed with the plaintext to create the ciphertext as shown in Figure 2.5. The ciphertext from a first block is directly encrypted to create the keystream for the second block.

CFB mode generates a *keystream* that is XORed with the plaintext to create the ciphertext.

CFB mode uses only the *encryption* function of the underlying block cipher for both the encryption operations and the decryption operations. In this way, both the encrypt operation and the decrypt

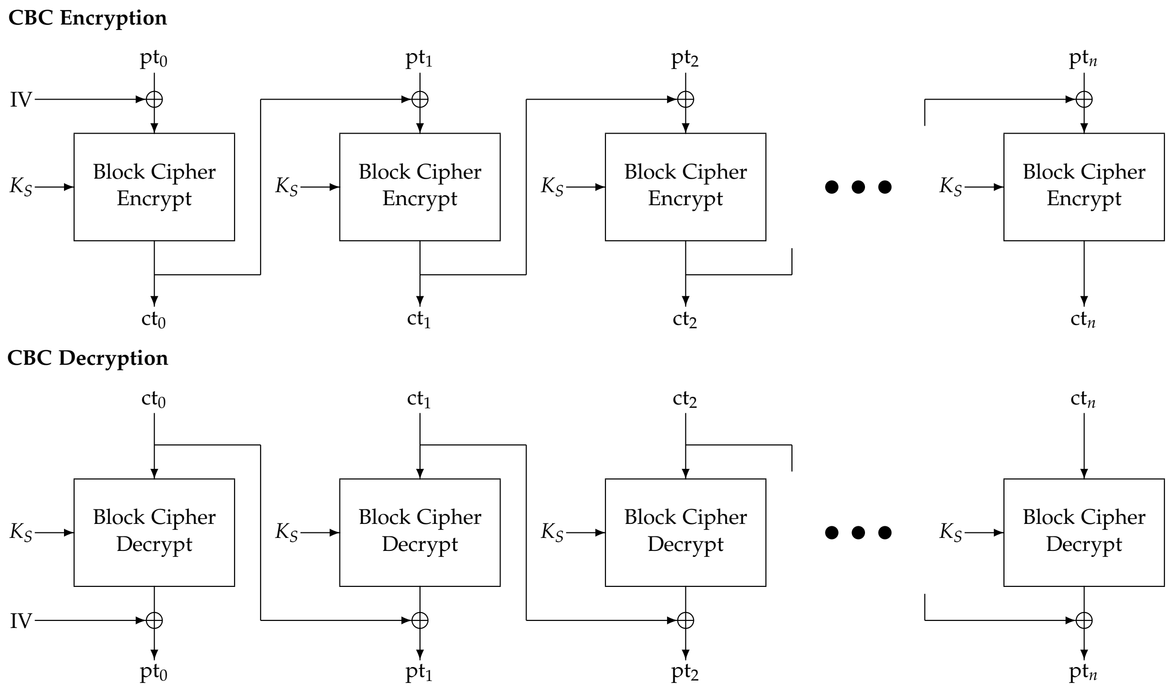
CBC Encryption
IV
pt_0 pt_1 pt_2 pt_n
K_S
Block Cipher Encrypt
ct_0 ct_1 ct_2 ct_n
CBC Decryption
ct_0 ct_1 ct_2 ct_n
K_S
Block Cipher Decrypt
IV
pt_0 pt_1 pt_2 pt_n

Figure 2.4: Cipher Block Chaining (CBC) mode of operation.

$$ct_i = E_{K_S}(ct_{i-1} \oplus pt_i)$$
$$pt_i = D_{K_S}(ct_i) \oplus ct_{i-1}$$
$$ct_{-1} = IV$$

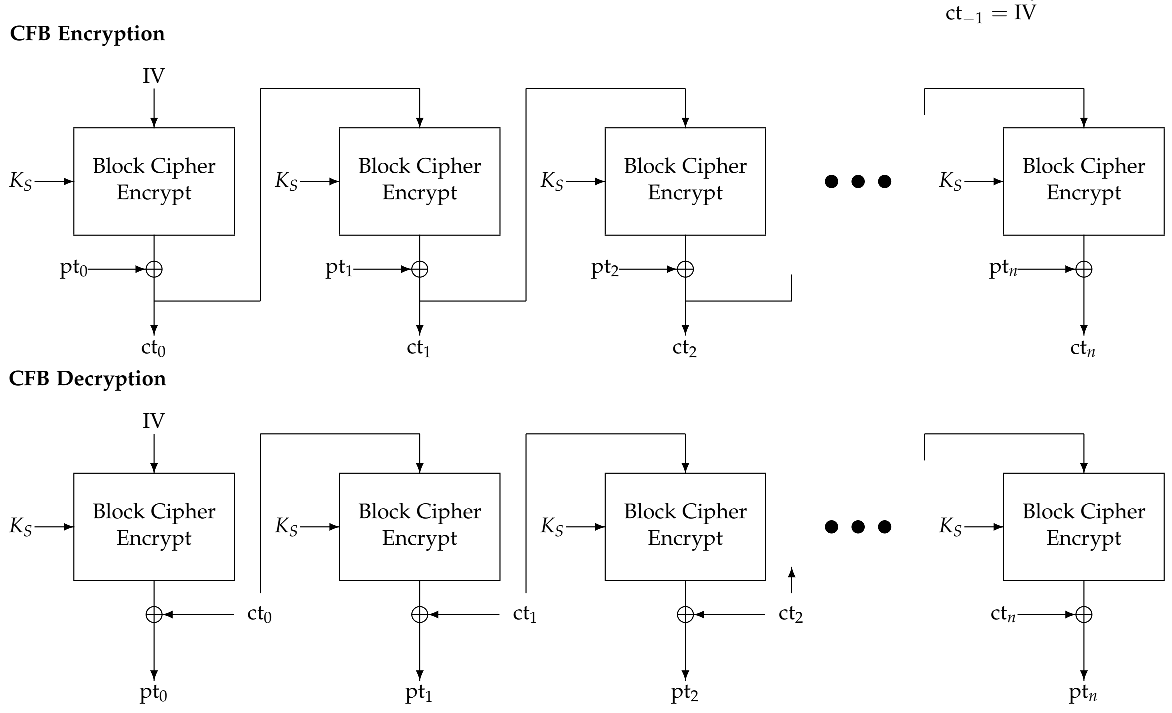
CFB Encryption
IV
K_S
Block Cipher Encrypt
pt_0 pt_1 pt_2 pt_n
ct_0 ct_1 ct_2 ct_n
CFB Decryption
IV
K_S
Block Cipher Encrypt
ct_0 ct_1 ct_2 ct_n
pt_0 pt_1 pt_2 pt_n

Figure 2.5: Cipher Feedback (CFB) mode of operation.

$$ct_i = E_{K_S}(ct_{i-1}) \oplus pt_i$$
$$pt_i = E_{K_S}(ct_{i-1}) \oplus ct_i$$
$$ct_{-1} = IV$$

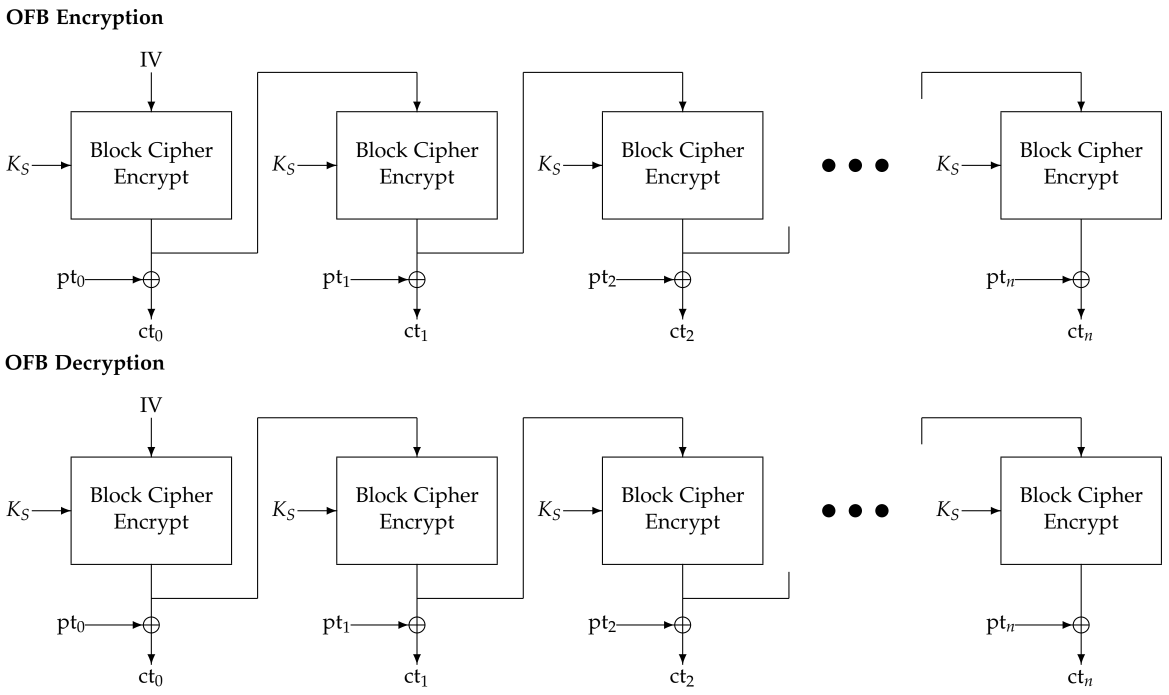

Figure 2.6: Output Feedback (OFB) mode of operation. Also referred to as Stream mode of operation.

$\mathrm{ct}_i = \mathrm{pt}_i \oplus O_i$
$\mathrm{pt}_i = \mathrm{ct}_i \oplus O_i$
$O_i = E_{K_S}(I_i)$
$I_i = O_{i-1}$
$I_0 = \mathrm{IV}$

operation utilize only the encryption function of the block cipher with the ciphertext as input. The encrypt operation is inherently sequential, just as with CBC mode. Similarly, the decrypt operations are inherently parallel, and an error or modification of one block of ciphertext affects at most two blocks of plaintext when decrypted.

Output Feedback (*OFB*) *mode,* also referred to as *Stream mode,* is a synchronous stream cipher mode of operation for block ciphers. In OFB mode, the plaintext is not directly encrypted by the block cipher. Instead, the block cipher is used to create a *keystream* that is XORed with the plaintext to create the ciphertext as shown in Figure 2.6. In OFB mode, the keystream block from a first block cipher encryption function is encrypted to create a second keystream block. Since the keystream has no dependence upon the plaintext, both the encrypt functionality and the decrypt functionality in OFB mode are identical.

OFB mode is also referred to as **Stream** mode.

A large number of modes of operation for block ciphers exist beyond the modes discussed here. Some of these modes, such as EAX mode and GCM mode, are used for *AEAD* (*Authenticated Encryption with Associated Data*) encryption functionality. Plaintext encrypted with an AEAD cipher can be decrypted and authenticated (i.e., integrity checked) simultaneously instead of requiring a separate integrity check function such as a hash.

2.4.4 *Security of Symmetric Key Ciphers*

An array of cryptanalytic attacks have been developed to attack the various symmetric key algorithms themselves. Vulnerabilities in the algorithms due to correlations between keys and outputs often lead to sub-brute force attacks against these algorithms. Some vulnerabilities can lead to real-time attacks against a specific cipher while similar vulnerabilities in another cipher may lead to only a slight reduction in the effort to retrieve the cryptographic key by analyzing and/or manipulating the outputs or operations of a cipher.

Developing a cryptographically strong cipher is difficult. A cipher must be strong against all attacks, while an attacker needs only one viable attack to succeed. Most ciphers that have been developed are easily broken. *Therefore, use only strong ciphers that have undergone extensive cryptanalysis.*

In addition to directly attacking the algorithm, either the ciphertext, the plaintext or both the ciphertext and the plaintext may be used in an attack. The most common types of attacks can be grouped based upon knowledge of or ability to manipulate the ciphertext and/or the plaintext. The resulting five groupings are shown in Table 2.6.

Attack	**Attacker Knowledge and Some Attacks**
Ciphertext only	Attacker knows the cipher used and the ciphertext. Attacker may perform brute force, statistical analysis attacks and replay attacks.
Known Plaintext	Attacker knows the cipher used, one or more plaintext-ciphertext pairs and additional ciphertext. Attacker may perform brute force, statistical analysis attacks, replay attacks and substitution pairings.
Chosen Plaintext	Attacker knows the cipher used, the plaintext-ciphertext pairs for plaintext chosen by attacker and additional ciphertext. Attacker may perform brute force, statistical analysis attacks, replay attacks, substitution pairings, insertion and modification attacks.
Chosen Ciphertext	Attacker knows the cipher used, the plaintext-ciphertext pairs for ciphertext chosen by attacker and additional ciphertext. Attacker may perform brute force, statistical analysis attacks, replay attacks substitution pairings, insertion and modification attacks.

Table 2.6: Types of attacks on encrypted messages.

A cipher is **computationally secure** if either of the following two conditions is met:

1. The cost of breaking the cipher exceeds the value of the encrypted information.

2. The time required to break the cipher exceeds the useful lifetime of the information.

2.5 *Public Key Cryptography*

Public key cryptography, or *asymmetric cryptography*, is a cryptographic approach that utilizes two different, but mathematically related, keys to perform both encryption and decryption operations. In asymmetric cryptographic systems, one key is made public, the *public key*,while the other key is kept private, the *private key*,and never shared with any other entity.

An **asymmetric (public key) encryption scheme** has six main elements:

1. Plaintext
2. Ciphertext
3. Public key
4. Private key
5. Encryption algorithm
6. Decryption algorithm

In a strong asymmetric cryptographic system, it is computationally infeasible to determine the private key, Pr, given only knowledge of the public key, Pu, and the cryptographic algorithm. Furthermore, in some asymmetric algorithms, such as RSA, either of the keys may be used for encryption with the other key being used for decryption. In effect, a public key cipher is a *trap door one-way function* where the trap door is enabled by the second key. A *trap door one-way function* is a function that is easy to calculate in one direction and infeasible to calculate in the other direction unless certain additional information is known. A public key cipher is an example of a trap door one-way function.

RSA key pair is created as follows:

1. choose two large prime numbers p and q, e.g., where each is 1024 bits in length.
2. compute $n = pq$ and $z = (p-1)(q-1)$.
3. choose e where $e < n$ that has no common factors with z, i.e., e and z are relatively prime.
4. choose d such that $ed - 1$ is exactly divisible by z, i.e., $ed \bmod z = 1$.
5. The public key is (n, e).
   The private key is (n, d).

RSA operations:

RSA public key operation $Pu(\cdot)$: Given public key (n, e) operate on message m $(< n)$ by computing

$$Pu(m) = c = m^e \bmod n$$

RSA private key operation $Pr(\cdot)$: Given private key (n, d) operate on ciphertext c by computing

$$Pr(c) = m = c^d \bmod n$$

Order of operations: Operation order may be interchanged

$$Pr(Pu(m)) = m = Pu(Pr(m))$$

Each key has a role to play in both encryption and decryption. When the public key is used for encryption, the resulting message provides for *confidentiality* since only the holder of the private key associated with that public key may retrieve the original message from the ciphertext. When the private key is used for encryption, the resulting message provides for *non-repudiation* since only the holder of the private key could have encrypted the message, but everyone that has the associated public key is able to decrypt the ciphertext to recover the message.

For many public key algorithms, including RSA, it is theoretically possible to make the 'private key' public while simultaneously keeping the 'public key' private. That is, it is theoretically possible to switch the roles of the two keys. In practice, however, it is generally not feasible to switch the roles of the two keys. This is because the public key values, e.g., the RSA public key exponent, are often sufficiently small to be amenable to real-time brute force attacks while the private key values are sufficiently large as to be computationally infeasible to determine.

The basic order of operations for secure communications using just public key ciphers is as follows:

1. Each entity generates a public-private key pair.
2. Each entity places their public key in a public key registry.
3. Bob retrieves Alice's public key from the registry.
4. Bob encrypts his message for Alice using his private key, st $= E_{Pr_{\text{Bob}}}(pt)$. This creates a *digitally signed* message st.
5. Bob encrypts his signed message for Alice using Alice's public key, ct $= E_{Pu_{\text{Alice}}}(st)$.
6. Bob sends the ciphertext ct to Alice.
7. Alice receives a message ct′ from Bob.
8. Alice retrieves Bob's public key from the registry.
9. Alice decrypts the received message using her private key, st′ $= D_{Pr_{\text{Alice}}}(ct')$.
10. Alice decrypts the message st′ using Bob's public key, pt′ $= D_{Pu_{\text{Bob}}}(st')$.

A **public key registry** must operate as a *trusted third party* and perform both identity and private key ownership authentication.

A **digital signature** is created by performing a private key operation over some (plain or cipher)text.

Public key retrieval is only secure if the public key party operates as a trusted third party.

Through this process, Bob has sent a confidential message, signed by himself, to Alice. Alice has decrypted the message and knows that the message pt′ could have only come from Bob. (The digitally signed message could have been done only by Bob.) However, the only way that Alice knows she has received a correct message from Bob is if she is able to interpret pt′ as a valid and correct message, e.g., a valid English message.

Note that message *integrity* is not provided by public key ciphers; it is provided, in part, by the structure of the message being sent.

2.6 Hash Functions

Hash functions are *one-way functions* that take a variable length input m and produce a fixed length output h that is also referred to as a *message digest.* Table 2.7 lists the primary requirements for all cryptographic hash functions.

Hash functions are used to provide message *integrity protection* by sending $m||H(m)$ and upon receiving $m'||H(m)'$ calculating $H(m')$ and comparing $H(m)' == H(m')$.

Requirement	Description
Variable input size	$H(m)$ can data as input any sized message m
Fixed output size	$H(m)$ produces the same size output h regardless of the size of m
Efficient	$H(m)$ can be calculated easily for any value of m
Pseudorandom output h	h meets standard tests for pseudorandomness
Preimage resistant (one-way property)	For any given hash value h, it is computationally infeasible to find a message n such that $H(n) = h$
Second preimage resistant (weak collision resistant)	For any given message m, it is computationally infeasible to find $n \neq m$ with $H(n) == H(m)$
Collision resistant (strong collision resistant)	It is computationally infeasible to find any pair (m, n) such that $H(m) == H(n)$

Table 2.7: Cryptographic Hash Function $H(\cdot)$ Requirements.
A **weak hash function** satisfies the first six of the seven requirements.
A **strong hash function** satisfies all seven of the requirements. Collision resistance is a subset of second preimage resistance that can mitigate chosen message attacks.

The *birthday paradox* arrises in collision attacks due to the non-intuitive high frequency at which two messages may be found to have the same hash value. In the birthday paradox, if we choose values uniformly at random from the range 0 through $N-1$, then the probability that a repeated element is encountered exceeds 0.5 after just $\sqrt{N}$ choices have been made. Thus, for an y-bit hash value, if we pick messages at random, we expect to find two messages with the same hash value with a 0.5 probability within $2^{y/2}$ attempts.

Table 2.8: The **Birthday Paradox** in numbers for a 50% chance of two messages having the same hash value.

No. Bits	No. Expected Searches
16	300
32	$77,000$
64	5.1×10^{9}
128	2.2×10^{19}
256	4.0×10^{38}
512	1.4×10^{77}

The most common cryptographically secure **hash functions** are *SHA-2* and *SHA-3*. Other common but less secure hash functions include *SHA-1* and *MD5*.

2.7 Message Authentication Code

Message *integrity* is critical to communications. More specifically, being able to automate the integrity check for a message is critical to communications. A simple approach to message integrity is to simply send a message m and its hash $H(m)$, i.e., send $m||H(m)$.

Clearly, sending a message and its hash value is not secure. Trudy can easily modify a message from Bob, calculate a correct hash value for the message and send both her modified message and calculated hash to Alice. Alice will verify the message integrity and not be able to detect that any message modification occurred. Since message integrity is supposed to be able to detect both intentional and unintentional message modifications, we need a secure approach for message integrity protection.

A *Message Authentication Code* (*MAC*), also called a *Message Integrity Code* (*MIC*), provides for secure message integrity detection. By using a shared secret *authentication key,* K_{Auth}, and hashing the key prepended to the message, Alice and Bob are able to detect any intentional modifications caused by Trudy. Therefore, by using the shared authentication key, Bob will send the message m and the MAC $H(K_{\text{Auth}}||m)$ to Alice, i.e., bob sends $m||H(K_{\text{Auth}}||m)$. Since only Alice and Bob know the shared authentication key, Alice will be able to verify the MAC she received. And, if verified, that the message came from Bob.

A **Message Authentication Code** (**MAC**) uses a shared secret **authentication key** and cryptographic hash function to provide for secure *message integrity,* i.e., the MAC $h = H(K_{\text{Auth}}||m)$.
Bob sends:

$$\text{packet sent} \quad = \quad m||h \qquad (2.1)$$

Alice receives:

$$\text{packet received} \quad = \quad m'||h' \qquad (2.2)$$

Alice determines integrity by evaluating the truth of this equation:

```
If    $(h' == H(K_{\text{Auth}}||m'))$   (2.3)
      then Authentic
      else Not from $K_{\text{Auth}}$
```

The most common message authentication code is the *keyed hash-based message authentication code,* or *HMAC,* defined in RFC 2104. HMAC uses two nested hash operations with a single shared authentication key to achieve a high level of security. The **HMAC** function is defined as:

$$\text{HMAC}(K, m) \quad = \quad H((K' \oplus opad)||H((K' \oplus ipad)||m)) \qquad (2.4)$$

where $H(\cdot)$ is a cryptographic hash function, m is the message to be authenticated, K is a secret key, K' is the secret key derived from the original key K, $opad$ is a constant $0x5c5c \ldots 5c5c$, and $ipad$ is a constant $0x3636 \ldots 3636$.

2.8 Digital Signatures

A *digital signature* is used as a way to demonstrate that the 'signer' has knowledge of a secret key. Digital signatures are typically created using public key cryptography by performing a private key operation. Since only one entity, e.g., Bob, has knowledge of their private key, only that entity could have created a valid signature. Since only they signer has knowledge of their private key, anyone using the

A **public key digital signature** simply says that the claimed owner of a public key has the associated private key. The signature by itself *does not* provide identity authentication.

public key disseminated by the signer is able to verify that the signer had knowledge of the corresponding private key. Note that a public key signature does not authenticate the identity of the signer. The signature merely authenticates that the signer has knowledge of a secret key, the private key in the public-private key pair.

A digital signature provides for **non-rediation**. That is, the signer cannot plausibly deny having signed the message.

If a **trusted third party** has performed identity authentication, then through our *trust* in the trusted third party, we know the identity of the entity that signed a message.

In a symmetric key signature, the signature is generated by a trusted third party using a secret key known only to that trusted third party. The message encrypted with the trusted third party's secret key can only be verified or decrypted by the trusted third party since the trusted third party does not share its signing key with any other entities. Note that a symmetric key signature does not authenticate the identity of the signer (the trusted third party) or the entity on whose behalf the trusted third party signed the message.

We make the distinction between identity authentication, e.g., authenticating that an entity is Bob, and the authentication operation provided by a digital signature. A public key digital signature allows us to authenticate that the signing entity has the private key associated with their claimed public key. This is the extent of our authentication capabilities with just a digital signature. If the claimed public key is obtained from a public key registry, i.e., a trusted third party, that performs identity authentication prior to allowing public keys to be posted, then through *trust* in the registry, we know that a verified signature came from a particular identified entity.

2.9 *Certificates*

Certificates were originally defined for public key cryptography as a way to obtain securely an identity's public key without needing to be online to retrieve it from the public key registry. The *purpose* of a public key certificate is to associate an identity with a public key. This is the only function that public key certificate performs.

The *purpose* of a **public key certificate** is to bind a public key to an identity.

A public key certificate is a file issued by and signed by the *Certification Authority* (*CA*) that issued the certificate. The certificate *binds* the identity in the certificate to the public key identified in the certificate. Like the public key registry, the CA is a trusted third party that is expected to have verified the identity of the entity identified in the certificate, and the CA is expected to have verified that the identified entity has the private key associated with the public key in the certificate. The existence of the certificate indicates that the Certificate Authority performed some form of due diligence to ascertain that the requester of the certificate is the entity identified in the certificate and that the identified entity has control and access to the private key corresponding to the public key identified in the certificate.

Table 2.9: Contents of an X.509 Certificate.

X.509 Cert Contents
Certificate
Version Number
Serial Number
Signature Algorithm ID
Issuer Name
Validity Period
Not Before
Not After
Subject Name
Subject Public Key Info
Public Key Algorithm
Subject Public Key
Issuer Unique ID (opt)
Subject Unique ID (opt)
Extensions (opt)
Certificate Signature Algorithm
Certificate Signature

The standard public key certificate illustrated in Table 2.9 is the

ITU-T X.509 certificate. The X.509 standard was first released on July 3, 1988. RFC 5280, released in May 2008, and RFC 6818, released in January 2013, define the IETF's version of the X.509 *Public Key Infrastructure (PKI) Certificate and Certificate Revocation List (CRL)* profiles.

A public key certificate may bind an identity to **attributes** in addition to a public key.

The X.509 standard assumes a hierarchy of certificate authorities for issuing certificates. This hierarchy results in a *certificate chain.*

A **certificate chain** is trusted if and only if every entity in the certificate chain is trusted. Simply verifying every certificate in the chain does not convey trust that every entity in the chain is who they claim to be.

In order to verify a certificate, the signature of the certificate (and every certificate in the chain) must be verified. The top level certificate authorities have well known public keys. These keys are typically embedded in a *root of trust* within your computer. All public key values in your root of trust are explicitly trusted to be valid, and they are used for certificate authentication purposes amongst other purposes.

Root of trust are public key-identity pairs that are explicitly trusted by a computing system. Modern browsers typically have more than 100 public key-identity pairs contained in their root of trust.

Self-signed certificates should be suspect unless the certificate was obtained securely from the trusted entity that signed the certificate.

In a certificate chain, if any entity in the chain is *untrusted,* then the chain after that entity should be untrusted. Even though every certificate in a chain can be verified to be correct, an untrusted entity can create a certificate for an attacker thereby providing a false identity for the attacker.

2.10 Network Security Devices

Firewalls provide for a single point along a communication path through which all communications are routed. The basic functionality of a firewall is designed to prevent intruders from getting in and unauthorized data from getting out. The firewall inspects all traffic through it allowing for *packet filtering* of both incoming and outgoing packets.

Firewalls are systems (preferred) or application software designed to secure access to an internal system or network primarily through **packet filtering**.

Packet filtering may occur based upon source or destination IP address, by using a white list, a black list or both a white list and black list of IP addresses. Filtering may also occur based upon a particular port that is being accessed. For example, if the `finger` function is not supported or not allowed on protected systems, then the firewall should block all traffic attempting to connect to port 79. Finally, filtering may occur based upon *deep packet inspection* that examines the contents of a packet to determine if the communication is allowed or not.

Packet filtering provides for extensive security; however, not all attacks may be mitigated with packet filters. *Denial of Service* (*DoS*) attacks and *Distributed Denial of Service* (*DDoS*) attacks, for example, are not thwarted by firewalls, and in some cases may be abetted by the firewall.

In order to overcome some of the limitations of firewalls, *proxy servers* may be used as the public facing accessible computer for the network. Proxy servers, or simply *proxies,* operate on behalf of the network and act as a gatekeeper to the services contained behind the firewall. In a common setup, in order to communicate with a device protected by the firewall, communication is first established with the proxy that is outside of the firewall. The proxy then communicates through the firewall on behalf of the entities outside of the firewall.

In this way, proxy servers provide for a level of indirection and obfuscation that provides additional protection for devices behind the firewalls.

2.11 *Summary*

Security, cybersecurity in particular, is a critical component to the safe and continued operation of the Internet and to computer systems and networks generally. Consequently, security needs to be a primary requirement for all computing systems and networks, especially during the original design process. The most commonly used network protocols, however, were designed for a layered system design that was developed without a strong security requirement. Secure protocols have been developed for use at each of the layers, such as TLS at the Transport Layer and IPSec at the Network Layer. These protocols have proven to provide significant protection, and all of them are built upon a basic set of security mechanisms.

Cryptographic ciphers are the foundational mechanisms for cybersecurity. Symmetric key ciphers may be used to provide for confidentiality and a limited form of authentication. By encrypting a plaintext message using a strong cipher such as AES, only the intended recipient that has the same key used to encrypt the message may decrypt the message using that shared secret key. The nature of the key being secret and the message being coherent allows for the cipher to provide a basic form of authentication in addition to confidentiality. The shared secret key nature of these ciphers requires key management support to securely use these ciphers.

Public key ciphers, particularly when used with certificates and other PKI services, overcome the basic symmetric key management and distribution problem. By using certificates to bind identities to public keys and sharing the certificates, secure communications may be created over an insecure channel. Use of the public key of a message recipient ensures confidentiality since only the recipient has the private key that can decrypt the ciphertext to retrieve the encrypted message. Use of the private key of a message sender ensures nonrepudiation since every can verify the resulting digital signature by using the sender's public key.

Message integrity is provided through the use of a cryptographically secure hash function such as SHA-2 or SHA-3. The hash function applied to a message creates a message digest that operates as a fingerprint of the message. By sending a message and its hash value, the receiver is able to calculate the hash value of the received message and verify that the received hash value is the same as the calculated hash value. By using a keyed hash function such as an HMAC, the receiver is able to both verify the message integrity and authenticate the identity of the message sender. The shared secret nature of a keyed hash function requires a shared secret key be created prior to verification.

In addition to these basic cryptographic ciphers and hash functions, a number of additional systems, protocols and mechanisms have been created to to provide all of the CIA-AA properties. Firewalls, proxies, password systems and secure protocols such as TLS and IPSec have all been developed to provide a level of attack prevention and attack detection. Additional systems and accepted practices may be used for attack recovery.

Cybersecurity is a difficult problem that will persist for computing systems and computer networks for the foreseeable future. The limited security requirements in the basic design of computer networks has led to complex security solutions being built on top of these insecure systems. The resulting need to trust more than simply the hardware to ensure security combined with the extreme complexity of today's computing systems and computer networks ensures that security functionality will continue to evolve and remain a critical component of computer networks.

3 The Physical Layer

4 *The Data Link Layer*

5 The Network Layer

6 *The Transport Layer*

7 The Application Layer

Network applications take advantage of the inherent parallelism and concurrency of computer networks and the hosts connected to them in order to perform a broad range of distributed services and perform distributed applications. The ability of hosts to operate concurrently while simultaneously being able to communicate reliably enables a broad range of functionality and services that is simply not possible with a single host. The underlying reliable communication capabilities of networks, particularly in and through the Internet, has fundamentally changed the way we think about problems; moving from the traditional sequential approach to a highly parallel, and often data, computation and communication intensive, approach. The ability to reliably communicate to massive computation and storage resources allows for cheap inexpensive sensors and interface devices at the edges of the network that communicate through the network to hosts that perform the functional computations and data storage on their behalf.

Primary Learning Objectives:

- Understand the basic concurrency and transparency requirements of network applications.
- Understand how network architectures are used by network applications.
- Understand the need for and use of security mechanisms and automation within network applications.

In this chapter, we explore the fundamental concepts and designs of network-based applications and distributed systems and briefly review basic services and widely used applications.

7.1 Overview

Distributed computing and distributed applications have become utilities due to the widespread deployment and use of high performance computer networks. Software-as-a-Service (SaaS) and all of its many *as-a-Service* cousins have enabled a rebirth and widespread adoption of a Client-Server model that is reminiscent of the original mainframe designs. This rebirth is possible only because of the high speed, reliable communications that are possible over the Internet from nearly everywhere within our modern, civilized world.

Cloud computing is used to refer to services that are provided by shared servers accessible over the network.

A **cluster computer** is commonly used to provide cloud computing functionality from a set of interconnected computers that provide the appearance of a single high performance computing host.

In distributed systems, the programming, use, operation and concurrency of these systems should be transparent to the users and developers of the distributed systems. Concurrency is a major tenet

Concurrency refers to the ability of different applications and different operations within the same application to operate at the same time.

of parallelism, and all network-based and distributed systems that have concurrency must be designed to operate correctly in a concurrent, shared resource environment. Transparency is necessary to allow the system to be perceived as a single system instead of as a collection of interacting components. ISO, in their basic reference model of open distributed processing, defines eight types of transparency that are summarized in Table 7.1.

Transparency refers to making certain aspects of a distributed system's operations invisible to the user and the application programmers.

Transparency Form	**Notes**
Access Transparency	resources are accessed in the same manner regardless of their locality to the accessing process
Location Transparency	resources are accessed without regard to their location
Concurrency Transparency	multiple processes operate at the same time on the shared resources
Replication Transparency	multiple instances of resources may be created without the knowledge or explicit use by users and application programmers
Failure Transparency	non-critical hardware and software failures are automatically recovered from and are not visible to the user and the application programs using the distributed system
Mobility Transparency	resources and processes may be moved within a system without affecting the operation of users or programs
Performance Transparency	system is reconfigurable to improve performance without impacting users or application programs
Scalability Transparency	system and applications may scale either up or down in resource usage without changing either the system structure or the application algorithms

Table 7.1: Forms of transparency defined in *Basic Reference Model of Open Distributed Processing, Part 1: Overview and Guide to Use*, ISO/IEC JTC1/SC212/WG7 CD 10746-1, 1992.

Network Transparency is the combination of **access transparency** and **location transparency**. Network transparency most strongly affects the usability and utilization of distributed resources.

Network applications provide concurrency and transparency within the context of a chosen architectural model. Client-server architectures are commonly used due to their resemblance to most distributed application operations and their ease of programming and maintenance relative to peer-to-peer and multi-hop architectures. Hierarchical models are often used in highly scalable systems and in decentralized systems, particularly when multiple entities must interact for the system.

The **Client-Server** architecture is the dominant architecture used by distributed applications over a network.

In the context of two processes communicating with one another, the **client** is the process that initiates the communication. The **server** is the process that waits to be contacted to begin a session.

Network applications utilize the Application Layer and its interfaces and protocols to communicate using the network. The Application Layer is the most flexible layer in the Network Model since it has full access to the host system computing resources. This also limits its network functionality of the Application Layer to the end systems.

An Application Layer process sends messages into and receives messages from the network through **sockets.**

The Application Layer interfaces with the Transport Layer through ports. Processes in the Application Layer register with a particular port, typically through the *socket Application Programming Interface* (*API*). The socket API defines an interface usable at the Application Layer to create an abstraction that a local program can use for network communications. The socket API provides access to the services at the Transport Layer, and the *socket address* is the combination of the IP address of the host and the port number used. Sockets do not need to be bound to a port in order to send communications. However, a socket must be bound to a port in order to receive communications.

A **socket** must be bound to a port in order for a process to receive communications from the network.

Communication between Application Layer processes requires that each process communicates using the same Application Layer protocol. The most widely used Application Layer protocols include the Hypertext Transfer Protocol (HTTP) used to transfer files in the World Wide Web, the File Transfer Protocol (FTP) used for interactive file transfer and the Simple Mail Transfer Protocol (SMTP) used to transfer mail messages.

7.2 *Domain Name System (DNS)*

The Domain Name System (DNS)

DNS was first detailed in RFC 882, authored by Paul Mockapetris, with the details building upon the early broad design outlined in RFC 819, authored by Jon Postel and Zaw-Sing Su. DNS was created as a distributed database solution to enable the easy management of the Internet as it grew, and grew rapidly, during the 1980's.

7.3 *Electronic Mail (email)*

Electronic mail, or simply email, is a method of exchanging messages between users on either the same computer system or on different computer systems. The first email systems gained substantial use in the 1960's, and its popularity led email to become the first killer application for the computer networks.

Email was the first killer application of the Internet.

Modern email works on a *store-and-forward* system. An *email server* accepts email messages from clients and other email servers, forwards email messages to other email servers, delivers email messages to the intended recipients and stores email messages for retrieval by the intended recipients. The use of an email server allows for the sender and the receiver of an email message to operate asynchronously with one another since the email server will store a message at least until it is retrieved by the recipient.

Email systems use SMTP (Simple Mail Transfer Protocol) to transfer email messages between machines.

SMTP was initially defined in 1982 with RFC 821. The modern Extended SMTP protocol is defined in RFC 5321.

SMTP by default uses TCP port 25.

Originally, email allowed for ASCII text messages only. The adoption of the Multipurpose Internet Mail Extensions (MIME) standards allowed for a range of non-ASCII content to be sent via email messages.

MIME is defined in RFC 2045, RFC 2046, RFC 2047, RFC 2049, RFC 4288 and RFC 4289. The integration with SMTP for email messages is defined in RFC 1521 and RFC 1522.

7.4 World Wide Web

World Wide Web uses HTTP (Hypertext Transfer Protocol) as the primary application layer protocol for communications.

A *Uniform Resource Locator* (*URL*) is an identifier from a particular identification scheme. Every URL has the following general structure:

scheme : scheme-specific-identifier

mailto:john@johndoe.com

ftp://ftp.downloadit.com/instructions.pdf

The HTTP scheme is the most widely used in the World Wide Web.

HTTP URLs are of the following form:

http:// servername [:port] [/pathName] [?query] [# fragment]

Items in square brackets are optional

HyperText Markup Language (*HTML*) is used to specify the content for a Web page and the formatting of that content within the browser window. HTML allows for the specification of pointers, or *links*, to other Web pages. These links are referred to as *hypertext*.

7.5 Streaming Audio and Video

7.6 Network Management

Even small networks may have thousands of interacting hardware and software components, all of which must be managed for no other reason than to maintain security and proper system operation.

Network management includes the deployment, integration and coordination of hardware, software and humans to monitor, test, configure, analyze and control the network and its constituent components to meet the performance, cost and application requirements of the network.

major components

Managing entity manages the components of the network.

The *Simple Network Management Protocol* (*SNMP*) is used for communication between the managing entity and *agents* on the managed devices. SNMPv3 is the current version of SNMP, and it contains significant security and administration capability additions beyond previous versions of SNMP.

managed devices contain *managed objects* whose performance and operational data is gathered by the local agent and sent into a Management Information Base (MIB). The MIB is a distributed information store containing the network management data.

Structure of Management Information (SMI) is the data definition language for MIB objects. SMI was designed to provide a well defined and unambiguous syntax and semantics of management to simplify the management and maintenance of the network.

SNMP uses encryption to provide for confidentiality on all of its messages. Authentication is provided through the use of a Message Integrity Code (MIC). A keyed hash (the MIC) is calculated over the message with shared secret key. A nonce is used to protect against replay attacks.

SNMP managing entity maintains a database of access rights and policies for users, and the database itself is accessible as a managed object.

7.7 *Summary*

Index