



CISCO TM

AN INTRODUCTION TO NETWORKS AND
CISCO – BEGINNERS COURSE

DANIEL PLAZA
DANIEL.ROE.PLAZA@HOTMAIL.COM

An Introduction to Networks and Cisco

This page is intentionally left blank.

TABLE OF CONTENTS

CHAPTER 3 ITN – NETWORKING PROTOCOLS AND COMMUNICATION.....	1-6
<i>Communication Fundamentals</i>	<i>1</i>
<i>Protocol Fundamentals</i>	<i>1-2</i>
<i>Network Protocols</i>	<i>2-3</i>
<i>Internet Standard Organizations</i>	<i>4</i>
<i>OSI and TCP/IP Model.....</i>	<i>4-5</i>
<i>Segmentation of Data</i>	<i>5</i>
<i>Protocol Data Units</i>	<i>5-6</i>
<i>IP Packet Header.....</i>	<i>6</i>
<i>Ethernet Frame Header</i>	<i>6</i>
CHAPTER 7 ITN – IP ADDRESSES.....	7-18
<i>IPv4 Address Expression</i>	<i>7</i>
<i>IPv4 IP Proportions.....</i>	<i>8</i>
<i>ANDING</i>	<i>8</i>
<i>Prefix Lengths.....</i>	<i>9</i>
<i>Types of Addresses in Network</i>	<i>9</i>
<i>Unicast, Multicast and Broadcast.....</i>	<i>10</i>
<i>IPv4 Address Blocks (Legacy).....</i>	<i>10-11</i>
<i>IP Address Assignment</i>	<i>11</i>
<i>IPv6 Address Representation.....</i>	<i>12</i>
<i>Omitting Leading 0s.....</i>	<i>12-13</i>
<i>Omitting All 0s Segments</i>	<i>13</i>
<i>IPv6 Address Types</i>	<i>13</i>
<i>IPv6 Link-Local Unicast Addresses</i>	<i>14</i>
<i>IPv6 Global Unicast Addresses</i>	<i>14</i>
<i>Configuring IPv6 on a Router</i>	<i>14-15</i>
<i>Stateless Address Autoconfiguration (SLAAC)</i>	<i>15-16</i>
<i>IPv6 Link-Local Addresses.....</i>	<i>16-17</i>
<i>Assigned IPv6 Multicast Addresses</i>	<i>17</i>
<i>ICMPv4 and ICMPv6</i>	<i>17-18</i>
CHAPTER 8 ITN – SUBNETTING	19-32
<i>Reason for Subnetting</i>	<i>19</i>
<i>Octet Boundaries</i>	<i>19-20</i>
<i>Classless Subnetting with the /24 Prefix.....</i>	<i>21-22</i>
<i>Magic Number Technique</i>	<i>22</i>
<i>Assigning Subnets to a Router</i>	<i>23-24</i>
<i>Classless Subnetting with the /16 Prefix.....</i>	<i>24-25</i>
<i>Classless Subnetting with the /8 Prefix.....</i>	<i>25-26</i>
<i>Subnetting Based on Requirements.....</i>	<i>26-28</i>
<i>Traditional Subnetting Problem</i>	<i>28</i>
<i>Variable Length Subnet Mask (VLSM)</i>	<i>28</i>
<i>VLSM in Practice</i>	<i>29-30</i>
<i>Network Address Planning</i>	<i>30-31</i>
<i>IPv6 Global Unicast and Subnets</i>	<i>31-32</i>

TABLE OF CONTENTS

CHAPTER 1 RSE – INTRODUCTION TO SWITCHED NETWORKS 33-38

<i>Introduction to Switched Networks.....</i>	<i>33</i>
<i>Access, Distribution and Core Layers</i>	<i>34</i>
<i>Switch Form Factors and Requirements</i>	<i>35</i>
<i>General Concept of Switching</i>	<i>36</i>
<i>Switch Forwarding Methods</i>	<i>37</i>
<i>Collision and Broadcast Domains.....</i>	<i>38</i>

CHAPTER 2 RSE – BASIC SWITCHING CONCEPTS AND CONFIGURATION 39-54

<i>Preamble</i>	<i>39</i>
<i>Beginning of Switch Booting.....</i>	<i>39</i>
<i>Switch LEDs and Indicators.....</i>	<i>40</i>
<i>Configuring Basic Switch Management Access with IPv4.....</i>	<i>41-42</i>
<i>Duplex Communication.....</i>	<i>43-44</i>
<i>Verifying Switch Port Configuration and Network Access Issues.....</i>	<i>45-46</i>
<i>Basic Troubleshoot Concept in the Network Access Layer.....</i>	<i>46</i>
<i>Secure Shell (SSH) Operation and Configuration.....</i>	<i>47-48</i>
<i>Common Security Attacks.....</i>	<i>49-50</i>
<i>Network Security Practices.....</i>	<i>51-54</i>
<i>Catch-up</i>	<i>54</i>

CHAPTER 3 RSE – VLANs..... 55-62

<i>Prologue</i>	<i>55</i>
<i>VLAN Terminology</i>	<i>55</i>
<i>Types of VLANs.....</i>	<i>55-56</i>
<i>VLAN Trunks (Brief).....</i>	<i>56</i>
<i>VLAN Tagging.....</i>	<i>56-57</i>
<i>VLAN Implementation.....</i>	<i>57-58</i>
<i>VLAN Trunks (Configuration).....</i>	<i>59</i>
<i>Introduction to Dynamic Trunk Protocol (DTP).....</i>	<i>60</i>
<i>VLAN Troubleshooting</i>	<i>61</i>
<i>VLAN Attacks.....</i>	<i>61-62</i>
<i>VLAN Design Guidelines.....</i>	<i>62</i>

CHAPTER 4 RSE – ROUTING CONCEPTS..... 63-76

<i>Preface.....</i>	<i>63</i>
<i>Network and Router Characteristics</i>	<i>63-64</i>
<i>Packet Forwarding Mechanisms</i>	<i>65</i>
<i>The Assembly of a Network.....</i>	<i>66-67</i>
<i>Configuration of a Router.....</i>	<i>67-69</i>
<i>Expressions and Filters.....</i>	<i>70</i>
<i>Routing Switching Functions</i>	<i>71-72</i>
<i>The Routing Table</i>	<i>73-74</i>
<i>Assignment of Directly Connected Interfaces.....</i>	<i>74</i>
<i>Static Routing.....</i>	<i>75</i>
<i>Dynamic Routing.....</i>	<i>76</i>

TABLE OF CONTENTS

CHAPTER 5 RSE – INTER-VLAN ROUTING	77-84
<i>Foreword</i>	<i>77</i>
<i>What is Inter-VLAN Routing</i>	<i>77</i>
<i>Legacy Inter-VLAN Routing</i>	<i>77</i>
<i>Router-on-a-Stick</i>	<i>78</i>
<i>Multilayer Switch</i>	<i>78</i>
<i>Configuration of Legacy Inter-VLAN Routing</i>	<i>79</i>
<i>Configuration of Router-on-a-Stick Inter-VLAN Routing.....</i>	<i>80</i>
<i>Troubleshooting Inter-VLAN Routing.....</i>	<i>81</i>
<i>Introduction to Layer 3 Switching</i>	<i>82</i>
<i>Routed Ports.....</i>	<i>83</i>
<i>Configuring Static Routes on Catalyst 2960</i>	<i>83-84</i>
CHAPTER 6 RSE – STATIC ROUTING.....	85-96
<i>Prelude</i>	<i>85</i>
<i>Static Routing Basics.....</i>	<i>85-86</i>
<i>IP ROUTE Command</i>	<i>87</i>
<i>Next-Hop Static Route</i>	<i>88</i>
<i>Directly Connected Static Route.....</i>	<i>89</i>
<i>Fully Specified Route</i>	<i>89</i>
<i>Default Static Route.....</i>	<i>90</i>
<i>IPv6 Static Route.....</i>	<i>90</i>
<i>Next-Hop IPv6 Static Route</i>	<i>91</i>
<i>Directly Connected IPv6 Static Route.....</i>	<i>91</i>
<i>Fully Specified IPv6 Route</i>	<i>91</i>
<i>Default IPv6 Static Route.....</i>	<i>92</i>
<i>Classful Network Addressing.....</i>	<i>92-93</i>
<i>Classless Inter-Domain Routing (CIDR).....</i>	<i>93</i>
<i>Fixed-Length Subnet Masking (FLSM).....</i>	<i>94</i>
<i>Variable-Length Subnet Masking (VLSM).....</i>	<i>94</i>
<i>Route Summarization</i>	<i>95</i>
<i>Route Summarization for IPv6</i>	<i>95</i>
<i>Floating Static Routes.....</i>	<i>96</i>
<i>Troubleshooting Static Routes</i>	<i>96</i>
CHAPTER 7 RSE – ROUTING DYNAMICALLY.....	97-108
<i>Introduction</i>	<i>97</i>
<i>Dynamic Routing History.....</i>	<i>97</i>
<i>Static VS Dynamic</i>	<i>98</i>
<i>Dynamic Routing Protocol Operation</i>	<i>98-99</i>
<i>Routing Protocols.....</i>	<i>99</i>
<i>IGP, EGP, and AS.....</i>	<i>100</i>
<i>Distance Vector Routing Protocol.....</i>	<i>100</i>
<i>Link-State Routing Protocols</i>	<i>100-101</i>
<i>Classful and Classless Routing Protocols</i>	<i>101</i>
<i>Distance Vector Technologies (RIP and IGRP)</i>	<i>102</i>
<i>RIP Configuration.....</i>	<i>103-105</i>
<i>Link-State Technologies (OSPF and IS-IS).....</i>	<i>106-107</i>
<i>The Routing Table.....</i>	<i>107-108</i>

TABLE OF CONTENTS

CHAPTER 8 RSE – SINGLE-AREA OSPF	109-119
<i>Introduction to Open Shortest Path First</i>	<i>109-110</i>
<i>Link-State Operation</i>	<i>110</i>
<i>OSPF Messages</i>	<i>111</i>
<i>OSPF Operational States.....</i>	<i>112-113</i>
<i>Introduction to OSPFv2.....</i>	<i>113-115</i>
<i>OSPF Cost</i>	<i>115</i>
<i>Verifying OSPF Neighbors.....</i>	<i>116</i>
<i>OSPFv3 Differences</i>	<i>117</i>
<i>Configuration of OSPFv3 and Troubleshooting</i>	<i>118-119</i>
CHAPTER 9 RSE – ACCESS CONTROL LISTS	120-129
<i>Introduction to Access Control Lists.....</i>	<i>120-121</i>
<i>The Operation of ACLs</i>	<i>121-122</i>
<i>ACL Wildcard Masking.....</i>	<i>122-124</i>
<i>Guidelines for ACL Creation.....</i>	<i>124</i>
<i>Standard and Extended ACL Placement.....</i>	<i>125</i>
<i>Criteria Statements</i>	<i>126-127</i>
<i>Extended IPv4 ACLs.....</i>	<i>128</i>
<i>Troubleshooting ACLs</i>	<i>129</i>
CHAPTER 10 RSE – DHCP	130-135
<i>Introduction to Dynamic Host Configuration Protocol.....</i>	<i>130</i>
<i>DHCPv4 Operation</i>	<i>130-131</i>
<i>DHCPv4 Message Format.....</i>	<i>131-132</i>
<i>DHCPv4 Server and Client Configuration</i>	<i>132-134</i>
<i>Troubleshooting DHCPv4.....</i>	<i>134-135</i>
CHAPTER 11 RSE – NETWORK ADDRESS TRANSLATION	136-144
<i>Introduction to Network Address Translation</i>	<i>136</i>
<i>Network Address Translation Details and Terminology.....</i>	<i>136-137</i>
<i>NAT in Action.....</i>	<i>137</i>
<i>Static NAT and Dynamic NAT</i>	<i>137-138</i>
<i>Port Address Translation (PAT)</i>	<i>138-139</i>
<i>Benefits and Disadvantages of NAT.....</i>	<i>139</i>
<i>Configuration of Static and Dynamic NAT</i>	<i>140-141</i>
<i>Configuration of PAT and Port Forwarding</i>	<i>142-144</i>
<i>Troubleshooting NAT</i>	<i>144</i>

I made this manual for the sake of my own note taking, referencing and help during competitions I may be a part of. This is not intended to be used as a book, but a reference to a couple particular chapters in Cisco Curriculum. Along with that, this book is *not* verbatim to the course material offered to me from <http://netacad.com>, some may be rephrased, but I typed every single thing on here for my own need and for the need of others. **Please don't judge me for any mistakes, some information might be wrong, in that case, just send me what I need to fix.**

An Introduction to Networks and Cisco

All rights reserved to Cisco Systems Inc. These are just my own notes for Cisco for my own understanding.

Communication Fundamentals

The devices must have a common way of communication, for networks can be of any size, shape or form. For instances, humans use the following for communication (1) while computers use the following (2).

1. Message Source > Transmitter > Transmission Medium > Receiver > Message Destination
2. Message Source > Encoder > Transmitter > Transmission Medium > Receiver > Decoder > Message Destination

Protocol Fundamentals

Protocols are used to establish a general ground for communicating, if one is not there, communication will be much harder. The following requirements must be used with a protocol for successful communication:

- Ability to identify sender and receiver
- Common language and grammar
- Speed and timing of delivery
- Confirmation or acknowledgment requirements

OR

- **Messenger Encoding** - Encoding the information into another acceptable form for transmission
- **Message Formatting and Encapsulation** - When a message is sent from source to destination, it must use a specific format or structure. It depends on the type of the message and the channel used to send it. When a message is sent over a computer network in the format of a frame. Below is a frame reference.

Destination (MAC ADDR)	Source (MAC ADDR)	Start Flag (BEGIN)	Recipient Flag (DESTINATION)	Sender Flag (SOURCE)	Encapsulated Data (BITS)	End of Frame (END)
Frame Addressing		Encapsulated Message				

- **Message Size** - The messages are said in fragments or the sentences are limited to size. When a long message is sent from one host to another, it's necessary to break the message into smaller pieces and send the most it can in each packet so that the original message will be created.
- **Message Timing** – Message timing follows a set of rules:
 - **Access Method**
Access method determines when someone is able to send a message, if two people talk at the same time, a collision of information occurs and the people must start again. So. They both need to know when to begin.
 - **Flow Control**
in network communications, source and destination hosts use flow control methods to negotiate correct timing.
 - **Response Timeout**
If a person asks a question and does not hear a response within an acceptable amount of time, the person acts accordingly and may either repeat the question/request or end the conversation. This happens in networking.
- **Message Delivery Options** - There are three different types of delivery options among a network. Unicast, meaning a message is meant for only a single destination. Multicast, meaning it's for more than one host, it can send to multiple hosts simultaneously. Broadcast, meaning it's sent to everyone among the network. Some protocols use a special multicast that is sent to all devices, basically making a broadcast. Hosts may be required to acknowledge the receipt of some messages while not needing to acknowledge others.

An Introduction to Networks and Cisco

When a group of inter-related protocols necessary to perform a communication function is called a *protocol suite*. Protocol suites are implemented by hosts and networking devices in software, hardware or both. The style that it may follow would be the following:

CONTENT LAYER - The message within the frame.

RULES LAYER

1. Use a common language – This means use the same protocol as the source or destination.
2. Wait your turn – This means detect if another device is already transmitting or not, this is called **Carrier Sense**.
3. Signal when finished – This can be compared to the TCP handshake.

PHYSICAL LAYER - How will we physically communicate?

Network Protocols

For devices to successfully communicate, a network protocol suite must be described with precise requirements and interactions. All of these protocols will belong to different categories, such as network, transport, etc. But there are some important things that we need to clear before we can begin to talk about protocols.

How are messages formatted and structured? Saying you're sending the message via IPv4, the router receives and says it can forward it because it can send IPv4, and the server is accepting it because it can understand IPv4.

What is the process by which networking devices share information? For instance, a sharing of similar error codes is one way that they can share information, for instance: If one device/path is down, all the other devices will be notified.

What happens beforehand of setup and termination of data transfer sessions? The client would ask prior to initiating, and when the server agrees both the client and server initiate a connection.

All protocols belong to a different layer; all of these layers when combined together are called a *protocol stack*. One protocol stack that we would use is called TCP/IP. In the Application Layer, we would use HTTP. It is an application protocol that governs the way a web server and web client interact. It relies on the layers below it.

TCP belongs to the transport layer; it is a transport protocol that divides the TCP messages into smaller pieces called segments. They run between the server and client and TCP cannot work without the layers below it.

IP that belongs to the Internet layer is responsible for taking the formatted segments from TCP, encapsulating them into packets and then assigning them to appropriate destinations for delivery. This cannot work without the next layer.

The last layer we're mentioning here is the network access Layer. Ethernet would be considered the protocol here. Ethernet serves the purpose of communication over a datalink and the physical transmission of the data on the network. Essentially it goes from the *bottom to the top for inbound data* and from the *top to bottom for outbound data*.

Layer Name	Relating Protocol Example (Or, the stack)
Application	Hypertext Transfer Protocol (HTTP)
Transport	Transmission Control Protocol (TCP)
Internet	Internet Protocol (IP)
Network Access	Ethernet

An Introduction to Networks and Cisco

It should be noted that almost every industry of protocols will follow this type of process. These layers will use a lot of different protocols; here are the ones that will be used in everyday processes.

APPLICATION LAYER

- **DNS** – Domain Name Service, Translates domain names like cisco.com into IP addresses
- **BOOTP** – Bootstrap Protocol, Enables diskless workstation to discover its own IP address, the IP address of the BOOTP server on the network and a file to be loaded into memory to boot the machine, BOOTP can also be considered a very old version of DNCP.
- **DHCP** – Dynamic Host Configuration protocol, dynamically assigns IP address to client stations at start-up and allows reuse of IP addresses.
- **SMTP** – Simple Mail Transfer Protocol, enables clients to send emails and servers to send emails to other servers
- **POP or POP3** – Post Office Protocol, enables clients to retrieve email from a mail server.
- **IMAP** – Internet Message Access protocol, enables clients to access email stored on a mail server, maintains emails on server.
- **FTP** – File Transfer Protocol, allows users on one host to access and transfer files to and from another host over a network
- **TFTP** – Trivial File Transfer Protocol, a simple connectionless file transfer protocol.
- **HTTP** – Hypertext Transfer Protocol, set of rules for exchanging things around the World Wide Web.

TRANSPORT LAYER

- **UDP** – User Datagram Protocol, enables a process running on one host to send packets to a process running on another host, does not confirm successful transmissions, thus, faster but less reliable.
- **TCP** – Transmission Control Protocol, reliable transmission that confirms successful delivery.

INTERNET LAYER

- **IP** – Internet Protocol, receives message segments from transport layer, packages messages into packets and addresses them for end-to-end delivery over networks, includes IPv4 and IPv6.
- **NAT** – Network Address Translation, used to translate internal and external IP addresses.
- **ICMP** – Internet Control Message Protocol, provides feedback from destination host to source host about errors with packet delivery.
- **OSPF** – Open Shortest Path First, self-explanatory, design based on areas, a cisco routing protocol.
- **EIGRP** – Enhanced interior gateway routing protocol, cisco proprietary routing protocol, uses composite metric based on bandwidth, delay, load and reliability.

NETWORK ACCESS LAYER

- **ARP** – Address Resolution Protocol, used to learn a layer 2 address when a layer 3 address is known.
- **Ethernet** – Defines the rules for wiring and signaling standards of the network access layer.
- **PPP** – Point-to-Point Protocol, establishes direct connection between two nodes.

Internet Standards Organizations

- **Internet Society (ISOC)** - Promotes open development and development of internet use through world
- **Internet architecture board (IAB)** is responsible for the overall standards and development of Internet Standards.
- **Internet Engineering Task Force (IETF)** – Develops, updates and maintains internet and TCP/IP technologies.
- **Internet Research Task Force (IRTF)** – Focuses on long-term research related to the internet and TCP/IP Protocols
- **Internet Corporation for Assigned Names and Numbers (ICANN)** – Coordinates IP address allocation
- **Internet Assigned Numbers Authority (IANA)** – Responsible for overseeing and managing IP address allocation
- **Institute of Electrical and Electronics Engineers (IEEE)** – Organization dedicated to advancing and creating standards in a wide area of technology fields
- **Electronic Industries Alliance (EIA)** – best known for wiring, connectors, etc.
- **Telecommunications Industry Association (TIA)** – Responsible for developing communication standards
- **International Telecommunications Union-Telecommunication Standardization Sector (ITU-U)** – Large organization defining standards for video compression and broadband, very old.

OSI and TCP/IP Model

The reason that we choose to use a layer model can assist in protocol design, competition, preventing technology from affecting each other negatively and providing common grounds for communication. With a model that all people can reference, it helps keep a common ground for everyone. It works similarly to the TCP/IP model, which will be

OSI MODEL		DESCRIPTION OF LAYER	PROTOCOLS	TCP/IP MODEL	
7	Application	Contains protocols used for process-to-process communication	HTTP, DNS, DHCP, FTP	Application	4
6	Presentation	Handles raw data and provides common data representation			
5	Session	Keeps session established with dialogue and data exchange			
4	Transport	Used to segment, transfer and reassemble data between devices	TCP, UDP	Transport	3
3	Network	Provides the ability to exchange data over networks	IPv4, IPv6, ICMP	Internet	2
2	Data Link	Exchanges data frames between devices over common media	PPP, Ethernet	Network Access	1
1	Physical	Anything mechanical, electrical			

I made this model to give you a good idea as to how the OSI model and TCP/IP model share various traits and features. For a more in-depth explanation of each layer, look below.

OSI MODEL

- **Application** – Contains protocols used for process-to-process communication, defines the end-user.
- **Presentation** – Presentation layer provides for common representation of the data transferred between application layer services, defines program and data translation.
- **Session** – Manages and keeps the session established with dialogue and management of data exchange, defines ports and session management.
- **Transport** – Used to segment, transfer and reassemble the data for individual communications between end devices, defines the way data is transferred via TCP or UDP.
- **Network** – Network layer provides services to exchange the individual piece of data over the network between identified end services, defines how packets and data will reach their destination. (*Routers work here*)
- **Data link** – Data link layer protocols describe methods for exchanging data frames between devices over common media. (*Switches work here*)
- **Physical** - Anything mechanical, electrical, etc. Helps with transmission between network devices.

An Introduction to Networks and Cisco

TCP/IP MODEL

- **Application** (OSI 7 TO 5) - Represents data to user, with encoding and dialog control.
- **Transport** (OSI 4) - Supports communication between various devices across diverse network.
- **Internet** (OSI 3) - Determines best path through networks.
- **Network Access** (OSI 2 AND 1) - Controls hardware devices and media making up network.

Segmentation of Data

In theory, single communication could be sent across the network from a source to a destination as one massive, uninterrupted stream of bits. In practice, this would make it so that no other device would be able to do so if your device does it. This would cause a big delay within the network, resulting in terrible performance. This is where the way we now handle data comes in handy Dividing data into smaller pieces, it's called Message Segmentation. Benefits of it include:

- Sending smaller pieces makes it so that many different conversations can be interleaved on the network, this is called multiplexing. Think of it as sending a piece of your data, then the other users sends their small piece, then you send yours and it continues.
- This can increase the efficiency of network communication as if a small piece was missing from the communication, only that missing part will need to be retransmitted.

Protocol Data Unit (PDU)

As data passes down the protocol stack, various information is added along with it. This is called the encapsulation process. The form that a piece of data takes at any layer is called a Protocol Data Unit (PDU), so let's say the Ethernet later has data in it, that frame is called a Protocol Data Unit, or, PDU. During encapsulation, each succeeding layer encapsulates the PDU that it receives from the layer above in accordance with the protocol being used.

For example, the four items are the PDUs that are encapsulated.

- **Data** – User Data
- **Segment** –Transport Header
- **Packet** – Network Header
- **Frame** – Ethernet Frame

What will happen is that the user data within the Data PDU is encapsulated in the Segment Header, those two are inside packet with the network header and everything is encapsulated within the frame.

The network and data link are responsible for ensuring the data is going where it needs to go, for instance:

- **Network layer source and destination addresses** – Responsible for delivering the IP packet from the original source to the final destination
- **Data link layer source and destination addresses** – Responsible for delivering the data link frame from one network interface card (NIC) to another NIC on the same network

IP packets will only contain the two following IP addresses:

- **Source IP address** – The IP address of the original source
- **Destination IP address** – The IP address of the final destination

As for Data Link addresses, the purpose is to deliver data link frames from one network interface to another.

Before IP packets are sent, it must be encapsulated in a data link frame so it can go through a physical medium. As it travels through each physical device, the IP packet gains a new data link frame. Each data link frame contains the source data link address of the NIC card sending the frame, and the destination data link address of the NIC card receiving the frame. Basically meaning that during every hop, the Destination and Source NIC are changed (hop being from one device to another).

To understand how devices communicate, you need to understand that roles of both network layer and data link addresses must work with each other.

(Network Layer) IP Packet Header

An IP address contains two parts: The **network portion** indicates which network the IP address is a member of, *if* the network portion is the same as other devices, then they are on the same network. The **host portion** is the part that uniquely identifies the device on the network.

When IP addresses communicate with each other, there are two addresses within the IP Packet Header: The **Source IP address** is the IP address of the sending device. The **Destination IP address** is the IP address of the receiving device.

The *subnet mask* is used to identify the network portion of an address from the host portion.

When data is being sent to a different network, it will be indicated based on the network portion of the IP address.

(Data Link Layer) Ethernet Frame Header

When the sender and receiver of the IP packet exist on the same network, the data link frame is sent to the receiving device from the sending device. Data link addresses, known as Ethernet addresses (or Media Access Control Addresses, also known as MAC addresses) are physically embedded on the NIC (Network Interface Card). MAC addresses are written in hexadecimal notation. *Data Link layer addresses are used for local delivery.* **The Ethernet sublayer that handles communication between layers 2 and 3 is called LLC, or, Logical Link Control.**

The **Source MAC address** is the data link address of the sending device. The **Destination MAC address** is the receiving device's MAC address.

When the sender and receiver are on a different network, Ethernet data link frames cannot be sent directly to the destination host because they are not directly reachable. As a result, the Ethernet frame must use the MAC address of the router or default gateway. If the default gateway is **not** configured, then you cannot communicate with a remote network.

Some terms to remember:

- **Encoding** – Process of modifying data to an acceptable format for transmission
- **Decoding** – Process of modifying transmitted data into a format that is usable by a device
- **Encapsulation** – Process of inserting a formatted message inside another formatted message
- **De-capsulation** – Process of removing a formatted message from another formatted message
- **Segmentation** – Process of splitting data into smaller pieces for transmission on a network
- **Unicast** – A form of message delivery in which a message is delivered to a single destination
- **Multicast** – A form of transmission in which a message is delivered to a group of hosts
- **Broadcast** – A form of transmission in which a message is delivered to all hosts on a network
- **Open Standard** – Refers to protocols that are available to the public at no cost
- **Proprietary** – Refers to protocols that are developed by companies and are available for purchase
- **PDU** – Protocol Data Unit, the form of a piece of data that is associated with each protocol layer
- **Default Gateway** – A router that is responsible for redirecting the packets to their destination

IPv4 Address Expression

IPv4 Addresses are expressed in binary and decimal. Binary is a numbering system consisting of the numbers 0 and 1 called bits. Binary is understood universally so it is important to understand. Each IP address consists of 32 bits divided into 4 sections, these sections are called octets.

Each octet contains 8 bits (or, 1 byte) separated by periods. Working with binary numbers can be challenging, for ease of use by people, IPv4 addresses are commonly expressed in dotted decimal notation (regular decimal numbers 0-9). Just to show you what it will look like when you translate them, here is a quick example:

209.165.200.225 translates to 11010001.10100101.11001000.11100001
 192.168.10.10 translates to 11000000.10101000.00001010.00001010

At first, you may not understand this, and that's because I haven't shown you one of the first techniques for conversion. Learning to convert to binary to decimal for the first time requires an understanding of positional notation.

Positional notation means that a digit represents different values depending on the "position" the digit occupies in the sequence of numbers. You already know the most common numbering system, the decimal (base 10) notation system.

For this example, we're going to use 11001011.

Positional Value	128	64	32	16	8	4	2	1
Binary Value	1	1	0	0	1	0	1	1
Add	128	64	0	0	8	0	2	1
Results	128 + 64 + 8 + 2 + 1 = 203							

To explain, the *Positional Value* is a potential value can add to your decimal, it does not change.

The *Binary Value* section can change depending on what exactly you'll be putting in there, which will either be 0s or 1s.

The *Add* section will also change, if a one exists in the binary value, then you will add the Positional Value above it.

If a zero exists where the binary value is, then you will not add the positional value above it.

The Positional Value once again, does not change. The Binary Value will depend on what set of numbers you have, as you will place them in the appropriate spaces. The Add section depends on what you have in the Binary Value section.

Ultimately, when you've added the positional values that exist within your octet, you'll have your decimal value.

It is also important to learn the skill of converting decimal back to binary. Doing this is rather simple, I'll explain it in steps.

1. Let's say your number is 192. This number will be subtracted by 128, 64, 32, 16, 8, 4, 2, 1, etc.
2. In that order, you need to consider the following "Is my number greater than or equal to 128?"
3. 192 is greater than 128, as a result, you will subtract 128 from 192 and continue to the next step. But before you go onto the next step, you will put a 1 where 128 would be, so your octet would go from 00000000 to 10000000.
4. Next, you ask yourself, is my new number, 64, greater to or equal than 64? If it is, you will subtract 64 from 64. In doing this, you do so and add a 1 to the second place value.
5. Your new octet is also going to be 11000000.
6. You no longer have any numbers to work with, as a result, you cannot subtract anymore, so the octet will stay 11000000.

Ultimately, your octet at the end of going through the table (if your decimal was 192) will be 11000000.

IPv4 IP Portions

An IP has two segments. As a 32 bit value, there will be a network portion and a host portion. The **Network Portion** will determine what network your host exists on. The **Host Portion** is the number that uniquely identifies your host. It's perfectly fine for your Network Portion to be the same as some others, however, the host portion must be unique.

How do hosts know which portion of the 32-bits is for the network portion and which is the host portion? This is where the subnet mask is important. A computer cannot simply tell what portions of the IP address are what. A subnet mask is used to find the network portion where the device belongs. The most common subnet mask you will find would be the subnet mask 255.255.255.0.

Notice how 3 of the octets are 255, that must mean that the binary value for the first three octets are all 1's. This is true, with every one that exists in the subnet mask; it means that it is a part of the network portion.

ANDING

One of the ways to find out what belongs to the network and what belongs to this host is ANDING.

ANDing is one out of three basic binary operations used in digital logic by computers. The purpose of the process is for the computer to determine the network and host portion. The other two operations are OR and NOT. While all three are used in data networks, only AND is used for determining the network address, meaning AND is the only logical operation.

The process of ANDING compares the Host Address and Subnet Mask to determine what parts belong to the network, in ANDING, here is the way the bits are ANDed.

1 AND 1 = 1
0 AND 1 = 0
0 AND 0 = 0
1 AND 0 = 0

Notice how only 1 AND 1 only produces 1. Once again, to identify the network address of an IPv4 host, the host address is ANDed bit by bit with the subnet mask. *ANDing with the host address and the subnet mask yields the network address.*

Host Address in Decimal	10	188	200	242
Subnet Mask in Decimal	255	255	255	224
Host Address in Binary	00001010	10111100	11001000	11110010
Subnet Mask in Binary	11111111	11111111	11111111	11100000
Network Address in Binary	00001010	10111100	11001000	11100000
Network Address in Decimal	10	188	200	224

The main areas you'll be comparing when it comes to ANDING once again are the Host Address in Binary and the Subnet Mask in Binary. With a host address and subnet mask in decimal, you should be able to convert them back to binary to be able to initiate in an ANDing operation. When comparing once again, you have to be looking for when the Host Address and the Subnet Mask share 1's in the same place, and if they do, then it contributes to the network mask.

Some people have trouble with ANDing (me in the beginning for instance) so I tried to be as thorough as possible with this method.

Prefix Lengths

There is an easier way to define subnet masks, this can be done with Prefix Lengths. If you had the subnet mask 255.255.0.0, the prefix will be 16. Why will it be 16? Simply the subnet mask 255.255.0.0 has 16 bits in the subnet mask. Here's a chart of prefix lengths. You should get the idea of how they work simply by observing the examples.

Subnet Mask	32-bit Address	Prefix Length
255.0.0.0	11111111.00000000.00000000.00000000	/8
255.255.0.0	11111111.11111111.00000000.00000000	/16
255.255.255.0	11111111.11111111.11111111.00000000	/24
255.255.255.128	11111111.11111111.11111111.10000000	/25
255.255.255.192	11111111.11111111.11111111.11000000	/26
255.255.255.224	11111111.11111111.11111111.11100000	/27
255.255.255.240	11111111.11111111.11111111.11110000	/28
255.255.255.248	11111111.11111111.11111111.11111000	/29
255.255.255.252	11111111.11111111.11111111.11111100	/30

Types of Addresses In Networks

The types of addresses reserved on various networks are used for specific purposes. There are some addresses that need to be defined before you can continue. In the next couple examples, I'm going to be using the network 192.168.1.0.

- **Network Address** - All hosts within the network share the same network address. The host portion is all 0s in binary. An example address would be 192.168.1.0.
- **Host Addresses** - Unique IP addresses assigned to hosts. The host portion always contains sorted 0s and 1s, but is never all 0s or all 1s, so it would never be 0 or 255. Example addresses would be the ranges 192.168.1.2 to 192.168.1.3.
- **First Host Address** - The first available host IP address in the network. The host portion in binary is 0s and ends with a 1. The address is usually occupied by the Router/Default Gateway. An example address would be 192.168.1.1.
- **Last Host Address** - The last available host IP address in the network. The host portion always has all 1s and ends with a 0. (254)
- **Broadcast Address** - A special address for communicating with all hosts in a network. The broadcast address uses the highest address in the network range. The host portion is all 1s (255).

How are IP addresses assigned? Either *statically* or *dynamically*. Some devices require fixed IP addresses, such as printers, servers and networking devices need an IP that does **not** change.

Static IP addresses are the ones that are fixed. Dynamic is when the IP address has been received from a DHCP server. DHCP is Dynamic Host Control Protocol, it is used to assign IP addresses. The DHCP server provides an IP address, subnet mask, default gateway and other configuration information. The IP address is leased for a small time, and when the host shuts down or leaves, their IP is removed from them and given to another or stored into the DHCP IP address pool.

Unicast, Multicast and Broadcast

Unicast is the process of sending a packet to an individual host. *IPv4 Unicast host addresses are in the range 0.0.0.0 to 23.255.255.255*, but within this range many addresses are reserved for special purposes.

Multicast is the process of sending packets to multiple hosts. It reduces traffic by allowing a host to send a single packet to a selected set of hosts using the multicast support. IPv4 reserved the 224.0.0.0 to 224.0.0.255 for multicasting on a local network only. Routers will normally recognize this and prevent them from forwarding any further.

Broadcast is the process of sending packets to all hosts. Network protocols such as DHCP, use broadcasts. Broadcasts can be directed or limited, meaning it's sent to all hosts on a specific network or all hosts. Broadcasts should be limited in use as it does use a lot of network resources. Routers will never let them forward any further as well.

Routers connected to the local network know these packets are addressed to a local network multicast group and does not forward them any further than they need to go. If a multicast packet was sent to a host that wasn't supposed to get it, it will be tossed.

IPv4 Address Blocks (Legacy)

The Public IPv4 addresses are addresses globally routed between ISP (Internet Service Providers) routers, however, not all available IPv4 addresses can be used on the internet. Due to the increasing use of IP addresses the following IP address blocks have been reserved.

- 10.0.0.0 /8 **or** 10.0.0.0 to 10.255.255.255
- 172.16.0.0 /12 **or** 172.16.0.0 to 172.31.255.255
- 192.168.0.0 /16 **or** 192.168.0.0 to 192.168.255.255

The reason why is because these are considered internal IP addresses for C, B and A class networks. So understand that if any IP addresses starting with those are sent out, then it will be normally blocked out by the router.

There are some addresses that have been reserved for specific uses.

- **Loopback addresses** (known as local host) (127.0.0.1 /8 or 127.0.0.1 to 127.255.255.254)
This is a special address used by the host to direct traffic to himself/herself.
- **Link-Local addresses** (169.254.0.0. /16 or 169.254.0.1 to 169.254.255.254)
This is known as an automatic private IP addressing (APIPA) addresses, they're used by the Windows DHCP client to self-configure when there is no DHCP server available. Also used for P2P (peer-to-peer) connections.
- **TEST-NET addresses** (192.0.2.0 /24 or 192.0.2.0 to 192.0.2.255) useful for teaching and learning. Can be used in documentation and network examples.

The next thing that I need to mention is legacy classful addressing. **They are also known as public addresses.** This is called classful addressing and it is defined in RFC-790. There are 3 ranges of legacy addresses.

- **Class A** (0.0.0.0/8 to 127.0.0.0 /8) Designed to support the biggest networks with more than 16 million host addresses, it uses the prefix /9 with the first octet to indicate the network address and the remaining three octets for host addresses. Class A networks are reserved for the highest organizations with the possibility of 128 possible A class networks in the world.
- **Class B** (128.0.0.0 /16 – 191.255.0.0 /16) was designed to support the need for larger sized networks. It can hold approximately 65,000 addresses and uses the prefix of /16. There can be over 16,000 of this network type.

- **Class C** (192.0.0.0 /24 – 223.255.255.0 /24) was designed for small networks, normally home networks. They allow a maximum of 254 hosts. This can create over 2 million Class C networks.

There is also a D class network of 224.0.0.0 – 239.0.0.0 and a Class E experimental network with the addresses 240.0.0.0 – 255.0.0.0.

Long ago, the classful system allocated 50% of the addresses to 128 Class A networks. 25% of the addresses were given to Class B and Class C was given the rest to share with Class D and E. This system was considered wasteful, for example, if a network had 260 hosts, the 64,740 addresses would be wasted by giving them a B class network. What we use today is called *classless addressing*. It was a new set of standards to delay the eventual depletion of IPv4 addresses.

IP Address Assignment

Who is responsible for assigning IP addresses? There are organizations for each region, for North America, we have ARIN, the American Registry for American Numbers. There are 4 other organizations for the rest of the regions. Both IPv4 and IPv6 addresses are managed by the Internet Assigned Number Authority (IANA).

These registries can also be referred to as Regional Internet Registries, or, RIRs. There are 5 RIRs, AFRINIC (Africa), APNIC (Asia), ARIN (America), LACNIC (Latin America) and RIPE NNC (Europe).

Now, with the ever increasing use of the internet comes new needs for addresses. With IPv4, it's theoretically possible to have a maximum of 4.3 billion addresses, but as we continue to have more devices get connected, we begin run out of addresses much faster than anticipated. IPv6 is designed to be the successor to IPv4. IPv6 has a large 128-bit address providing 340 undecillion addresses (340,000,000,000,000,000,000,000,000,000,000).

However, with the introduction of IPv6 comes new features and protocols, such as ICMPv6, which will introduce a newer generation of networking complexity and efficiency.

Now, there will not be one day where everyone switches from IPv4 to IPv6. For now, there are three ways to migrate from IPv4 to IPv6.

- **Dual Stack** – Allows IPv4 and IPv6 to coexist on the same network segment. Basically IPv4 and IPv6 run on the same device so they stack simultaneously.
- **Tunneling** – This is the process of transporting an IPv6 packet over an IPv4 network. The IPv6 packet is encapsulated inside an IPv4 packet.
- **Translation** – Network Address Translation 64 (NAT64) allows IPv6 devices to communicate with IPv4 enabled devices using a translation technique similar to NAT for IPv4.

Note: Tunneling and Translation is only where it's needed, it may be best to use Dual Stack.

IPv6 Address Representation

IPv6 addresses are 128-bits in length and written as a string of hexadecimal values. Every 4 bits are represented by a single hexadecimal digit; for a total of 32 hexadecimal values. IPv6 addresses are not case-sensitive and many of the rules that apply to binary will also apply here. To the right I have a chart that will give conversions between decimal, binary and hexadecimal.

At first, hexadecimal may seem a bit intimidating, however, it should be relatively simple. If you remember from positional notation, the last 4 digits in binary for an octet are 8, 4, 2 and 1. Look closely at decimal and binary to understand how they're the same thing.

When it comes to hexadecimal, the most important thing to understand is that it does *not* go beyond 9, from that point, it continues onward to A, B, C, etc...

The thing being, the A, B, C are actually 10, 11 and 12. It continues as D equals 13, E equals 14 and F equals 15.

Within every 4 hexadecimal digits, there are 16 binary digits, for instance, if we had F1A3 as one of my 4 digits in my address, it would look like this in binary...

1111000110100011

4 hexadecimal values create a hextet, just like an octet. 4 hexadecimal digits consist of 16 binary digits.

One certainly notable thing is if we create a 128-bit address, it could get rather lengthy. For example, here is an IPv6 address:

2001:0DB8:0000:1111:0000:0000:0000:0200

How can we shorten this? One method would be Omit Leading 0s; it can make the address above look like this:

2001:DB8:0:1111:0:0:0:200

That's defiantly a lot shorter, how can this be done? I'll explain the first rule.

Decimal	Binary	Hexadecimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

RULE #1 – OMIT LEADING 0s

To explain, this rule applies to leading 0s. To reduce the amount of work needing to be done, you simply remove any 0s that are in the lead of the rest of the numbers. For example:

- 01AB can be represented as 1AB
- 09F0 can be represented as 9F0
- 0A00 can be represented as A00
- 00AB can be represented as AB

The example I gave above was a good one on how to make IPv6 addresses much smaller.

RULE #2 – OMIT ALL 0 SEGMENTS

Another way to reduce the notations is the use of double colons. It can be used to replace any single, continuous string of one or more 16-bit segments of all 0s (tip: By 16 bit segment of 0s, I mean 4 hexadecimal 0000's), for example:

1. The average IPv6 address would be represented as 2001:0DB8:0000:1111:0000:0000:0000:0200.
2. If we were to remove the Leading 0s, it would look like 2001:DB8:0:1111:0:0:0:200.
3. If we were to **compress it by omitting all 0 segments**, it would look like 2001:DB8:0:1111::200

To explain what I did, in #2, I turned 0000:0000:0000 into 0:0:0, in #3, I turned 0:0:0 into ::, I can do this because if there is a continuous string of 4 hexadecimal 0000's, you can shorten it with ::.

IPv6 Address Types

There are three types of addresses for IPv6.

- **Unicast** – An address used to uniquely identify a device with IPv6 enabled
- **Multicast** – An address used to send a single packet to multiple destinations
- **Anycast** – An IP address that can be assigned to multiple addresses and when a packet is sent to an anycast address, the packet will go to the closest one.

In IPv6, there is no broadcast address. There is an IPv6 all-nodes multicast address which is basically the same thing.

When it comes to prefix length, IPv6 uses the prefix length to represent the prefix portion of the address. IPv6 does not use the dotted-decimal subnet mask notation. Since IPv6 has such a large address, the prefix length can range from 0 to 128. Typically, the prefix length will be /64, leaving the rest as a host portion.

IPv6 unicast addresses are also uniquely identifiable. A packet sent to a unicast address is received by the interface that assigned it. Just as in IPv4 where the source must be a unicast address, the source IPv6 address must also be unicast. Destination IPv6 addresses can either be unicast or multicast. Here are the most common types of unicast addresses.

- **Global Unicast** – Similar to IPv4 public addresses, these are globally unique and internet routable. These addresses can be configured statically or assigned dynamically.
- **Link Local** – These addresses are used to communicate with other devices on the same local link. *With IPv6, the term link refers to a subnet.* Their uniqueness must only be confirmed on that link because they are not routable beyond so. *Meaning routers will not forward packets with a link-local source or destination address.*
- **Loopback** - ::1/128, ?
- **Unspecified Addresses** - ::/128, ?
- **Unique Local** – This is another type of unique local unicast address. IPv6 unique local addresses are similar in a way to the private addresses for IPv4, but with some differences that pose as important. Unique local addresses are used for local addressing within a site or between a limit numbers of areas. These addresses are not considered routable in the global IPv6 and should not be used as a global IPv6 address. The range of these addresses are FC00::/7 – FFFF::/7. *Unique local addresses can be used for devices that will never need or have access from another network.*
- **Embedded IPv4** - ?

IPv6 Link-Local Unicast Addresses

Just to clarify, Link-Local enables a device to communicate with others on the same link and only that link (**Note! Link is basically the equivalent of subnets in IPv4**). Link-local addresses are in the FE80::/10 range. /10 indicates **the first 10 bits** are 1111 1110 10XX. The first hextet has a range of 1111 1110 1000 0000 (FE80) to 1111 1110 1011 1111 (FEB8).
Typically, it is the link-local address of a router and not the global unicast address that is used as the default gateway.

IPv6 Global Unicast Addresses

IPv6 global unicast addresses are globally unique and routable. These are basically the equivalent of IPv4 public addresses. The Internet Committee for Assigned Names and Numbers (ICANN) and the operator for IANA allocates IPv6 address blocks to the five Regional Internet Registries (RIRs). At the moment, only global unicast addresses with the first three bits of 0001 or 2000::/3 are being assigned, only 1/8th of all available IPv6 addresses, excluding only a very small amount. (**Note!** The 2001:0DB9::/32 address has been reserved for documentation purposes, such as the examples in this guide).

The global routing prefix is the network portion of the address assigned by the ISP. It's typically assigned with a /48 global routing prefix. The structure of a global routing prefix is as follows:

48 bits	16 bits	64 bits
Global Routing Prefix	Subnet ID	Interface ID
A /48 routing prefix + 16 bit Subnet ID = /64 prefix		64 bits

What this would mean that the first 48 bits is for global routing, the next 16 bits are for the subnet and the last 64 bits are to identify the host. Here are some example addresses: 2001:0DB8:ACAD:0001:0000:0000:0000:0010 /64

From that, you can tell that the entire prefix (64) consists of the global routing prefix (48) and the subnet ID (16 bits), in more detail, 2001:0DB8:ACAD is the global routing prefix. 0001 is the Subnet ID. For practice, it would look like the following if compressed: 2001:DB8:ACAD:1::10 /64. Remember that the prefix tells us what kind of address it is.

You *must* remember that the interface ID is basically the same as the host portion of the IP address. The subnet ID is used by an organization to identify subnets within its site. The larger the subnet ID, the more subnets available.

Configuring IPv6 on a Router

To configure IPv6 on a router, here are the console commands (**Note:** Command examples will be in **consoles**):

R1(config)# interface gigabitethernet *interface_id* - This command is used to enter an interface.

The interface command is used to access the physical ports, it can vary from fastethernet to gigabitethernet, for now, we'll use gigabitethernet. The ***interface_id*** is where you will place the interface, for this. Normally, the ports might be things such as fastethernet 0/0 or gigabitethernet 0/1, but it varies from device to device.

R1(config-if)# ipv6 address *ipv6_address* – This command is executed on an interface.

The IPv6 address command can be used to add a certain IPv6 address to the interface, for now, we're adding an IPv6 interface. For now, put **2001:db8:acad:1::1/64** as the address for this example.

R1(config-if)# no shutdown – This command is executed on an interface to change the state to up for access.

The command “**shutdown**” will turn off the port, or, place it into the down state. The opposite command “**no shutdown**” will turn it on, or, will shift the state back into up.

When it comes to adding an interface, what you’ll be doing is accessing the interface, adding the IP address, changing the state of the interface to up and moving on to the next. There are two ways in which a device can automatically receive a global unicast IPv6 address; these methods are Stateless Address Autoconfiguration (SLAAC) and DHCPv6.

Some notable addresses are 2001:0DB8::/32 as the global unicast address and ::1 is a unicast loopback address.

Stateless Address Autoconfiguration (SLAAC)

Stateless Address Auto configuration (SLAAC) allows a device to obtain prefix, prefix length, default gateway and other information from the IPv6 configured router without a DHCPv6 server. The way it does this is by sending a Router Advertisement (Known as an RA) message every 200 seconds to all IPv6 enabled devices. The RA message is also sent in response to a host sending an ICMPv6 Router Solicitation (RS) message.

By default, IPv6 routing is not enabled on a router, to do so, put **ipv6 unicast-routing** in global configuration. If you do not want ipv6 routing, put **no ipv6 unicast-routing** in global configuration.

The ICMPv6 RA is a suggestion as to how to obtain the IPv6 global unicast address. Ultimately, the operating system will decide the address for itself. The ICMPv6 RA includes:

- **Network prefix and prefix length** – Tells the device which network it belongs to.
- **Default gateway address** – This is an IPv6 link-local address, the source IPv6 address of the RA message
- **DNS addresses and domain name** – Addresses of DNS servers and a domain name

There are three messages the RA message comes with:

- **Option 1:** This means that the device will **only use SLAAC**, remember that SLAAC is stateless, meaning there is no central server to keep everything in check, meaning the client will use the information from the RA to create their own global unicast address. The Prefix is received from the RA and the interface ID is generated from EUI-64, meaning a random 64-bit number is generated.
- **Option 2:** By default, the first message is the only one they use. The router can be configured to send a message that advertises **SLAAC and a stateless DHCPv6 server**. The RA message will suggest the following:
 - SLAAC is used to create its own IPv6 global unicast address.
 - The router’s link-local address will be given as a gateway address.
 - A stateless DHCPv6 server will be provided for DNS server address and a domain name.
- **Option 3:** With this option, only a stateful (dedicated) DHCPv6 server will be used to give the user a global unicast address, the prefix length and addresses of DNS servers. With this option, the RA will suggest and give:
 - The router’s local-link address, being the default gateway address.
 - A stateful DHCPv6 server to obtain global unicast address, DNS server address, domain name and other information that will be useful.

Stateful DHCPv6 servers are dedicated and maintain the list of all devices using IPv6. The default-gateway can only be obtained dynamically from RA messages. Stateless or stateful DHCPv6 servers do not provide the default gateway address.

The client knows the prefix portion and subnet ID of the address, but they must create their own interface ID. Extended Unique Identifier (EUI) or EUI-64 is the process of incorporating a client's MAC address (48-bits) and inserts another 16 bits in the middle of the 48-bit address. Ethernet MAC addresses are usually represented in hexadecimal by two parts:

- **Organizationally Unique Identifier (OUI)** – A 24-bit (6 hexadecimal digits) vendor code assigned by IEEE.
- **Device Identifier** – The device identifier is a unique 24-bit (6 hexadecimal digits) value within a common OUI.

A EUI-64 Interface ID is made up of 3 parts:

- The first 24-bits are the OUI from the client MAC address, but the 7th bit is reversed (0 → 1 or 1 → 0)
- The 16-bit value FFFE is inserted in the middle (1111 1111 1111 1110)
- The 24-bit device identifier from the client is left alone

Randomly generated numbers are also generated if the EUI-64 method is not mentioned. To ensure the uniqueness of any IPv6 unicast address, the client uses a process known as Duplicate Address Detection (DAD). This is similar to an ARP request for its own address, if there is no reply, then ultimately the address created is unique.

IPv6 Link-Local Addresses

All IPv6 devices require an IPv6 link-local address. This can be established dynamically or configured manually as a static link-local address. These can be established dynamically or configured manually.

When an address is created **dynamically (automatically)**, it uses the prefix FE80::/10, so the first 10 bits are occupied by that, the rest is occupied by the interface ID using EUI-64 or a randomly generated 64-bit number. For example:

Let's say that this was your used EUI-64 to get your link-local, your global unicast address would be 2001:DB8:ACAD:1:FC99:47FF:FE75:CEE0 (made with EUI-64), now, because this has to do with global unicast addresses, this means that the first 64-bits are automatically determined for us, which means the other 64-bits are going to be our address. There are two things we could use, an address configured via SLAAC or DHCPv6.

Let's say we had an address configured via EUI-64, because the first-64 bits are FE80::/10 (in binary, that's 1111 1110 1000 0000, 0000 0000 0000 0000, 0000 0000 0000 0000, 0000 0000 0000 0000, remember what I said about compressed addresses?) Anyway, so let's say our we used the address from above, we'd have to remove the first 4 hextets, leaving us with FC99:47FF:FE75:CEE0, if we combine both, we get our link-local IPv6 address: FE80::FC99:47FF:FE75:CEE0.

The same applies if our link-local address was made with a random 64-bit generated interface ID, but regardless, we need to remember one thing, *for link-local addresses, the first four hextets are FE80::/10.*

When an address is created **statically (manually)**, it would follow the same rules as the command **ipv6 address** command, the only difference is that you'll be adding some extra parameters. So you'll type how it is, the IP address, and the parameter link-local at the end, the result will be the addition of a static link-local address, for example, your command should look like the following:

```
ipv6 address fe80::1 link-local
```

Of course, it should follow the fact that the first 64-bits belong to the FE80::, however, after the first 4 hextets, you're free to put what you'd like. The convenience of a static is it is more unique and possibly easier to remember. I believe we did say that if you set an address statically, it might be because the device should have a link-local address that should not change.

You can use the commands **show interface** to view MAC addresses and Ethernet interfaces. If you also do the command **show ipv6 interfaces brief**, you'll be able to view each interface in more detail.

When you do the command **show ipv6 interface brief**, you'll notice that some of the interfaces have two interfaces, remember that the first IPv6 address is the link-local unicast address, the second address is the global unicast address. Also using the command **show ipv6 route** will allow you to view the routing table.

Assigned IPv6 Multicast Addresses

Recall that multicast is used to send a single packet to multiple destinations, IPv6 addresses have the prefix FF00::/8. *Now multicast can only be destination addresses and **not** source addresses.* There are two different types of IPv6 multicast addresses:

- **Assigned Multicast**
- **Solicited Node multicast**

Assigned multicast addresses are reserved addresses for predefined groups of users. *Assigned multicast addresses are used in context with specific protocols such as DHCPv6.* There are two common assigned multicast groups, including:

- FF02::1 All-nodes multicast group – This is a group that all IPv6-enabled devices join. A packet sent to this group is received and processed by all IPv6 interfaces on the link. For example, if IPv6 routers sends an ICMPv6 RA message to FF02::1, it'll get to everyone, including all the information that normally comes with it.
- FF02::2 All-routers multicast group – This is a group that all IPv6 routers join. They become a part of this group when it's enabled on the router. A packet sent to this group is received and processed by all IPv6 routers on the link or network. **Another use is solicited-Node IPv6 multicast addresses. This is when a multicast address is mapped for a specific group.**

ICMPv4 and ICMPv6

Special messages are sent via a certain protocol, this is ICMP. Internet Control Message Protocol, is used to provide feedback about issues with IP. Most of the time, it is disabled for security reasons.

ICMP is enabled for both IPv4 and IPv6, allowing basic functionality, such as:

- **Host confirmation** - ICMP messages can check whether a host is currently active or not.
- **Destination or Service Unreachable** – ICMP can give certain error messages to determine why you cannot connect to that certain host, there are some message codes, such as:
 - 0 – Network unreachable
 - 1 – Host unreachable
 - 2 – Protocol unreachable
 - 3 – Port unreachable
- **Time exceeded** – When a packet takes too long to determine if a host is there or not, it will timeout and send another packet.
- **Route redirection** – This means you most likely took a different route

ICMP can also be used to find errors within the network, especially ICMPv6. Some features for routers and devices include Router Solicitations (RS), Router Advertisements (RA). IPv6 devices exclusively have neighbor solicitation and neighbor advertisements, those are used for things such as address resolution and duplicate address detection (DAD).

Address resolution is when a device on the LAN knows a unicast address, but does not know the MAC address, thus, they will ask for the MAC Address. Duplicate address detection (DAD) will be performed to ensure that no duplicate addresses exist on the link, this is done by sending a message out with the address you currently have, if you get no response, your address is unique.

Great ways to test your network are pinging and tracerouting. Here are some important terms to remember:

- **Binary** - Numbering system consisting of digits 0 and 1, used by computers to perform operations
- **Bit** - A single binary digit, represented by a 0 or a 1
- **Octet** - One section of an IP address, consisting of a group of 8 bits
- **Positional notation** - Determining a digital value based on the digit position, for system-to-system conversion
- **Subnet mask** – Continuous sequence of 1 bits, followed by sequence of 0 bits, used to determine the network portion of an IP address
- **ANDing** – Binary process used to determine the network address of a host with the host address and subnet mask
- **Prefix length** – Number of bits set to 1 in a subnet mask
- **Network address** - A dotted-decimal number that represents a unique IP network
- **Host address** - Any IP address in an IP network that can be assigned to an interface or host
- **Broadcast address** - A dotted-decimal number that represents all hosts in an IP network
- **Unicast address** – The IP address of a single host on a network
- **Multicast address** - An IP address representing a select group of hosts
- **Public address** - An IP address that can be routed on the internet
- **Private address** - An IP address used internally and not routed on the internet
- **Classful addressing** - Specific ranges of IP addresses that make up classes of addresses in which the number of available network and host addresses are defined
- **Classless addressing** - IP addressing standards created in 1993 that allow address allocation based on prefix length rather than predefined class ranges
- **Hexadecimal** - A base-sixteen numbering system using numerals 0 through 9 and letters A through F to represent binary numbers in a more condensed form
- **IPv6 link-local address** - An IP address that is required for every IPv6-enabled network interface and that allows a device to communicate with other IPv6-enabled devices on the same link
- **ICMP/Ping** - A utility that sends a series of echo requests from one IP host to another IP host
- **Traceroute** - A utility that generates details about the route traveled by data sent between two devices

Reasons for Subnetting

To be able to moderate networks, you need to be able to devise a plan as to how exactly you will administrate it. This means that organization and planning for an infrastructure is a big key of success. When IPv4 originally came out, there were two levels of hierarchy: a network and a host. These allowed basic network groupings that ensured that everything went smoothly; however, as soon as networks began to grow and expand, it added complexity and disorganization.

This is where subnetting is needed. Adding subdividing, or, subnetting, added a third level to network hierarchy, making it three levels: Networks, subnetworks and hosts. With these sub-groups added to networks, it created faster packet delivery and filtration, helping to ensure the network is running at top condition.

In an Ethernet LAN, devices use broadcasts to locate things such as:

- Other devices – A device will use Address Resolution Protocol (ARP) which sends a layer 2 broadcast to a known IPv4 address on the network to find the MAC address.
- Services – A host will typically acquire its IP via Dynamic Host Configuration Protocol (DHCP) which sends broadcasts on the server to locate a DHCP server.

Routers do not forward broadcasts outward, therefore, a broadcast domain is exclusive only to *that* network. The problem is having a large broadcast domain will result in excessive broadcasts and will ultimately reduce network performance and the use of host resources.

The solution to any network problems is the creation of subnets, smaller network divisions. The reason we initiate subnetting is to reduce network traffic, improve network performance and implement new security features.

The way we can subnet can depend on various reasons, for example:

- Location, such as floors on a building or sections
- Organizing organizations or people (ex: students and teachers work on different subnets)
- Device type (printers and people work on different subnets)

Other reasons to divide people can make sense, regardless, there are many reasons why subnetting can benefit networks.

Octet Boundaries

Every interface on a router is connected to the network, IP addresses and subnet masks are configured on the router interface are used to identify the specific broadcast domain. Remember that prefix length and the subnet mask are different ways of identifying the network portion of an address. IPv4 subnets are created by using one or more of the host bits as network bits. A majority of the time, they use a whole octet as a part of the subnet.

Subnet Mask	Subnet Mask in Binary n and 1 = network portion, h and 0 = host portion	# of Hosts Available
255.0.0.0 (/8)	nnnnnnnn.hhhhhhhh.hhhhhhhh.hhhhhhhh 11111111.00000000.00000000.00000000	16,777,214
255.255.0.0 (/16)	nnnnnnnn.nnnnnnnn.hhhhhhhh.hhhhhhhh 11111111.11111111.00000000.00000000	65,534
255.255.255.0 (/24)	nnnnnnnn.nnnnnnnn.nnnnnnnn.hhhhhhhh 11111111.11111111.11111111.00000000	254

Notice how the more 1's in the subnet mask, the less users. The 1's represent the network portion of the subnet mask while the 0's represent the host portion.

An Introduction to Networks and Cisco

To understand how subnetting on the octet boundary can be useful, you should look at particular examples.

Let's say we have a subnet mask with the prefix /16. Along with that, we have the IP address 10.0.0.0, the first octet belong to the network portion of the IP address. The rest belongs to the host portion. How many possible subnets do we have? Assuming that we organize the subnet address how we're supposed to with the prefix of /16, we should be able to gain 256 possible subnets.

If we were to try identifying the possible amount of subnets, here is how it would look like if we decided to list just a few of them:

10.0.0.0/16
10.1.0.0/16
10.2.0.0/16
10.3.0.0/16
...
10.255.0.0/16

If we were to attempt identifying the possible amount of hosts per subnet, it would look like the following:

10.0.0.0 – 10.0.255.254
10.1.0.0 – 10.1.255.254
10.2.0.0 – 10.2.255.254
10.3.0.0 – 10.3.255.254
...
10.255.0.0 – 10.255.255.254

If we wanted to identify the broadcast address for each of these subnets, it would look like the following:

10.0.255.255
10.1.255.255
10.2.255.255
10.3.255.255
...
10.255.255.255

With the first column, you get 256 potential subnets and the network addresses. With the second column, you get 65,534 users per subnets. The final column displays what the broadcast addresses would be for each network, *the broadcast address is the last address*. If you subnet the network differently, let's say, using a network with the /24 prefix, you'll be able much more than what you had.

If we were to try identifying the possible amount of subnets, here is how it would look like with the prefix of /24:

10.0.0.0/24
10.0.1.0/24
10.0.2.0/24
...
10.0.255.0/24
10.1.0.0/24
10.1.1.0/24
...
10.255.0.0/24
...
10.255.255.0/24

If you were to attempt to identify the possible amount of hosts per subnet, it would look like the following:

10.0.0.1/24 - 10.0.0.254/24
10.0.1.1/24 - 10.0.1.254/24
10.0.2.1/24 - 10.0.2.254/24
...
10.0.255.1/24 - 10.0.255.254/24
10.1.0.1/24 - 10.1.0.254/24
10.1.1.1/24 - 10.1.1.254/24
...
10.255.0.1/24 - 10.255.0.254/24
...
10.255.255.1/24 -
10.255.255.254/24

If you were to attempt identifying the broadcast addresses for each of these, it would look like the following:

10.0.0.255/24
10.0.1.255/24
10.0.2.255/24
...
10.0.255.255/24
10.1.0.255/24
10.1.1.255/24
...
10.255.0.255/24
...
10.255.255.255/24

With the first column, you will get 65,536 possible subnets. With the second column, you will get 254 possible hosts per subnet, and in the third column, you get all of the broadcast addresses. Remember that the network portion in these examples is underlined. Also remember that the reason you don't get 254 possible hosts is due to the fact that the last address is occupied by the broadcast address.

Classless Subnetting with the /24 Prefix

This is a particularly important subject as it does change the way we use octet boundaries for subnetting. For example, we normally went with a subnet using up octets, but with classless subnetting, we borrow parts of the host portion. Subnets can borrow from any host bit position to create other masks.

For example, the /24 network address is usually subnetted using longer prefixes by borrowing from the fourth octet, or, the host portion. This gives an administrator the ability to assign network addresses to smaller number of end devices:

Prefix Length	Subnet Mask	Subnet Mask in Binary	# of subnets	# of hosts
/25	255.255.255.128	nnnnnnnn.nnnnnnnn.nnnnnnnn.nhhhhhhh 11111111.11111111.11111111.10000000	2	126
/26	255.255.255.192	nnnnnnnn.nnnnnnnn.nnnnnnnn.nnnhhhhh 11111111.11111111.11111111.11000000	4	62
/27	255.255.255.224	nnnnnnnn.nnnnnnnn.nnnnnnnn.nnnnhhhh 11111111.11111111.11111111.11100000	8	30
/28	255.255.255.240	nnnnnnnn.nnnnnnnn.nnnnnnnn.nnnnhhhh 11111111.11111111.11111111.11110000	16	14
/29	255.255.255.248	nnnnnnnn.nnnnnnnn.nnnnnnnn.nnnnnhhh 11111111.11111111.11111111.11111000	32	6
/30	255.255.255.252	nnnnnnnn.nnnnnnnn.nnnnnnnn.nnnnnnhh 11111111.11111111.11111111.11111100	64	2

This is just a chart for these, but, for each bit that we borrow, we double our available subnets while cutting our number of hosts in half. If you don't understand it, perhaps I did not explain it well enough. I'll be doing some scratch work to try and explain it.

1. The first step would be understanding that borrowing a bit will give you an additional subnet while cutting the amount of hosts in half. If we take one bit from, let's say, 192.168.1.0 /24, we would now have 192.168.1.0/25.
2. So we know that we have divided our network, but how can we actually see it? Let's say for example that an address such as 192.168.1.64 /25, if we want to see what part of the network it exists on, we would use an ANDing operation.

Let me explain how the process goes. In the first row we have the IP address that we had selected, 192.168.1.64 /25.

The second row has our subnet mask, that mask is using the prefix /25, and you can tell that it is the prefix /25 by the fourth row.

192	168	1	64
255	255	255	128
11000000	10101000	00000001	10001010
11111111	11111111	11111111	10000000
11000000	10101000	00000001	01000100
192	168	1	0

In the third row we have the address that we had chosen, 192.168.1.64 in binary. If we do an ANDing operation, we can figure out what the network portion of this address is, to do this, we have to check the host address (third row) and the subnet mask (fourth row). If you want to check again how to do the ANDing process, look at page 8. Now, if we do that, in the fifth row, we get our results. What is the network address? 192.168.1.0. *Remember that this is **with** the prefix /25.*

3. We're going to do the same operation, but this time, we'll be doing this with the address 192.168.1.138 /25.

In the first row, we have the host address in decimal again, this time, we loaded in our new address. In the second row, we have the subnet mask of /25 in decimal.

192	168	1	138
255	255	255	128
11000000	10101000	00000001	10001010
11111111	11111111	11111111	10000000
11000000	10101000	00000001	10000000
192	168	1	128

In the third row, we have the host address in binary, and in the fourth row we have our subnet mask in binary. If we do the ANDing method again, we get a completely different item, our *network*

address is actually 192.168.1.128! That is the second subnet. Hopefully this in-depth explanation has helped you.

4. Ultimately, we've created two different subnets with the methods we learned. We created the 192.168.1.0 /25 subnet, which can hold 126 potential hosts, going from 192.168.1.0 to 192.168.1.127. We also have the 192.168.1.128 /28 subnet, going from 192.168.1.128 to 192.168.1.255.

Magic Number Technique

The Magic Number Technique is a technique that many networkers like to use to be able to determine how much the networks will go up in increments? This is determined by the *last 1* in the subnet mask. For this example, we're going to use 192.168.1./26.

Now, the part that matters the most here is the fourth row, the subnet mask in binary. By checking the last binary value, we can tell that it is in the place of 64. In our last example, if we used the magic number technique, we could realize that our magic number was 128.

192	168	1	0
255	255	255	192
11000000	10101000	00000001	10001010
11111111	11111111	11111111	11000000
11000000	10101000	00000001	00000000
192	168	1	0

Right now, we know that our magic number is 64, so our subnet masks will be going up in increments of 64. Each subnet will have 62 usable hosts (**Note! – Remember that we usually subtract 2 due to the network and broadcast address, we have 62 potential hosts**). Anyway, so our network goes up in increments of 64, so these are the ranges from our prefix.

192.168.1.0 /26, 192.168.1.64 /26, 192.168.1.128 /26 and 192.168.1.192 /26.

Assigning Subnets to a Router

Let's say I gave you a topology that consisted of a router. This router has two switches connected to it. Each switch has a multitude of hosts connected. Now, what I'm going to need is two different subnets.

For this topology, let's say we need 2 equal subnets, the /25 prefix works perfectly for this. Now, assuming we remember how to assign IP addresses to interfaces, we'd have to issue the following commands to the router:

```
R1(config)# interface gigabitethernet 0/0
R1(config-if)# ip address 192.168.1.1 255.255.255.128
R1(config-if)# exit
R1(config)# interface gigabitethernet 0/1
R1(config-if)# ip address 192.168.1.129 255.255.255.128
```

Notice how we used the command `ip address`; we used 255.255.255.128 for the subnet mask because that matches the prefix that we're using for our interfaces. It is important to note that the router interfaces must be assigned an IP address within the valid host range for the assigned subnet. The IP address that is assigned will be the address that hosts on that network will use for their default gateway.

Normally, it's common to have the first or last available address in a network range on the router interface. The hosts on the subnet must be assigned the correct subnet mask to be able to join the network.

Now that we know what the interfaces for the routers are, we should assume that the switches should have the first IP address available on the network. So, the first switch (connected to gigabitethernet 0/0) will have the IP address 192.168.1.0 /25. The second switch (connected to gigabitethernet 0/1) will have the IP address 192.168.1.128 /25.

So let's assume we needed 3 subnets. This is another thing about topology and what-not.

We have 2 switches with 2 networks, a router in the middle and a router that is connected to our router. We need 3 networks but taking one bit from the host portion will only give us 2 subnets. To satisfy the need of subnets, we'll need to take 2 bits from the host portion. Even if we had one subnet we weren't going to use, this is completely fine. What you need to remember is *as long as we satisfied the amount of subnets required, we'll be fine*.

Before we can continue, there are some formulas that you can use to figure out the little details, such as, how many hosts am I going to have or how many subnets will I get? The first formula is the **subnet formula**. This formula determines how many subnets you will have. The formula goes as follows: 2^n . The n equals how many bits we borrow from the host portion. Assuming we borrow 2 bits from the host portion, the formula should be 2^2 , as a result, we get 4.

So how many hosts will we have? We can figure this out with the **host formula**. This can be done as shown: $2^h - 2$. For this formula, h equals how many bits belong to the host portion still. This means that for every host bit that is **NOT** taken, we can apply that to h . If we took 2 bits from the host portion, then we should be doing the formula as follows: $2^6 - 2$, what is the result from that? 62, 62 potential hosts.

Now, we'll need to assign IP addresses to all 3 router interfaces, 192.168.1.1 /26, 192.168.1.65 /26 and 192.168.1.129 /26. Since the prefix is now /26, the subnet mask will be 255.255.255.192.

Assigning the interfaces via command-line would look like this:

```
R1(config)# interface gigabitethernet 0/0
R1(config-if)# ip address 192.168.1.1 255.255.255.192
R1(config-if)# exit
R1(config)# interface gigabitethernet 0/1
R1(config-if)# ip address 192.168.1.65 255.255.255.192
R1(config-if)# exit
R1(config)# interface serial 0/0/0
R1(config-if)# ip address 192.168.1.129 255.255.255.192
```

Everything should be correct from that point. Let's talk about smaller prefixes.

Classless Subnetting with the /16 Prefix

If you ever need more subnets or more hosts, you could always use the /16 prefix. Of course that means with the /16 prefix you get one subnet and 65,534 hosts. When it comes to using prefixes and formulas with prefixes' under 24, it means you'll have more stuff to work with, but that does not mean that you should be intimidated by it.

Prefix Length	Subnet Mask	Subnet Mask in Binary	# of subnets	# of hosts
/17	255.255.128.0	nnnnnnnn.nnnnnnnn.nnnnnnnn.hhhhhhhh 11111111.11111111.10000000.00000000	2	32766
/18	255.255.192.0	nnnnnnnn.nnnnnnnn.nnnnnnnn.hhhhhhhh 11111111.11111111.11000000.00000000	4	16382
/19	255.255.224.0	nnnnnnnn.nnnnnnnn.nnnnnnnn.hhhhhhhh 11111111.11111111.11100000.00000000	8	8190
/20	255.255.240.0	nnnnnnnn.nnnnnnnn.nnnnnnnn.hhhhhhhh 11111111.11111111.11110000.00000000	16	4094
/21	255.255.248.0	nnnnnnnn.nnnnnnnn.nnnnnnnn.hhhhhhhh 11111111.11111111.11111000.00000000	32	2046
/22	255.255.252.0	nnnnnnnn.nnnnnnnn.nnnnnnnn.hhhhhhhh 11111111.11111111.11111100.00000000	64	1024
/23	255.255.254.0	nnnnnnnn.nnnnnnnn.nnnnnnnn.hhhhhhhh 11111111.11111111.11111110.00000000	128	510
/24	255.255.255.0	nnnnnnnn.nnnnnnnn.nnnnnnnn.hhhhhhhh 11111111.11111111.11111111.00000000	256	254
/25	255.255.255.128	nnnnnnnn.nnnnnnnn.nnnnnnnn.nnnnnnnn 11111111.11111111.11111111.10000000	512	126
/26	255.255.255.192	nnnnnnnn.nnnnnnnn.nnnnnnnn.nnnnnnnn 11111111.11111111.11111111.11000000	1024	62
/27	255.255.255.224	nnnnnnnn.nnnnnnnn.nnnnnnnn.nnnnnnnn 11111111.11111111.11111111.11100000	2048	30
/28	255.255.255.240	nnnnnnnn.nnnnnnnn.nnnnnnnn.nnnnnnnn 11111111.11111111.11111111.11110000	4096	14
/29	255.255.255.248	nnnnnnnn.nnnnnnnn.nnnnnnnn.nnnnnnnn 11111111.11111111.11111111.11111000	8192	6
/30	255.255.255.252	nnnnnnnn.nnnnnnnn.nnnnnnnn.nnnnnnnn 11111111.11111111.11111111.11111100	16384	2

An Introduction to Networks and Cisco

For this example, we're going to be creating a 100 equal-sized subnet enterprise network with 172.16.0.0 /16.

To subnet the network correctly, we need to borrow bits from the host portion. To get the amount of subnets we need, we need to borrow an amount that will satisfy the requirement of 100 equally-sized subnets. If you remember our formula, 2^n , we can determine how many we need, we can also use the chart on page 24. In this case, we need 100 equally-sized subnets, we can get that if we borrow 7 bits from the third octet, or we do 2^7 .

From that formula, we get 128 subnets, which is enough to satisfy the need for 100 equally-sized subnets. This is going to be our new subnet mask: 11111111.11111111.11111110.00000000 or 255.255.254.0.

Remember our magic number that the last 1 in the subnet mask will determine the increments of how our network will go up, in this case, our network will be going up in increments of 2.

The first network will be 172.16.0.0 /23, the next will be 172.16.2.0 /23, if we continue from this point, the last subnet we will have will be 172.16.254.0 /23. That gives us 128 subnets.

For our host portion, we have nine 0s in our subnet mask. This is where we bring the formula $2^h - 2$. With this, we're going to do $2^9 - 2$. This gives us 510 users per subnet. At first, it may seem confusing considering the fact that we can only have a maximum of 254 users per subnet, but the thing we need to consider is that we have nine 0s in our subnet mask, which means the host segment has 9 values, so here is how it's going to go.

I know I said that the subnets would be like 172.16.0.0 /23 and 172.16.2.0 /23 and so on and so forth, but to ensure that you get all 510 users per subnet, this is how it'll go:

1. The first part of the subnet is 172.16.0.0 /23 right? If we had all the users for the first part of this subnet filled, it would look like 172.16.0.254 /23.
2. The second part of the subnet (for the other half of 510 users) would start with 172.16.1.0 /23, and would continue to fill until 172.16.1.254 /23.
3. The next subnet would fill normally, like 172.16.2.0 /23. If you define the subnet mask, the computer will know what is going on. The only reason this might be confusing at first is due to the fact that you get much more users than you can fit into one octet, we didn't have that problem before.

I'm glad we got the confusion for that one done, it makes sense hopefully. If it does, then in that case, we're going to learn how to subnet a /8 prefix.

Classless Subnetting with the /8 Prefix

If you do subnetting, there may ultimately come a time when you need to subnet at least a thousand subnets while they have plenty of space for hosts. The network address 10.0.0.0 has a subnet mask of 255.0.0.0, or, /8.

In order to create these subnets (because without any subnets, you have 16,777,216 users, think of the broadcast domains you're dealing with here), you're going to need to borrow from the host portion. Remember our formula for creating subnets, 2^n , with this (which still applies to our situation) we can figure out how many bits we will allocate to the network portion. After some experimentation, you should figure out that 10 host bits is sufficient for our need.

2^{10} will give us 1024 potential subnets. The subnet mask is 255.255.192.0, the prefix is /18. How many hosts will we have per subnet? With the formula $2^h - n$, we can find out that we will have 16,382 users per subnet; we found this out by getting the amount of bits left for the hosts, putting it into our formula, and calculating the number.

To find out the host address range, we first need to find out magic number. To find out magic number, we will need to view the third octet. The reason why I look at the first octet is because it is partially 1s and then 0s. We find that the last 1 in the third octet is on the 64 place, as a result, the subnets will be going up in increments of 64.

For our network with 1000 subnets, we will now determine the host address range:

The network address will be 10.0.0.0 /18. Our first host address for the first subnet will be 10.0.0.1 /18. The last host address on our network will be 10.0.63.254 /18. The broadcast address for our network will be 10.0.63.255 /18.

The next subnet will start similarly, with 10.0.64.0 /18 being the network address. The first host address for that network will be 10.0.64.1 /18. The last host address will be 10.0.127.254 /18. The broadcast address will be 10.0.127.255 /18.

The next subnet after this one will start with the network address 10.0.128.0 /18 and so forth. This cycle will go on for about a thousand more subnets; we do not need to go in-depth into it really.

Subnetting Based on Requirements

There are things that need to be taken into consideration whenever you begin subnetting a network. For example, the number of host addresses required and the number of individual subnets. No matter what you do whenever subnetting, there is an inverse relationship going on between the subnets and hosts. The more bits you borrow, the less hosts you have, but the more subnets you receive in return.

At times, subnetting may be required without a specific need when it comes to hosts, such as hosts per subnet. This is due to the fact that an organization can choose to separate network traffic based on their infrastructure. For instance, an organization may want all of their technical support on one subnet, while the administrators are on another subnet. In this case, knowing how many subnets you need is important.



Note: Remember the following formulas for determining hosts and subnets:

- **Subnets:** 2^n : Every bit that takes away from the host portion (n) will decide how many subnets you have.
- **Hosts:** $2^h - 2$: Every bit that belongs to the host portion (h) will decide the amount of hosts per subnet minus two.

You have to remember that when you're borrowing from the host portion, for every bit that you borrow, your number of subnets increases while the amount of hosts decreases. The big thing here is that the amount of subnets you have will double every time. For example, let's say that you need 300 equally sized subnets for 20,000 hosts each. The address you will be working with will be 10.0.0.0 /8. From this, you'll have to borrow from each bit, and for every bit, it doubles.

The first bit will provide you with 2 subnets, the second will provide 4, and the third will provide 8 and so on... Eventually, you will receive 512 bits when you've borrowed 9 bits (formula: 2^9), you will gain 512 usable subnets, which satisfies the requirement of 300 subnets. What about hosts? The formula would go $2^{15} - 2$, which gives us 32,766 potential hosts.

So what are the magic numbers? Let's examine the subnet mask. The subnet mask that we're currently going with:

11111111.11111111.10000000.00000000, which is 255.255.128.0.

The thing about this mask is how exactly will the IP addresses be going up? The magic number will actually be going up in 1s and 128s. To explain, the first magic number is in the second octet, 1, the second magic number is in the third octet, 128. Remember the original mask, /8. This can be shown better through an example. The first subnet would begin with

An Introduction to Networks and Cisco

10.0.0.0 /17, the next one would begin with 10.0.128.0 /17. The next would be 10.1.0.0 /17, going on to 10.1.128.0 /17. This continues on for a while until eventually you get to 10.255.128.0, thus giving you the 512 usable subnet addresses.

There are also things that come into play, such as requirements from actual people. For instance, there is an office that has the IP address of 172.16.0.0 /22 (borrowed 6 bits from the host portion, 10 host bits remaining). This will give us 1,022 host addresses and 64 subnets. Let's say that we have a topology, **some of things that we have to consider when building a network based off of a topology**, are things such as LAN segments and internetwork connections, which are connections between routers.

Let's say we have 5 LAN segments and 4 internetwork connections, thus, 9 subnets are needed. Let's say the largest host needs 40 host addresses available. Because 172.16.0.0. /22 has 10 host bits, and the largest subnet requires 40 hosts. The bare minimum of host bits required would be 6 bits to be able to provide 62 hosts per subnet. But where do we get these bits while retaining the amount of hosts? We can take bits away from the host bit to create a subnet, the minimum amount of host bits required would be 4 for 16 subnets. This would change our prefix from /22 to /26.

With this new information, we can set up our network, the only problem is the fact that our network is kind of confusing right now. For instance, how are we going to subnet it? The answer is simple, since we have borrowed more from the host portion, increasing our subnet mask, we have the 4 bit that we took dedicated to our network, this is how it looks.

#	Network Portion	Borrowed Host Bits For Subnets	Host Bits	Dotted Decimal
ORIGINAL	10101100.00010000.00000000	00.00	00000000	172.16.0.0 /22
0	10101100.00010000.00000000	00.00	00000000	172.16.0.0 /26
1	10101100.00010000.00000000	00.01	00000000	172.16.0.64 /26
2	10101100.00010000.00000000	00.10	00000000	172.16.0.128 /26
3	10101100.00010000.00000000	00.11	00000000	172.16.0.192 /26
4	10101100.00010000.00000000	01.00	00000000	172.16.1.0 /26
5	10101100.00010000.00000000	01.01	00000000	172.16.1.64 /26
6	10101100.00010000.00000000	01.10	00000000	172.16.1.128 /26
7	10101100.00010000.00000000	01.11	00000000	172.16.1.192 /26
8	10101100.00010000.00000000	10.00	00000000	172.16.2.0 /26
9	10101100.00010000.00000000	10.01	00000000	172.16.2.64 /26
... Skipping to the last ranges ...				
14	10101100.00010000.00000000	11.10	00000000	172.16.3.128 /26
15	10101100.00010000.00000000	11.11	00000000	172.16.3.192 /26

This may confuse you at first, but allow me to explain. The network portion stays unchanged and is needed as it is normally needed. The first address in the ORIGINAL row is what we started out with, and as we continue, we see that the prefix has changed from /22 to /26. The reason for this is because we borrowed 4 additional bits from the address that we were originally going to use. Two bits from the third octet and two bits from the fourth octet. They retain the same values as if they belonged to the same octet still; I only separated them to show you what can happen.

Now, we are using those 4 bits for our subnet, we can tell how our subnets will be going up in with the last 1 value in our mask, which is on 64, as a result, the networks are going up in increments of 64. This may seem confusing at first, but it is not. As we go along in changing out addresses from 0 to 64 to 128 to 192, we eventually hit our limit on that and start over on another segment, being that the third octet goes from 0 to 1. It continues and the third octet goes to 2.

Our network would function like any other now that we've established our masks and address ranges. If we are given a certain network address (once again, 172.16.0.0 /22) and it has a prefix that is not /8, /16 or /24, we can treat it like any other address. This takes practice.

Traditional Subnetting Problem

With traditional subnetting, every subnet has fixed address blocks. For example, let's say that there is a topology that requires seven subnets. With traditional subnetting, if we had the address 192.168.20.0 /24, we would simply have to allocate 3 bits to the subnet and meet our goal, this will leave 5 host bits and 30 usable hosts per subnet. This meets the criteria of what you had requested. While this meets the needs of the largest LAN (28 hosts), it wastes many addresses. I didn't mention this at first, but three of the subnets are required for three WAN connections between routers. We don't need 30 potential host addresses for that, thus wasting many addresses (84 addresses since the WAN connections only require 2 addresses).

This also limits the growth of an infrastructure. This is where using Variable Length Subnet Mask (VLSM) is useful.

Variable Length Subnet Mask (VLSM)

Previously we were learning that the subnet mask is what every network shares, and every network is required to use the exact same subnet mask, have the exact same amount of hosts, etc. Well, what if I told you that you could essentially subnet subnets? A traditional subnet would allocate everyone a chunk of all of the addresses, what VLSM allows is the division of network space into unequal parts. With VLSM, the subnet mask will vary depending on how many bits have been borrowed from a particular subnet, thus bringing the whole "variable" part of VLSM.



Note: Always use VLSM with the biggest network first!

To better understand it, let's establish a simple network. The network address is going to be 192.168.20.0 /24. So we have a /24 address here, for our example, we're going to allocate three bits of the host portion towards the subnet, increasing our prefix to 192.168.20.0 /27. We need to establish that our subnet is going up in increments of 32. Let's assume that we need to allocate 4 subnets to 4 LANs on a network, these 4 LANs occupy the ranges .0 to .31, .32 to .63, .64 to .95 and .96 to .127. This means we have a couple more unused networks, such as the .224 to .255 network. Let's say we're going to divide the last subnet, 192.168.20.224 /27.

The reason that we're subnetting this is because we're using our example from Traditional Subnetting Wastes Addresses. We talked about how we wasted 84 host addresses due to the fact that we had allocated 3 subnets with 30 hosts each to the WAN connections that had only required 2 addresses per subnet.

What we're going to do is get the last subnet, 192.168.20.224 /27, and we're going to borrow more bits from it. Approximately 3 more, giving us the /30 prefix (highest prefix), from here, we can show that we now have the following networks created for us:

192.168.168.20.224 /30, 192.168.20.228 /30, 192.168.20.232 /30, 192.168.20.236 /30, 192.168.20.240 /30, 192.168.20.244 /30, 192.168.20.248 /30, 192.168.20.252 /30. Based on the last bit on our prefix, we can tell that our magic number is going to be eight, meaning we get eight subnets.

The whole point of the VLSM subnetting scheme reduces the number of addresses per subnet to the point where you can get both accurate and efficient subnets.

VLSM in Practice

In this next example, we're going to look at VLSM in practice. We're going to be using the address blocks that we had just configured. To get a good idea as to what the network looks like, I'm going to create a visual example.

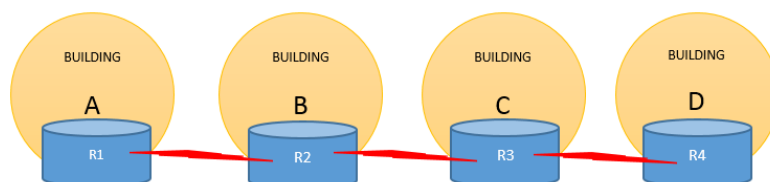
Here's how it's going to work

Building A has the network 192.168.20.0 /27.

Building B has the network 192.168.20.32 /27.

Building C has the network 192.168.20.64 /27.

And finally Building D has the network 192.168.20.96 /27.



So all of these building have their networks, this means that we figured out and have the first four subnets figured out. However, what about the rest of the subnets for the WAN connections (The red links between the routers)? The subnets we have configured (because of VLSM) are the following:

The first serial connection (R1 to R2) has 192.168.168.20.224 /30.

The second serial connection (R2 to R3) has 192.168.20.228 /30.

The third serial connection (R3 to R4) has 192.168.20.232 /30. The last other subnets are not in use and reserved for a later purpose, wasting less host addresses.

Note: If I refer to interfaces as serial, gigabit Ethernet or fast Ethernet, the reason I do so is because interfaces can vary from devices. To view interfaces, do the **show interface** command in global configuration.

Those red connections are essentially the wire between each router, each router needs to have an IP address connected through that interface, for instance, R1 (router 1) is only connected to building A and the connection to R2, so R1 needs to have an interface established for building A's network and the serial connection to R2, how can we configure this? Through the console interface on R1, we can configure the first thing we require:

```
R1(config)#interface gigabitethernet 0/0
R1(config-if)#ip address 192.168.20.1 255.255.255.224
R1(config-if)#exit
R1(config)#interface serial 0/0/0
R1(config-if)#ip address 192.168.20.225 255.255.255.252
```

Let me explain what happened right there. The first interface that we configured was the gigabit ethernet 0/0, which is directly connected to building A. Remember that the first address is the **network address** and the last address is the **broadcast address**, so we don't want to use those. The default gateway for a network (or, the IP address of the router's interface) is going to be usually the first address on the network, if you refer to the address block for building A, the first address is going to be 192.168.20.1, the subnet mask will be based on the prefix.

In the second part with the serial interface, this is for the direct connection to R2; as a result, we had to configure our special address block specifically for it. Remember how the first and last address are reserved for network and broadcast,

so we only have .225 and .226 for the subnet of 192.168.20.224, as a result, the IP address for serial interface 0/0/0 is going to be 192.168.20.225 and the subnet mask (based on prefix) is going to be 255.255.255.252.

I'm going to configure R2 and end it from there, since you should be able to understand it from that point. For R2, the configuration is just going to be a tad different:

```
R2(config)#interface gigabitethernet 0/0
R2(config-if)#ip address 192.168.20.33 255.255.255.224
R2(config-if)#exit
R2(config)#interface serial 0/0/0
R2(config-if)#ip address 192.168.20.226 255.255.255.252
R2(config-if)#exit
R2(config)#interface serial 0/0/1
R2(config-if)#ip address 192.168.20.229 255.255.255.252
```

If you followed how the configuration of R1 worked, you should be able to finish this one easily and quickly. The only difference here is that I configured an additional interface; that interface being the one linking R2 to R3. Notice how once I configured the interface linking R2 to R3; we ended up being on a different subnet. That was the whole point of using VLSM to change our blocks.

Network Address Planning

The thing about designing a network is planning. Address assignment should not be random. The planning of networks require the question, what does the organization do with the network? You need to consider the following:

- What will the organization do with the network?
- How will subnets be structured?
- How many subnets will exist?
- What is their geographical location?
- How many hosts per subnets?
- What devices are being used?
- Will we require the use of VLSM?
- Which hosts will require static addresses and which will require the use assistance of DHCP?

Remember the fact that the reason we subnet is to group devices to improve management and security and to decrease the size of broadcast domains. There are other reasons, but these are just two benefits I like to keep in mind.

Remember that private IP address ranges on the LAN need to be configured correctly and need to be able to change for future expansion or in the event of immediate changes. Remember that there are 3 private IP address ranges you can use:

- 10.0.0.0 – 10.255.255.255 with the subnet mask 255.0.0.0 or /8
- 172.16.0.0 – 172.31.255.255 with a subnet mask of 255.240.0.0 or /12
- 192.168.0.0 – 192.168.255.255 with a subnet mask of 255.255.0.0 or /16

Knowing IP requirements will assist in making decisions for the design of your infrastructure; ensure that you are subnetting the IP range correctly. The public IP address used to connect to the internet are typically allocated the same

An Introduction to Networks and Cisco

way we would, so while the same principles apply to them, it's not something we need to worry about, just remember that when it comes to setting up your network, various things need to be considered.

When planning your network, there are three things that require the most consideration.

- **Preventing Duplication of Addresses** – You need to ensure that every user has a completely unique address, without this, this could result in conflicting IP addresses, which results in issues in the network
- **Providing and Controlling Access** – This is to ensure that some hosts provide resources to internal and external hosts, for example, a server will likely need a static IP address instead of a random, dynamic one.
- **Monitoring Security and Performance** – The efficiency of the network must be kept to a standard and network traffic must be optimized completely, there must also be thorough documentation of your infrastructure and security must be kept to the highest standards to protect your network from intrusion or other threats. Along with that should your network be monitored, if done correctly, anything wrong with the network should be easy to find.

Within a network, there are different types of devices that require IP addresses; this may include devices such as:

- **End user clients** – Most networks usually use Dynamic Host Configuration Protocol (DHCP) to allocate IP addresses from a pool of IP addresses to those who need it. This makes the configuration of all devices much easier. This pool of IP addresses that DHCP provides is usually reassigned to the people that come and go, for example, if someone were to disconnect from the network, their IP address would return to the pool and be used when needed again.
- **Servers and peripherals** – These should have predictable, static IP addresses. This is so that people on the network can easily connect to the network.
- **Servers that are accessible from the internet** – In many cases, servers can be made to where they can be accessed over the internet via remote access (for example, the use of telnet or SSH). These servers must be configured with the firewall and router to allow the translation of internal addresses into public addresses
- **Intermediary devices** – These devices are assigned for network management, monitoring and security. These devices should have predictable, statically assigned addresses.
- **Gateway** – This is probably the most important out of all of the IP addresses. Routers and firewall devices will usually have the first IP or last IP address on the network to make it the most predictable. This addresses is important due to the fact that it the only way to get out of the network.

When organizing and developing your network, you need to plan every detail to ensure that you have a strong, efficient and well-documented network.

IPv6 Global Unicast and Subnets

IPv6 has a much more different approach towards subnetting compared to IPv4. The main reason why IPv6 is so much different when it comes to subnetting is due to the fact that there are so many addresses. IPv6 is not only good about limiting broadcast domains, but managing address scarcity as well. We used things such as subnet masks and VLSM with IPv4 due to the scarcity of the addresses, IPv6 is not concerned with that due to the amount of addresses IPv6 can hold.

Much rather, the subnet ID (which is usually a 16-bit value when it is a global unicast address) is used to identify the network much rather than subnetting it. Remember that link-local IPv6 addresses are **never** subnetted due to it existing only on a local link. IPv6 global unicast addresses can be subnetted however. Remember that IPv6 global unicast

addresses usually consist of the global routing prefix (48-bit value), a subnet ID (16-bit value) and the interface ID (64-bit value).

You can still subnet the global link address however; in all honesty, I do not think currently it would be possible to use all of the subnets you can get with IPv6. The subnet mask can go from 0000 to FFFF, which provides 65,536 subnets, giving you 18 quintillion host IPv6 addresses per subnet.

When it comes to the allocation of IPv6, the fact that you have 65,000 subnets you can implement, along with the fact that you have 18 quintillion host IPv6 addresses per subnet, you do not need to worry about wasting IP addresses. Just remember the fact that you just go up by one hexadecimal every time.

To finish off this chapter, we are going to keep in mind a couple terms and ideas:

- **Subnetting** – The process of subdividing networks into smaller groupings of devices, or subnets.
- **Broadcast Domain** – An area within which a broadcast transmission will be shared.
- **Subnet** – A subdivision of a network that is created either to conserve addresses or to support specific network requirements.
- **VLSM** – Stands for Variable Length Subnet Mask, allows subnets of different sizes to be created from the same network address.
- **Network, Subnetwork, Host** – The three levels of the IP addressing hierarchy.
- **Two effects of excess broadcast traffic** – Slow networks due to excess traffic AND slow devices due to the need to process all broadcasts.
- **Classful Prefix-Lengths** - /8, /16, and /24.
- **2ⁿ Formula** – Calculates the number of subnets created for ⁿ bits borrowed.
- **2^h – 2 Formula** – Calculates the number of hosts per subnet for ⁿ bits remaining in the host field.
- **IPv6 unicast addresses** – 48-bit global routing prefix, 16-bit subnet ID, 64-bit interface ID
- **Gateway** – Router or firewall device, mostly likely router, a device that allows communication of hosts from one network to another network.

Personal Note

We'll be switching from Introduction to Networks (ITN) to Routing and Switching Essentials (RSE) in the next section, from there, the things you learned from ITN 3, 7 and 8 you will be required to thoroughly understand, be prepared to learn a lot more and to use the information you used here. The big thing about this guide is that I wanted to above and beyond what CyberPatriot wanted me to study, so after I'm done with Routing and Switching Essentials and some of CCNA security, I'll be jumping into other cisco material for my own being in an expansion pack of this book.

Introduction to Switched Networks

As networks have evolved over the years, many protocols became highly efficient, mature and developed along with various technologies. Devices must be able to work on the same platform to assure a secure, fast and stable network. This is where LAN switches come into play. With routers, switches and other technologies, you can achieve an infrastructure with wonderful traits. This chapter goes over traffic examination, how networks are created, how LAN switches are built and the use of MAC addresses to get data to where it needs to be.

Switches are to provide quality service, the transfer of data and security. As the digital world changes rapidly, the technologies are changing as well. These days, next generation of networks require the ability to adapt to new protocols and integrate one network with another. *Converged networks are also very affordable.* To support the ability to share so much network resources, networks are converged with the following items, making them a *converged network*:

- **Call Control** – Telephone call processing, the use of Cisco Phones, VoIP, etc.
- **Voice Messaging** – Voice mail and voice calls.
- **Mobility** – The ability to access network resources immediately.
- **Automated Attendant** – Increase speed of voice services by routing directly to individuals or departments.

To be able to provide a network with all of these features, a network must be properly designed and have thorough documentation of all of the details for it. With the increase of converged networks, networks must now be developed with an idea that the network you are designing is an infrastructure; you must build it from the perspective of an architect, requiring intelligent systems, simple operations with the ability to be scalable to meet demands later on.

This is where a Cisco Borderless Network comes into place. The Cisco Borderless Network is a network that combines innovation and design that allows organizations to connect anyone, anywhere, anytime on any device with security, reliability and seamlessness. By combining hardware infrastructure with policy-based software solutions, you receive two primary sets of services: network services and user and endpoint services, all managed by integrated management.

Creating a borderless switched network requires the principles of network design to ensure maximum availability, flexibility, security and management. The borderless switched network must deliver on current requirements and future required services and technologies. **Borderless switched networks keep the following principles in check:**

- **Hierarchical** – Helps for every device on every tier to employ a specific role.
- **Modularity** – Allows network expansion and integration of other devices and applications.
- **Resiliency** – Satisfies user expectations for keeping the network uptime consistent.
- **Flexibility** – Allows intelligent traffic load sharing by using all network resources efficiently.

Understanding that all of these principles are absolutely required together to keep the network completely efficient is one of the most important concepts. To truly design a borderless switched network in a hierarchical form, you must ensure that **security, mobility and unified communication features exist**. The infrastructure must be designed to accommodate to three very important layers, the **access layer, distribution layer, and core layer**.

Access, Distribution, and Core Layers

The **Access Layer** represents the network's edge, where network traffic will go in or out. The primary function is to provide network access to the end-users. As an example, the access layer switches connect to the distribution layer switches, which implement network technologies such as routing, resiliency, flexibility and security. To meet network application and end-user demand, switching must provide a more converged, integrated platform. *Remember that a converged network is a well-designed, infrastructure capable of providing various features without lacking things such as security or efficiency.* **Simply**, providing direct, switched network connectivity to the user *and* helps applications to operate on a switched network more safely and securely.

The **Distribution Layer** is between the access layer and core layer, providing a variety of functions, such as:

- The clustering of large-scale wiring networks
- The clustering of broadcast domains (layer 2) and routing boundaries (layer 3)
- Providing intelligent switching, routing and network access policy
- Providing of high resiliency and efficiency through repeated and well-designed distribution of layer switches to the end users and core
- Providing of various services to various classes of applications to the Access layer

This layer ultimately deals with being in-between of the Access and Core layer, utilizing network technologies in a clustered formation to provide many services and advances using highly efficient network equipment. **Simply**, it is the backbone and users to provide intelligent switching, routing, security, the flow of data on equal-cost switching paths. Also including redundancy and the support of layer 2 broadcast domains and later 3 routing boundaries.

And finally, the **Core Layer** is the backbone of the network and campus network. It is used to connect areas together, for example, a school campus would require various buildings to have different networks, the point of the Core Layer (which can also be considered a management type of layer) is to connect everything together and provide high-speed backbone connectivity and all of the other principles we have discussed. **Simply**, it is also the backbone area for switching, providing fault isolation and high-speed backbone switch connectivity.

In some cases, we cannot scale the network to such a thing, as a result, we will create a two-tier campus network, which consists of the Access Layer, (which still serves the same function) and the Collapsed Distribution/Core Network (which serves the purpose of both layers) it can be referred to as a collapsed design as well.

The role of switched networks has vastly changed with the evolution of technologies available today. For example, originally, the basic layer 2 network had basic Ethernet and hubs, a network like that would result in inefficiency and a lack of security. Eventually, networks changed to adapt to a switched LAN. A switched LAN allows more flexibility, traffic management, and more features, including quality service, increased security, wireless support and new technologies.

Switch Form Factor and Requirements

In the world of networking, you can use various types of switches to administrate your network, when selecting switching equipment for your network, there are some things that need to be considered before purchasing any equipment:

- **Cost** – The cost of the switch will determine how well the switch will perform to your needs, for example, interface speed, features, modularity and how many interfaces you get.
- **Port Density** – How many interfaces you will have for use, this will depend on how many hosts you require.
- **Power** – It is now a common feature for switches to power access points, IP phones, and other items using *Power over Ethernet* (PoE). Meaning, the switch and Ethernet will deliver power to devices.
- **Reliability** – The switch should always have a high up-time, constantly providing a stable network connection.
- **Port Speed** – This is how quickly the interfaces will be able to transfer data.
- **Frame Buffers** – To be able to store frames can be an important feature.
- **Scalability** – The network should be ready for expansion and growth in hosts and other categories.

When selecting a type of switch, the network designer must choose between three different switches. From the three switches, they must choose a thickness of the switch, as switches are mounted on a rack. Here are the types of switches:

Fixed Configuration Switches

Fixed configuration switches are basic switches that do not have any more features or customization options besides factory settings. For instance, it cannot support additional ports or the configuration of extra modules, normally, a fixed configuration switch will come with some additional options, however, they are not built to support expansion.

Modular Configuration Switches

Modular configuration switches are more flexible with their configuration. They normally come in various different shapes and sizes that allow the installation of a variety of modular line cards. These line cards actually contain the ports for the switch; it fits into the switch chassis. The larger the chassis, the more it can support. For example, a switch with 24-ports can have a line card that allows the installation of 24 additional ports.

Stackable Configuration Switches

Stackable configuration switches are interconnected with a cable that specializes in providing high-bandwidth to each of the switches. The technology used for this, Cisco StackWise, allows the interconnection of up to nine switches. Switches can just be stacked upon each other and connected in a daisy chain fashion (look up stackable configuration switches for reference as to how it looks). Stacked switches essentially operate as one, giant switch. *These types of switches are **desirable** where fault tolerance and bandwidth availability are critical.*

General Concept of Switching

The general idea of switching and forwarding frames is universal in networking and telecommunications. Within every network, switching is used, be it LAN, WAN, PSTN, the concept refers to devices making decisions based on two ideas:

The first idea, which is how a switch forwards the traffic, is based on ingress port. Ingress means where the frame entered the device on the port. The term egress is the frame leaving the device from a particular port. When a switch makes decisions, it's based on the ingress port and the destination address of the frame.

LAN switches keep a table to determine how to forward that traffic out of the switch. This table gives destination addresses and port numbers, for example, if port 1 (AA) has to send something to port 3 (CC), it will send in the most efficient way possible. The only intelligence a LAN switch has is its ability to use its table to forward traffic based on the way the frame entered the device *and* the destination address of the message. There is only one master switching table that describes each port and destination address, so ultimately, *a message and a given destination always exit the same egress port, regardless of the ingress port it enters.*



Note: Cisco LAN switches forward Ethernet frames based on the destination MAC address of the frames!

So we know that switches will forward frames based on MAC addresses, switches keep a table of addresses and the ports that they are assigned to. A switch is made up of various circuits and software that will allow the data to be transmitted where it needs to go. As a switch learns of the hosts that populate it, it will build a table called a MAC address table, also known as a Content Addressable Memory (CAM) table. I need to remind you that CAM memory is very fast, as a result, data transmission in switches is extremely fast and efficient.

So we know that switches determine where to send data based on a MAC address table, if it does not have any ways of building the table, it may begin to build the table dynamically, and this process can be described in this example:

1. In the example, PC1 sends a frame to S1 (Switch 1).
2. The switch is going to check the source MAC address and checks the MAC address table, if the address is not in the table, it will assign port 1 with the MAC address of PC1. If the MAC address already has an entry, it will reset a timer for the MAC address on the table, as MAC addresses on the table are kept for five minutes usually.
3. Once that information has been confirmed, the switch will examine the destination MAC address of the frame, *if* the destination MAC address is in the table, then the frame will be forwarded to the correct destination. If the destination address is *not* in the MAC table, the switch will flood all ports with the frame except for the ingress port. The flooding will also happen if it is a MAC broadcast address (All F's, because MAC addresses are in hexadecimal).
4. The destination device (PC3 let's say) will reply with a unicast frame addressed to PC1.
5. Since the switch did not originally know the MAC address, it will assign that MAC address of PC3 with the egress port that frame came from (port 3), thus adding it into the MAC table.
6. After this, PC1 and PC3 can now effectively communicate.

Once again, this is the manual building of a MAC address table, in networks with multiple interconnected switches; the MAC address table will have multiple MAC addresses for a single port when connected to other switches.



Note: *When a frame is flooded to all ports, it is flooded only to those with devices connected to interfaces. All F's on a MAC address indicates a broadcast address. The frame flooded is also not sent to itself, switches will not flood like hubs.*

Switch Forwarding Methods

When networks were expanding back then, there was an experience of the network slowing down, the first version of switches (called Ethernet bridges) were added reduce the size of collision domains. In the 1990s, advancements were made to the current technologies of Ethernet bridges to allow LAN switches to exist. Some of the advances made include integrated circuit technologies, LAN switches were able to move Layer 2 forwarding from software to application-specific-integrated circuits, known as ASICs. ASICs reduced the amount of time a device would handle packets, allowing an increase in port density without negatively effecting systems. This method is referred to as store-and-forward.

Store-and-Forward Switching

The entire process of store-and-forward begins when a switch receives the entire frame, and computes the CRC (CRC is cyclic redundancy check; it is used to check the frame for errors using mathematical error-checking). If the CRC comes out good, the switch looks up the destination address and sends the frame to the correct ingress port.

There are two characteristics that make it different from cut-through, which is the ability to check for errors and the ability to initiate automatic buffering. A switch using store-and-forward checks for errors for incoming frames, when the frame is received, the switch will compare the frame-check-sequence (known as FCS, it is located at the end of the frame) value against its own FCS calculations. The FCS is an error-checking process that ensures that the frame is free of any errors from Layer 2 or 3, if the frame is good to go, the frame is forwarded towards the ingress port, if not, it's dropped.

Another feature is called automatic buffering; it is used to support the mix of Ethernet speeds. For example, an incoming frame going at 100 Mb/s being sent out of a 1 Gb/s interface will not be sent unless store-and-forward is allowed. The frame is placed in a buffer; it goes through a FCS check and is sent out. Store-and-forward conserves bandwidth.

Cut-Through Switching

There is another method that is different from store-and-forward called cut-through. Cut-through forwards the frame before it receives the full frame. It can only do this if the destination address is read first, once it reads the destination address and determines the egress port, it will forward the frame to the ingress port. There are two characteristics that make cut-through more efficient than store-and-forward; rapid frame forwarding and fragment free switching.

Remember how I said that with cut-through you can send frames almost immediately? The only thing that a cut-through switch will require is the destination MAC address of the frame in its MAC address table. It will not require any more things. This can allow the forwarding of the frame within the first 14 bytes of an Ethernet frame (which is the source MAC address, the destination MAC address, and the EtherType fields). Cut-through switching does not drop invalid frames, as a result, potential frames with errors can be forwarded, and this can result in a high error rate caused by invalid frames within a network. Cut-through switching has the potential to negatively impact the network by clogging up bandwidth with invalid frames due to the lack of error checking via FCS and CRC.

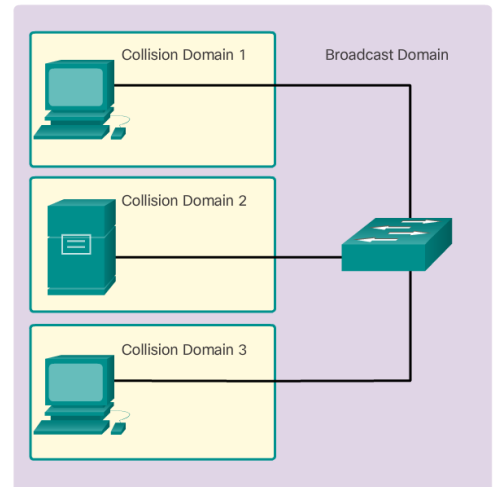
Fragment free is also another feature with cut-through switching. This is a modified form of cut-through switching where the switch waits for the collision window (the 64 bits of the Ethernet frame) to pass before forwarding the frame. This allows each frame to be checked to ensure no fragmentation has occurred. This mode provides better error-checking than cut-through without any increase in latency. The lower latency of cut-through switching makes it more appropriate in an environment where applications require extremely low latencies. The destination NIC will discard incomplete frames if this method is used.

Collision and Broadcast Domains

Let's talk about collision domains again. In hub-based Ethernet segments, network devices compete for who goes first due to the fact that when data is transmitted, everyone takes turns.

Networks that share the same bandwidth between devices are known as collision domains, it gets its name from the fact that when two or more devices communicate, collision can occur. It is possible to use a switch to reduce the number of collision domains along with allowing bandwidth to be available to all devices as each collision domain will belong to each device within a broadcast domain.

To get a better picture, here is an actual picture (I got this picture from <http://netacad.com> by the way). Anyway, a switch can divide a network into segments, reducing the number of devices that compete for bandwidth. When a switch is used, each port represents a new segment within a new collision domain. With switches, it can allow one collision domain to not interfere with another. This can also be referred to as micro segmentation.



While switches filter things based on MAC addresses, they don't filter broadcast frames. For a different switch to receive a broadcast frame, a switch needs to flood that from one interface to the interface of another switch, then, that broadcast will be flooded to that network. A collection of interconnected switches forms a single broadcast domain. The only thing that can break up these broadcast domains is a router, and we know that routers can segment both collision and broadcast domains. Network bandwidth is used to propagate broadcast traffic, as a result, too many can result in network congestion, or, a network slowdown. Sometimes, it can get so bad to the point where it results in office riots.

To prevent network congestion from getting so bad, LAN switches will allow the segmentation of LANs into separate collision domains (micro segmentation). Each port will be a separate collision domain, they also provide full-duplex communication. Switches interconnected LAN segments (collisions domains) will use MAC address tables to make the sending of data much more efficient. There are many things that switches take advantage of to help with network congestion, some of these items include:

- **High port density** – Switches can hold 24 to 48-ports usually with just one rack unit (1.75 inches) and operate at various speeds.
- **Large frame buffers** – The ability to store more received frames before having to start dropping them can help when ports become particularly busy or congested.
- **Port speed** – Some switches these days can allow a variety of speeds, from 100 Mb/s to even 100 Gb/s.
- **Fast internal switching** – Having faster processing speeds can effect overall performance of the switches.
- **Low per-cost port** – Switches can provide a high-port density for lower cost. As a result, LAN switches can work with fewer users per segment, resulting in an increase for bandwidth per use.

When it comes to broadcast domains, remember that the only thing that can separate broadcast domains are routers. When a switch is separating collision domains, each link to each device becomes a collision domain. If a hub is connecting the network, it is the entire collision domain.

Preamble

So in the last section, we talked all about the Cisco Borderless Network architecture and how we can set up a network with the traditional three-layer hierarchical design. That design would split the network into three layers, the core, distribution, and the access layer. It would allow the network to be optimized to and efficient. We also talked about various types of switches and general switching concepts, which included mentions of MAC address tables, how switches work, the various switching methods and collision and broadcast domains.

In this section, we're going to learn about the configuration of Cisco switches (via Cisco IOS), adjustment of various options, remote and local management, security, and essentially the configuration of a LAN switch.

Beginning of Switch Booting

The moment a Cisco switch is turned on, it goes through a certain boot sequence. The beginning of it is loading the power-on-self-test (POST) program that's stored in the read-only memory (ROM). POST checks the central processing unit (CPU), dynamic random-access memory (DRAM) and the flash file system. Next, the switch loads the bootloader software, the bootloader software is a small program stored in ROM and it immediately runs after POST finishes up.

The boot loader will then perform minimal CPU initialization, which is when the CPU registers, which controls where physical memory is, the amount of memory allocate, and the speed. Almost to the end, the boot loader starts up the flash file system that comes with the switch, and the boot loader finally locates and loads up the Cisco Internetwork Operating System (also known as Cisco IOS), thus handing control of the switch from boot loader to the Cisco IOS.

The boot loader looks for the Cisco IOS on the switch by checking the information in the BOOT environment variable. If the variable is not set, the switch will load the first file it can by doing a search through the file system. The file is usually contained in a directory that has the same name as the image file (without the .bin file extension). The IOS operating system will then initialize the interface using Cisco IOS commands; it then begins the configuration file that was found in NVRAM.



Note: Non-Volatile Random Access Memory is memory that can be stored without power. This is normally where configuration files and other items are placed. Volatile memory will be lose when power is absent.

In the case that the boot loader cannot load the system due to the operating system being missing or corrupt, the boot loader has a command-line interface that provides access to files stored in flash, to access them, do the following:

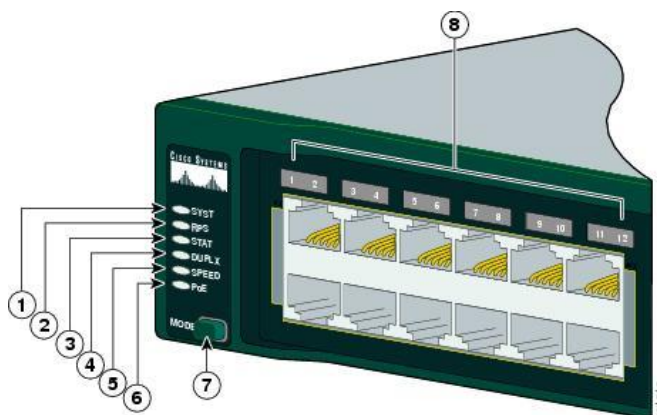
1. Connect a PC via console cable to the switch console port and use terminal software to connect.
2. Unplug the switch power cord.
3. Reconnect the power cord to the switch and press and hold the mode button within 15 seconds while LEDs flash.
4. Continue pressing the Mode button until the system LED turns amber and then solid green, then release it.
5. The boot loader switch prompt will appear in the terminal emulation, thus providing access to flash memory.

There are various commands that can be executed to format the file system, reinstall the operating system, and figure out any password problems.

Switch LEDs and Indicators


For this example, we're going to be looking at the Cisco Catalyst 2960 Switch LEDs. We'll be identifying each of the LEDs and indicator lights that come with them.

1. **System LED, SYST** – Shows whether the system is receiving power and is functioning. If it is off, obviously it's not getting power. If it's green, the system is working fine, if the LED is amber, then it's getting power, but *not* working right.
2. **Redundant Power System LED, RPS** – Shows the status of RPS if RPS is supported by the switch. RPS is a backup power for a switch. If it is green, the RPS is ready to provide back-up power. If the LED is blinking green, the RPS is currently in use by another device. If the LED is amber, it is in standby or fault. If the LED is blinking amber, the power supply has failed and RPS is active.
3. **Port Status LED, STAT** – Indicates port status mode is on when green. It normally is green by default. When selected, the LEDs will display different colors for different meanings. If the LED is off, there is no link, or the port has been shut down. If the LED is green, a link is present. If the LED is blinking green, there is activity and data is being transmitted inbound or outbound. If the LED is alternating green-amber, there is a fault in the link. If the LED is amber, the port is blocked to ensure a loop does not exist in the forwarding domain, this will usually happen for the first 30 seconds when a port is activated. If the LED is blinking amber, the port is blocked to prevent possible loop in the forwarding domain.
4. **Port Duplex LED, DUPLX** – When the LED is green, it means the ports are in duplex mode. When selected, the port LEDs that are off are in half-duplex while if port LEDs are green, they are in full-duplex mode.
5. **Port Speed LED, SPEED** – Indicates current port speed. If LED is off, the ports operate at 10Mb/s. If the LED is green, the ports are operating at 100 Mb/s. If the LED is blinking green, the ports are operating at 1000Mb/s.
6. **Power over Ethernet (PoE) Mode LED, PoE** – If PoE is supported on that switch; the LED for it will be active. If the LED is off, then PoE mode is not active, all ports have been denied power, or it is faulty. If the LED is blinking amber, PoE is not selected but at least one port has been denied power, or something is wrong with PoE. If it is green, PoE is active and port LEDs will have different meanings. If the port LED is off, PoE is not active. If the port LED is green, PoE is on. If the port LED is alternating green-amber, PoE is denied because it requires too much power. If the LED is blinking amber, PoE is off due to an error. And finally, if the LED is amber, PoE for the port has been enabled.



When preparing for switch management, the switch must be ready to be configured with an IP address and subnet mask. Remember that to manage a switch from a remote network, the switch must have a default gateway as well. The **switch virtual interface (SVI)** on S1 should be assigned an IP address. The SVI is a virtual interface, not a physical port on the switch. SVI is a similar concept to VLANs. VLANs being logical numbered groupings that can be assigned to certain interfaces. The same thing happens with SVI and switch interfaces.

By default, the switch is configured to have all management via VLAN 1, so all ports are assigned to VLAN 1 on a switch. A good security practice is to use a VLAN other than VLAN 1 for management.

 **Note:** These IP settings are only used for remote management. Remember that switches are Layer 2 devices.

Configuring Basic Switch Management Access with IPv4

The whole point of the creation of a management VLAN is to be able to connect to switches remotely, and then proceed to configure them from a remote location. There are particular steps that need to be taken before you begin to configure a management VLAN. For instance, we need the management interface IP address and subnet mask. We also need to know the interface we'll be configuring. For this example, we'll use the IP address 192.168.1.11 and 255.255.255.0 for the subnet mask. The VLAN we will use is VLAN 99.

When you boot up the switch, you will need to go into global configuration mode, which can be done like this:

```
S1# configure terminal
```

From that point, you should be in global configuration mode, to ensure you are, the prompt will change from S1# to S1(config)#. Next, we need to jump into the VLAN interface:

```
S1(config)# interface vlan 99
```

After that, your prompt will change from S1 (config)# to S1 (config-if)# to show you that you are now configuring an interface. For this part, we will need to set up the IP address for our VLAN 99 interface:

```
S1(config-if)# ip address 192.168.1.11 255.255.255.0
```

The thing about VLAN interfaces is that they are not on until you set them as active; issue the following command:

```
S1(config-if)# no shutdown
```

From that point, your **switch virtual interface (SVI)** for that VLAN will be enabled and an IP address has been set. Remember that these interfaces are **logical** and not physical.

There are some additional steps that you should take to configure the interface. For instance, this VLAN will not be accessible until there is a physical interface assigned to it. Before we get to that, there are some additional steps to help you manage, for instance, you can give a VLAN a name through global configuration mode:

```
S1(config)# vlan vlan_id
```

At that moment, you are entering VLAN configuration mode, *note that vlan_id* is where you will put the VLAN number, for instance, our VLAN we're currently working with is VLAN 99. Next, we'll give it name like so:

```
S1(config-vlan)# name vlan_name
```

From this point, you can do the exit command and go back into global configuration; next, we need to assign the VLAN to a physical interface. That interface will be the only port that can access the switch through remote access:

```
S1(config)# interface interface_id  
S1(config-if)# switchport mode access  
S1(config-if)# switchport access vlan vlan_id
```


The next step to this is to configure a default gateway. This is so that the switch can be managed remotely from networks not directly connected to that switch. *The default gateway will be the router the switch is connected to.* The switch will forward its IP packets to the networks outside of its own network via the default gateway, or, the router. We would have the IP address assigned to the router already, so let's say that we have the IP address for the router interface, the IP for it is going to be 192.168.1.1, so S1 is going to forward all packets going outbound to 192.168.1.1, which is the interface that S1 is using to connect to on R1. It sounds confusing I know, but it's not. To configure the default gateway, follow along:

S1# configure terminal


The first step of it is to hop onto global configuration mode. The next step is rather simple:

S1(config)# ip default-gateway 192.168.1.1

That's it for that step really. We already discussed how our default gateway would be 192.168.1.1, so we had that information. What I wanted to state next is **extremely important. Saving.** You need to save your configuration that you're currently running to ensure that on boot, your settings you had configured are going to stay in NVRAM. To do this, ensure that you are in only configuration mode (meaning, it'll look like how it is below, to go down in configuration modes, issue end until you go into the basic configuration mode below).

S1# copy running-config startup-config

Thought, the regular configuration mode is just the hostname and # symbol next to it. When you issue that command, it will copy the running-config (current changes) and overwrite your startup-config (what was last saved on the switch). The importance of ensuring any changes you made stick is **very, very important.**

 **WARNING!** Save your settings by issuing **copy running-config startup-config** in basic configuration mode.

I'm glad I've stressed the importance of saving your configurations. Now, the final thing we can do is ensure that our interfaces have been configured with the IP address, subnet mask, etc. To verify, issue the following command:

S1# show ip interface brief

It should give you some results that look like this if done correctly:

S1# show ip interface brief

Interface	IP-Address	OK?	Method	Status	Protocol
VLAN 99	192.168.1.11	YES	manual	up	down

This is probably some of the most simplistic switch configuration you can learn, but this pretty much shows you that our interface, VLAN 99, has an IP address of 192.168.1.11 and is working well. It's active and everything, ready for remote management if needed.

Duplex Communication and Auto-MDIX

There are two different types of communications that can happen, full-duplex and half-duplex. Full-duplex communication makes the efficiency of a LAN great overall. It increases bandwidth by allowing both ends to receive and send data at the same time, this is known as bidirectional, and this method requires the optimization of the network we discussed, micro-segmentation. A network is micro-segmented (which happens when you use switches, because if you remember, *switches create mini collision domains*) when a single device is connected to a single switch port, thus allowing it to run at full-duplex.

This results in the mini collision domains, and since there is only one device, the micro-segmented LAN is free of any collisions. Unlike full-duplex, you would expect that half-duplex is basically vice-versa, essentially it is. Half-duplex communication is unidirectional, meaning that data cannot be sent and received at the same time. This creates issues with performance due to the fact that data can only flow in one direction, resulting in potential collisions. Half-duplex technologies are seen in older hardware, like hubs, the fact that hubs would only be considered a repeater and would always just send data out.

Most types of networking technologies will offer full-duplex. Technologies like Gigabit Ethernet and above will require full-duplex to work. In full-duplex, collision detection on the NIC is disabled. Frames sent by two devices will not collide due to the fact that the devices use two separate circuits in network cables, so, full-duplex connection will require switches that support full-duplex or direct connection using an Ethernet cable between two devices. Full-duplex will offer 100 percent efficiency in both sending and receiving, resulting in a 200 percent potential bandwidth use.

Switch ports can be manually configured with settings specifying things such as duplex or speed. For this example, you'd first have to go into global configuration (as we go along in this book, I'm hoping you'll remember how to access global configuration, if not, use **configure terminal** to do so) and from there, access the interface you'd like to work with:

```
S1(config)# interface FastEthernet 0/1
```


From there, I'm assuming that the interface we're accessing is FastEthernet 0/1, or, Fa0/1 (It's a physical port by the way.) At that point, you will now be in interface configuration mode, let's say we wanted to set our duplex communication to full, you would issue the following command to modify duplex type:

```
S1(config-if)# duplex full
```

That command will change your duplex type to full duplex. The speed setting for both duplex and speed on the Cisco Catalyst 2960 and 3560 are normally automatic. The 10/100/1000 ports operate in either half- or full-duplex mode when they are set on 10 or 100 Mb/s, however, anything above 1000 Mb/s will operate full duplex. If you wanted to change the speed of the interface, issue the following command in the interface configuration mode:

```
S1(config-if)# speed 100
```

Auto-negotiation is useful when the speed and duplex settings of a device are unknown or will end up changing, as a result, the best practice is to configure your known devices manually. Those devices include servers, workstations or general network devices.

 **Note:** Speeds and duplex type should be the same between devices. Mismatched settings can result in connectivity issues. Auto-negotiation failure creates these types of failures.

The next thing we're going to talk about is cable types. Certain cable types have been required when connecting certain devices. For example, switch-to-switch or switch-to-router required different types of cables (straight-through or crossover). Using a feature called Auto-MDIX (which means automatic medium-dependent interface crossover) can provide a solution to this problem. When auto-MDIX is enabled on an interface, the interface will automatically detect the required cable connection and proceed to configure the connection appropriately. If you need to connect something without the Auto-MDIX feature enabled, you will need to consider the following.

You can connect switches to the following with a straight-through cable when auto-MDIX is not configured:

- Servers
- Workstations
- Routers

You can also use switches to connect to the following devices with a crossover cable when auto-MDIX is not configured:


- Switches
- Repeaters

So in a nutshell, *switch-to-switch communication (and switch-to-repeater) will require the crossover cable, while the switch-to-anything else will require a straight-through cable.* Now, with auto-MDIX configured, the type of cable can be used to connect other devices, and the interface will adjust accordingly to communicate with that device via that cable.

To configure auto-MDIX on a router or switch, you will follow the following steps beginning with global configuration:

```
S1(config)# interface fastethernet 0/1
```

To begin, you'll need to access the interface you'd like to connect.

 **Pro Tip!** You can shorten the commands issued on Cisco IOS. For example, instead of `interface fastethernet 0/1`, you can use a command like `int fa0/1` to access the interface.

```
S1(config-if)# duplex auto
S1(config-if)# speed auto
S1(config-if)# mdix auto
```

From that point, auto-MDIX should be configured on that interface. **Don't forget to save your changes with `copy running-config startup-config`** in basic configuration mode if you haven't! To verify if the settings are running correctly, execute the following command in basic configuration mode:

```
S1# show controllers ethernet-controller interface interface_id phy | include Auto-MDIX
```

Above we had configured fastethernet 0/1, so if we wanted to see our changes, the command would be the following:

```
S1# show controllers ethernet-controller fa 0/1 phy | include Auto-MDIX
```

Verifying Switch Port Configuration and Network Access Issues

Sometimes, you may miss a thing or two whenever you're working with your commands and whatnot. To be able to troubleshoot efficiently can be tough work if you don't know where to start. One of the ways you can troubleshoot is if you have some ways to view all of your configurations. Here is a list of configurations you can use:

Display interface status and configuration.	S1# show interfaces interface_id
Display current startup configuration.	S1# show startup-config
Display current operating configuration.	S1# show running-config
Display information about flash file system.	S1# show flash
Display system hardware and software models.	S1# show version
Display full command history.	S1# show history
Display IP configuration for certain interface.	S1# show ip interface_id
Display MAC address table.	S1# show mac-address table or show mac address-table

Instead of **show**, you can actually just put **sh** and it will be the same thing. The whole point of the show command is to help you verify your configurations. This table above can help with a wide variety of configuration help you might need. For example, if you wanted to view your current configuration, you would use **show startup-config** or **show running-config**. If you wanted to view the details of a certain interface, you would use **show interfaces interface_id** and view all the data you will need to know. If we wanted to view, let's say, fastEthernet 0/1, we would do **show interface fastethernet 0/1** in the command line.

When it comes to dealing with errors, you can find them very quickly just by looking at the interface status. For example, one of the most important details you will need can be viewed with **show interfaces interface_id** due to the fact that it can tell you whether an interface is active or not. If the interface status and line protocol status are both up, then the port is fully operational. If they are both down, then it may have been administratively down or have a problem.

There are various problems that you may find in the output of that command, it will be in one of the output omitted sections, and we're going to identify a couple of them and what they are:

Error Type	Description
Input Errors	The total number of errors on an interface. This includes runs, giants, no buffer, CRC, frame, overrun, and ignored counts.
Runs	These packets are discarded due to the fact that they are less than 64 bytes, which is a runt.
Giants	These packets are discarded due to the fact that they are more than 1,518 bytes, which is a giant.
CRC	CRC errors are generated when the devices checksum is not the same as the recieved checksum.
Output Errors	All of the errors that prevented the final transmission datagrams out of the interface.
Collisions	Number of messages that required retransmission due to collision over Ethernet.
Late Collisions	Collisions that occur <i>after</i> 512 bits of a frame were transmitted.

I'm going to talk about each of these types of errors in detail, but the output from various commands, such as the **show interfaces** command can help fix or identify the fact that many of these problems existed or worked on. For instance, the **show interfaces** command can help you with problems if the interface is up, if it has a different encapsulation type, if the interface has been shut down through administration, etc.

Input errors are all of the errors that would be found in datagrams that were received on the interface. This includes runs, giants, CRC, no buffer, frame, overrun, and ignored counts. Those reported errors can be viewed once again with the `show interfaces` command. We can talk about some of those in detail below:

- **Runt Frames** – These are frames usually shorter than the 64 byte minimum for frames. Malfunctioning NICs are usually the reason runs can exist, they're also caused by excessive collisions.
- **Giant Frames** – Giants are Ethernet frames large than the 1,518 byte maximum allowed. Giants usually appear the same way Runt frames would show up.
- **CRC Errors** – On Ethernet and serial interfaces, the CRC error will usually show up due to errors with the cables or interfaces. This can mean there is interference, loose or damaged cabling, or even the wrong type of cable. If you are getting many of these errors, you should inspect the interface affected, the type of cable, and do any checks for damage.

Output errors are all of the errors that did not get transmitted due to having some form of error being detected before exiting out of the interface, you can view the errors with the `show interfaces` command, some in detail are:

- **Collisions** – If you are dealing with half-duplex communication, then collisions are nothing to worry about as long as half-duplex communication is working out well. The problem is that full-duplex is a highly recommended over half-duplex as the communication type to implement.
- **Late Collisions** – This will mean that a collision happens after 512 of the bits transfer. One of the primary causes for this is excessive cable length. Another possible cause for this is duplex communication has been incorrectly configured, such as a half-duplex configuration attempting to communicate with full-duplex.

Basic Troubleshoot Concept in the Network Access Layer

If there is issues with your network after you've set it up, you may have done something on accident that you didn't intend to do during installation. There are also problems such as damaged cabling, new configurations, or new devices, then you'll need to provide maintenance for your systems. To troubleshoot your items, here is a quick list of methods you can follow:

1. Use the **show interfaces** command and check if the interface you're working with is up.
 - a. If the interface is up, check for any radio frequencies that could be disturbing it and remove them. Also check if the duplex settings are the same on both ends.
 - i. If the problem is solved, you're good to go!
 - ii. If the problem is not solved, you'll have to research the issue and document it.
 - b. If the interface is down, check to see if the cables are configured correctly. Check if they're connected with the proper cable (refer to page 44), and check if the speeds are the same on both ends.
 - i. If the problem is solved, you're good to go!
 - ii. If the problem is not solved, you'll have to research the issue and document it.

Secure Shell (SSH) Operation and Configuration

The problem with using Telnet at this point is the fact that you can capture Telnet packets and read them in plain text. To answer our first question, Telnet is a protocol that operates on TCP port 23. It is used for remotely accessing anything that has been configured for remote access via Telnet. The problem with Telnet is that it uses plaintext transmission for

everything, meaning, when you send a username and password over Telnet to authenticate, there is no encryption to protect it from interception.

At this point, Telnet should have been replaced with Secure Shell, known as SSH. SSH is far stronger than Telnet due to the encryption that it comes with. SSH provides a strong security for remote connections when doing anything from connecting to configuring the device. SSH runs on TCP port 22 and provides far more security than Telnet.

To identify if your switch can allow SSH features over Telnet, the switch must be using a version of the IOS that includes the ability to encrypt. To do so, you can run the following command during any configuration:

```
S1> show version
```

The output may be the following, what you're going to look for is **K9** in the IOS filename, which should look like this:

```
S1> show version
Cisco IOS Software, C2960 Software (C2960-LANBASEK9-M)
```

Now we're going to begin the configuration of SSH on a switch. Before doing so, there are some things that you will need to set up, such as a unique hostname and the correct network settings.

The **first step** is the verification of SSH support. *Before we do that, ensure that you are in global configuration mode.* Now, issuing the following command to view if your version of IOS supports cryptography, if you receive an output, your switch will allow SSH configuration, if not, the command will not be recognized:

```
S1# show ip ssh
```

Once again, that command should verify if SSH works. The **second step** is to configure the IP domain name. You can supply whatever you'd like as long as it is a domain name, for this example, we'll use netacad.com.

```
S1(config)# ip domain-name cisco.com
```

If you've established the IP domain name, the **third step** is going to be the generation of RSA key pairs. Now, not all versions of Cisco IOS support SSH version 2, and SSH version 1 does have some security flaws in it, so, if it is possible, set your SSH version to version 2 with the following command in global configuration:

```
S1(config)# ip ssh version 2
```

Generating an RSA key pair will automatically enable SSH. You will need to use the next command in global configuration:

```
S1(config)# crypto key generate rsa
```

When you generate the RSA keys, you will be prompted as to what to enter for the modulus length. **Cisco recommends a minimum modulus size of 1,024 bits.** The longer a modulus is, the more secure it is. The longer it is, the longer it will take to generate and use. To delete the RSA key pair you currently have, use the following command in global configuration:

```
S1(config)# crypto key zeroize rsa
```

If you delete the current RSA keys you have on your switch, SSH is automatically disabled due to lacking encryption. The **fourth step** is to configure user authentication. An SSH server on the switch can authenticate users through local

authentication *or* the use of an authentication server. For local authentication, the first thing you will need to do is configure a username and password in global configuration mode, like so:

```
S1(config)# username username secret password
```

In the next line, I'll type you an example:

```
S1(config)# username daniel secret password
```

The **fifth and final step** I suggest you pay a lot of attention to. I need to establish that there are two different ways you can connect to your switch; there are physical connections and remote connections. Console lines are for physical access and VTY lines are used to connect remotely. Virtual Terminal Lines (known mostly as VTY) are 1 of 4 main types of Standard Asynchronous Lines (TTY) used only to control inbound connections.

VTY is virtual, or, being software, there is no hardware with it. You have up to fifteen VTY connections available on your switch (meaning you can have fifteen simultaneous connections at a time) these connections go from 0 to 15. To modify one of the VTY lines, simply issue the following command in global configuration:

```
S1(config)# line vty 0
S1(config-line)#
```


Once you have done that, you should be in the line configuration interface for that line, if you wanted to modify multiple line's (for this example, VTY lines 0 to 15), you would issue the following command:

```
S1(config)# line vty 0 15
S1(config-line)#
```

If you wanted to deny all remote access to all vty lines, this would be a way to configure all of them at once. Right now, we want to learn how to establish SSH over telnet on all of our VTYS, how we would do this is listed below:

```
S1(config)# line vty 0 15
S1(config-line)# transport input ssh
S1(config-line)# login local
S1(config-line)# exit
```

This configuration prevents anything that is not SSH and limits the switch to *only* accept SSH connections. The **line vty** command will allow you to access those lines to configure them. The **login local** line configuration mode command is used to require local authentication for the SSH connection, meaning you can *only* access using credentials we have created, for instance, our user *daniel*. You can **verify SSH with** `show ip ssh` and `show ssh` in global configuration.

 **Reminder!** Unless told otherwise, ensure SSH version 2 is being used, you disabled unused VTYS, and **you saved your configuration!**

Common Security Attacks

Even if these aren't workstations or anything, a network infrastructure is very important. Deny access to others on the network or interrupting the flow of the network can cause a denial-of-service, known as a DoS. Back to the subject, basic switch security does *not* stop malicious attacks. Security is a process that will prevent an attacker from doing what they need to do, but no matter *what*, the attacker will eventually win. They might win, but it may be a short victory for those who cause problems if the team of network professionals are educated correctly on the threats an attacker may pose against an organization.

Soon, we'll be looking into the CCNA security course, for now, we can get a brief look at the types of security attacks a modern infrastructure may be victims to, and how we can protect against them.

MAC Address Flooding

This is one of the types of security attacks that may go against a switch. As we know, the MAC address table contains MAC addresses and their associated ports along with VLAN assignment for each port. When a frame is received by the switch, it will check if it has the source address. If the source address does not exist on the table, the switch will record the source address into the table and attempt to send the packet outward. If the frame's destination address does not exist on the MAC address table, the switch cannot send it to its location, and will proceed to flood all ports except for the port that frame originally came from to find the destination.

When MAC addresses begin to flood the switch, the MAC addresses will overflow into the table and use up all of the entries on the table. Since MAC address tables are limited, MAC address table overflow will easily take up all entries in the table quickly with fake source MAC addresses. For example, an attacker would use a tool to randomly generate source and destination MAC addresses and proceed to send them to the switch. The switch will update the table until it is full. From that point, the switch will enter a mode called *fail-open mode*. In this mode, the switch will broadcast all frames to all machines on the network, ultimately resulting in the attacker having the ability to see all frames from all hosts.

To prevent this attack, the configuration of ports is required.

DHCP Spoofing

From what we know, DHCP is a protocol that assigns hosts an IP address from a pool of IP addresses. The problem with this is that all it takes is one computer to take every single IP address. This type of attack is called DHCP starvation. This is when an attacker floods a DHCP server with requests to use all addresses, when no more IP addresses are available from the pool, this results in a denial-of-service, or, DoS. A DoS attack is when an attack overloads specific resources, preventing legitimate requests from being accessed.

Another type of attack is a DHCP spoofing attack. This is when an attacker configures themselves as a DHCP server. The main use of this type of attack is having the ability to impersonate another very important service, DNS, known as Domain Name System. Another reason for this type of attack is to hijack traffic and take control of users, becoming the gateway.

To prevent this attack (or, mitigate the risk), use DHCP snooping and port security configurations on your switches.

Leveraging CDP

Cisco Discovery Protocol (also known as CDP) is a protocol created by Cisco for all Cisco device configurations. The use of CDP is to discover other Cisco devices, which will allow them to auto-configure. Sometimes, this will make things easier. By default, most Cisco routers and switches will have CDP allowed on all ports. CDP information is sent in periodic broadcasts. The fact that this information is sent everywhere *and* unencrypted, it may pose a risk. The information is updated in the CDP database of each device. CDP is a layer 2 protocol, so routers will *not* forward these broadcasts.

CDP broadcasts contains various amounts of information about the system, such as the IP, IOS software version, native VLAN, and various other specifications. This type of information can give an attacker enough information to the point where they can execute an attack. An attacker could also forge CDP packets to give false information, or, they could use to launch specific attacks.

It's highly recommended that you disable CDP on all ports. To do so, you'd have to configure *all* ports, for example, if I'm going to configure a 24-port switch and want to disable CDP, I would issue the following commands:

```
S1(config)# int range fa0/1 - 24
S1(config-if-range)# no cdp run
```

Additional Attacks

There are additional attacks that an attacker can execute upon a device. For example, the use of telnet is discouraged since it authenticates and sends information in plain text, resulting in making information visible to anyone who can see it. This is a telnet attack, to mitigate this attack, use Secure Shell (SSH).

There are also Brute Force Password Attacks, which will allow someone to attempt to break a password using various combinations of passwords in a very quick manner, whether through dictionary, rainbow table, or just using every combination. To mitigate this attack, use strong passwords and change them frequently, also consider using a very complex password and limit access with Access Control Lists (ACL, learned later.)

Telnet DoS attacks can be run against a device by an attacker. This attack prevents the use of telnet by anyone by exploiting a flaw in the system. Most of the time, *attacks are combined operations*. These vulnerabilities are usually resolved with security patches. ***It is the best practice to use SSH instead of telnet.***

Best Practices



Reminder! The Best Practices are very important! Pay close attention to what they say!

The best practices to mitigate these attacks (and all attacks in general) can include things such as creating a security policy, turning off unused services and ports, using a strong password **and** changing them often, controlling physical access of devices, avoid using no encryption, perform backups, educate employees, encrypt end-to-end communication and sensitive data, implement security hardware and software (like firewalls) and keep everything up to date with the latest patches.

These are only some of the standards for attack mitigation. *No system is 100% secure*, as a result, everyone is to stay vigilant. Eventually, an attacker will break through, it's up to you and your systems to see how long they will linger.

Network Security Practices

When checking the security of the network, it is common to hire someone to initiate a *security audit*. A security audit reveals various types of information that could help an attacker figure out how to launch an attack on a network. For example, network security tools such as *packet sniffers* can pick up packets on the wire. Other tools can initiate the attacks we had just talked about. Security tests are usually initiated in very controlled environments.

There are various basic techniques you can initiate to increase security of your network that may impact the network much more than you might believe. For example, disabling ports may assist in preventing intrusion simply because a denied port will cut off access to the switch. You will want this if not all ports are being used.

Interface Security

To remove access to the port, you can do an administrative shutdown on the port by accessing its interface. *At this point, it's more than likely that you know how to do that*, so we won't go over that. What we *will* go over is how to access multiple interfaces at once. How we did this before was using the **int range** command. This can be done as followed:

```
S1(config)# int range fa0/1 - 24
S1(config-if-range)# shutdown
```

From there, all interfaces would be shutdown.

DHCP Snooping

DHCP snooping is a feature that was added Cisco switches to determine which ports can respond to DHCP requests. You can identify a port as *trusted* or *untrusted*. *Untrusted ports* can forward requests while *trusted ports* host a DHCP server or can be an uplink towards the DHCP server. Ultimately, DHCP snooping allows a switch to build a DHCP binding table, mapping client MAC addresses, IP addresses, VLAN, and port IDs for all *untrusted* ports. Any port that is considered untrusted attempting to send a DHCP offer will be shut down.

To set up DHCP snooping, you will need to issue the following command in global configuration:

```
S1(config)# ip dhcp snooping
```

The next step is to enable DHCP snooping for specific VLANs (*we're also going to assume VLANs 10 and 20 are configured for Fast Ethernet 0/1*) for this, we'll use VLANs 10 and 20:

```
S1(config)# ip dhcp snooping vlan 10,20
```

The next step is to define which ports are trusted and which are untrusted. For this example, we will say that fa0/1 is a trusted port and fa0/2 is not. We will also configure fa0/2 to make it so that if he sends 5 DHCP OFFER requests, his port will be shutdown. **Unless** a port is declared trusted, it will be automatically considered *untrusted*.

```
S1(config)# int fa0/1
S1(config-if)# ip dhcp snooping trust
S1(config-if)# int fa0/2
S1(config-if)# ip dhcp snooping limit rate 5
```


Let's assume that Switch 2 hosts a DHCP server. Remember that *untrusted* ports can only forward requests that were already created. On Switch 2, we'll need to configure the port that the DHCP server is connected to so it can be secure. To do this, we're going to do the following on Switch 2:

```
S2(config)# ip dhcp snooping
S2(config)# ip dhcp snooping vlan 10,20
S2(config)# int fa0/1
S2(config-if)# ip dhcp snooping trust
```

From that point, you should be completely set with snooping. *Be sure to enable snooping on the port that DHCP OFFERS will be coming from!*

Port Security

Ports should be secured before a switch is deployed in an infrastructure. We've already talked about a couple methods of security, such as administrative shutdown, Secure Shell Implementation and DHCP security. The next thing we're going to be talking about is Interface Security, or, Port Security. Port Security limits the amount of MAC addresses on a port. It can be configured to allow a certain amount of MAC addresses per port, if any more MAC addresses attempt to pass the limit, a security violation is reported. There are a couple ways to secure MAC addresses, some of these include:

 **Note:** Before initiating any interface configurations, issue the following command in the interface mode:

```
S1(config-if)# switchport mode access
```

- **Static secure MAC addresses** – This means that the MAC addresses were configured manually on that port. The address is added to running configuration and kept. To configure a static MAC address, do the following:

```
S1(config)# int fa0/1
S1(config-if)# switchport port-security mac-address mac-address
```

- **Dynamic secure MAC addresses** – This is when a MAC address is dynamically learned and are only stored into the address table. When the switch shuts down, the addresses learned are removed.
- **Sticky secure MAC addresses** – This is configured on an interface to convert dynamic MAC addresses into sticky secure MAC addresses. This adds them to the running configuration.

We're going to go in-depth into Sticky Secure MAC addresses. The first step is to enable this port security on an interface. You can issue that command with the following command:

```
S1(config-if)# switchport port-security
```

To begin configuration for Sticky Secure MAC addresses, you will need to enable sticky learning by issuing the following command in interface configuration mode:

```
S1(config-if)# switchport port-security mac-address sticky
```

Once you've issued that command, the interface will now convert all addresses learned into sticky addresses. Not only will they be in the address table, but they will be in running configuration as well. Sticky MAC addresses can be manually as well with the following command:

```
S1(config-if)# switchport port-security mac-address sticky mac-address
```



Reminder! Remember that static and sticky MAC addresses are placed in *running configuration*, that means the configuration must be saved with the following command if you want to keep the addresses. The command to save configuration is **copy running-config startup-config**



Pro Tip! You can issue **copy run start** to save your configuration as well. It's the same thing as **copy running-config startup-config**.

Sticky secure MAC addresses can be recorded up to a certain limit. When the maximum number of secure MAC addresses has been added to an interface or an address that was configured on one interface is seen on another interface, it creates a security violation.

There are different approaches the security system will take in the case of a violation:

- **Protect** – When the number of secure MAC addresses reaches the limit, packets with addresses that are not a part of the address table are dropped. There is no notification that this has happened.
- **Restrict** – When the number of secure MAC addresses reaches the limit, the same thing happens as protected. The only difference is that a syslog message is logged.
- **Shutdown** – This is the default violation mode. This will immediately cause the port to shut down and send a log message. It will stay shut down until it is turned on again with a **no shutdown**.

If you want to change the violation mode on a switch port, use the following command on the interface:

```
S1(config-if)# switchport port-security violation protect / restrict / shutdown
```

Now, by default, the maximum number of secure MAC addresses is 1, this feature is disabled by default, the default violation mode is shutdown. If you want to change the amount of secure MAC addresses on a port, use the following command on the interface you are configuring:

```
S1(config-if)# switchport port-security maximum value
```

The final step for working with sticky security is ensuring the settings you've configured stay. To do this, you can issue the following command in basic configuration (for this example, I'll use Fast Ethernet 0/10):

```
S1(config-if)# show port-security int fa0/10
```

If you want to check your sticky secure MAC addresses and the interfaces that they're related to, issue the following command in basic configuration:

```
S1(config-if)# show port-security address
```

Network Time Protocol

To be able take down accurate audits, you need the correct time for all devices. Not only does the correct time influence logging, but it also works with digital certificates. How time is kept is by using Network Time Protocol, or, NTP. It is used to synchronize clocks of computers over the networks. Network Administrators will typically set their own network master clock, if this cannot be done, there are other options. For this example, let's say Router 1 (R1) is a master clock server and R2 must follow along.

The first step is to register a device as a master clock server. To do this, we need to execute the following command:

```
R1(config)# ntp master stratum
```

The stratum is a value from 1 to 15, the stratum is a number that the system claims. If the system is configured as a master and no stratum is specified, it will manually set to 8. If the NTP master server cannot reach any clock with a lower stratum number, the system will claim to be synchronized at the current number and other systems will synchronize to it with NTP. If you want to configure a client, you would issue the following command:

```
R2(config)# ntp server ip-address
```

In privileged EXEC mode, you can issue the following to receive more information on your status with the NTP master:

```
R2# show ntp associations  
R2# show ntp status
```

Each of these commands will give you information about your NTP associations. The first command will display the IP address of the NTP master and other information. The second command will more in-depth information about the NTP server.

Catch-up

Let's talk about a couple things. At this point, you've completed ITN 3, 4, and 5, RSE 1 and 2. You should have a basic understanding of networking. The next couple days are going to get rough, there's going to be a lot of information that you will have to learn. So far, I've written 54 pages and over 27,000 words. This is a lot of information. You can only get good at this by studying.

From now on, I'm going to assume you understand what privilege EXEC mode is (S1#, R1#), how to shorten some things (do sh run, copy run start, int fa0/1, etc.), what global configuration is (S1(config)#, R1(config)#), what interface configuration is (S1(config-if)#, R1(config-if)#), basic networking, the OSI and TCP/IP model, etc.

You should also know the expression of addresses, various network protocols, the systems of IP addressing, subnetting, VLSM, network planning in practice, Access, Distribution and Core layers, general concept switching, how switches work, collision and broadcast domains. The basic configuration of switches and routes should also be down.

There are a lot of things you need to know. Hopefully taking notes and consistently reading will get you far in this, never forget though, the most important thing you need to do is *practice, practice, practice*. Let's jump into RSE 3, VLANs.

Prologue

Network performance is important when it comes to an infrastructure, as is security. By design, routers block broadcast traffic from going any further than it needs to, the problem is, you usually have less routers than switches. The whole point of a router is to move information from one network to another, not provide network access to end devices.

This is where we can implement a certain feature into switches called VLANs, also known as virtual local area networks. These can be used to reduce the size of broadcast domains. VLANs are usually implemented into a network to help divide networks into logical links.

VLAN Terminology

With VLANs, you can logically separate interfaces so they can communicate with only certain computers. Devices within certain VLANs act like they're their own network. Any interface can be assigned to a VLAN, unicast, broadcast and multicast packets are only sent to corresponding VLANs.

VLANs can improved network performance by separating large broadcast domains into smaller ones, if a device in one VLAN sends a broadcast Ethernet frame, the only devices that will receive the broadcast will be those within the VLAN.

VLANs provide a variety of benefits that could really work with your infrastructures growth and development. The first benefit is **security**; VLANs allow the separation of sensitive data. Second is **reduced cost**, meaning it will cost less to initiate certain upgrades and increase efficiency of current equipment. The third benefit is **better performance**, dividing a layer 2 network into multiple groups increases the efficiency of traffic and boosts performance overall. The fourth benefit, which may be considered extremely important is **shrinking broadcast domains**, VLANs allow only specific VLANs to receive broadcast frames. The fifth benefit is **improved management efficiency**, the reason for this is because VLANs are now easier to manage and administrate because of the grouping of hosts for easier identification.

Each VLAN in a switched network corresponds to an IP network, so, VLAN design must be considered when planning a hierarchical network addressing scheme. Hierarchical network addressing means that the IP network addresses are applied to network segments or VLANs in an orderly fashion for the entire network.

Types of VLANs

There are different VLANs that can be classified as certain traffic classes.

- **Data VLAN**
A data VLAN is used to carry user traffic; this does *not* include voice or management traffic.
- **Default VLAN**
All ports on a default configuration will automatically go to the default VLAN, or, VLAN 1.
- **Native VLAN**
The native VLAN is assigned to a 802.1Q trunk port. Trunk ports are links between switches that support the traffic of more than 1 VLAN. In the case that there is a switch-to-switch connection, the ports connecting those two switches will need to be set as a VLAN port and configured specifically for trunking.
- **Management VLAN**
A management VLAN is used to configure a switch via certain protocols, such as HTTP, Telnet, SSH, or SNMP (Simple Network Management Protocol). By default, VLAN 1 is the management VLAN, this cannot be allowed as it poses as a security risk. To create a management VLAN, you must create a dedicated interface for it.

- Voice VLAN

A separated VLAN is required for the use of Cisco Phones. Cisco phones use VoIP (Voice over IP) to communicate, to meet these requirements, the entire network has to be designed to support VoIP, to support VoIP, a network will require the following:

- Bandwidth for voice quality
- High transmission priority
- Ability to be routed around congested areas
- Delay of less than 150 ms across the network

- Black Hole VLAN

This type of VLAN is configured for unused ports. When you create a VLAN with no route or way outside.

VLAN Trunks (Brief)

To clarify once again, the trunk port is the link between switches. A VLAN trunk extends VLANs across an entire network using IEEE 802.1Q for trunk ports using Fast Ethernet, Gigabit Ethernet, and 10-Gigabit Ethernet.

Without VLAN trunks, devices on different switches and same VLANs would not be able to communicate, if there was no trunk port, they would need a router to communicate. VLAN trunks do not belong to specific VLANs, much rather, it is a port that you can assign multiple VLANs to, and this will make sense when we configure VLANs.

VLAN Tagging

With Layer 2 devices comes Ethernet frames. An Ethernet frame contains various bits of data. Ethernet frames do not contain information about the VLAN to where the frame belongs, much rather, when Ethernet frames travel through a trunk, information about the VLAN is added. This is called *tagging*; it is done with IEEE 802.Q1 Ethernet frames. They have a 4-byte tag inserted, specifying the VLAN to which the frame belonged to.

When switches receive a frame on a port configured in the trunk mode, the switch inserts the VLAN tag, recalculates the FCS (frame check sequence) and sends the new frame on its way.

The frame will look like this with the following data; a Ethernet frame that doesn't use VLANs will lack the Tag field:

Destination MAC	Source MAC	Tag (added for VLANs)	Type/Length	Data	Frame Check Sequence (FCS)
-----------------	------------	-----------------------	-------------	------	----------------------------

These are the specific items within the Tag field:

Type – A 2-byte value called the tag protocol ID, or, TPID. It is set to hexadecimal 0x8100.	User Priority (PRI) – 3-bit value that supports this implementation	Canonical Format Identifier (CFI) – 1-bit identifier that enables Token Ring frames	VLAN ID (VID) – A 12-bit VLAN identification that supports up to 4096 VID's.
--	---	---	--

When setting trunk ports, you should *always* ensure that the native VLAN number is the same for *all* switches. Failure to do this will result in the trunk port dropping the frame. When a trunk port receives a frame *that has not* been tagged, it is sent to the native VLAN, if there is not devices associated with the native VLAN, then the frame is dropped as well. For example, if VLAN 99 is the native VLAN, and an untagged frame is sent, that frame will be sent to anything in VLAN 99.

Let's talk about *VoIP* tagging. An access port that is used to connect a Cisco IP phone can be configured for two different VLANs. How this works is there are two VLANs configured from the device attached to the phone, one for voice traffic and another for data. The link between the switch and the IP phone act as a trunk to carry both voice VLAN traffics.

Cisco IP phones will have three ports, port 1 is used to connect the switch and other VoIP devices, port 2 is an internal 10/100MB interface for carrying voice traffic, the third port (access port) is used to connect an end device. So the IP phone is right in the middle of all of it, on the switch, the access is configured to send Cisco Discovery Protocol (CDP) packets to instruct an attached IP phone to send voice traffic to the switch in three ways:

- In a voice VLAN tagged with Layer 2 class of service (CoS) priority
- In an access VLAN tagged with a Layer 2 CoS priority value
- In an access VLAN untagged without a Layer 2 CoS value

So, you configure your switch port to support voice traffic. First, you assign a VoIP VLAN, next, you set the priority for the voice frames, next, you configure the data VLAN. From there, you connect the switch, Cisco phone and end device with a 3-port switch.

VLAN Implementation

The first step in implementing a VLAN is to understand the ranges for VLANs. There is a **normal VLAN range** that goes from 1 to 1005. These are used for small and medium sized infrastructures. VLANs 1002 through 1005 are reserved for Token Ring and FDDI VLANs. VLANs 1, 1002 through 1005 are automatically created and not removable.

Configurations for VLANs are stored within a VLAN database file called `vlan.dat`, this is stored in flash memory. The protocol used for managing VLANs configurations is called VLAN Trunking Protocol, or, VTP, can only learn and store these normal ranges.

Then, we have the **extended VLAN range**. The extended VLAN ranges are VLANs 1006 to 4094. This is used for large infrastructures. The configurations do not go into the VLAN database and they are more limited than normal VLANs. By default, they are saved in the running configuration and have no support for VTP.

Now, let's explain the creation of VLANs. When implementing VLANs, there is a certain file that these configurations go into called the `vlan.dat`. This is stored in the flash memory; flash memory is persistent unlike start-up configuration. Regardless, you should always initiate the **copy run-config startup-config** command whenever you have a chance.

As per usual, you will need to activate global configuration with the **configure terminal** command.



Pro Tip! You can also issue **config t**, which will get you the same result as **configure terminal**!

The next command you will need to do is issue the **vlan** command:

```
S1(config)# vlan vlan-id
```

What you put in *vlan-id* is the number of the VLAN. Once you enter a VLANs configuration, like so:

```
S1(config)# vlan 100
```

The VLAN will be created and you will jump right into the configuration for it, if you make a VLAN on accident, you can issue the following command to delete it:

```
S1(config)# no vlan vlan-id
```


Naming a VLAN can be done when you're in VLAN configuration mode, like so:

```
S1(config-vlan)# vlan vlan-name
```

The next step when creating and naming a VLAN would likely be assigning ports to that VLAN. Once that port has been assigned to a VLAN, that will be the only VLAN to that port, you can assign the port in this order:

```
S1(config)# int interface-id  
S1(config-if)# switchport mode access  
S1(config-if)# switchport access vlan vlan-id
```

I'll go ahead and explain each step. Assuming you're in global configuration, the first thing you will need to do is go into the configuration of the interface you'd like the VLAN to belong to. I'm using **int** instead of **interface** because it's shorter, just to let you know. So, once you're in the interface that you want to assign a VLAN to, the next command you will execute is **switchport mode access**. This command is used to access port configuration. The next step is to assign the VLAN. You simply replace *vlan-id* with the VLAN you'd like and you're done!



Note: If you assign a VLAN that does not exist, it will create that VLAN.

If you assign a port to another VLAN, it will belong to the new VLAN and no longer be a part of the VLAN you originally had it on. In the event that you'd no longer want a port associated with a VLAN, you can issue this command on the interface:

```
S1(config-if)# no switchport access vlan
```

If you'd like information on VLANs or ports associated with the VLAN, you can issue either command:

```
S1(config)# show vlan brief  
S1(config)# show vlan id vlan-id  
S1(config)# show vlan name vlan-name  
S1(config)# show vlan summary
```

That command is used to show a brief view of current VLAN status, to view a port, use this command:

```
S1(config)# show interfaces interface-id  
S1(config)# show interfaces vlan vlan-id  
S1(config)# show interfaces switchport
```

VLAN Trunks (Configuration)


We talked about trunks and how they're used to carry more than one type of VLAN traffic, so, the next step would be learning how to configure them. Before hand, I only gave a small explanation of what VLAN trunks are in the first place. A VLAN trunk is an OSI Layer 2 link between two switches, to enable trunk links; you need to configure ports on both ends with the same configurations.

The first step is to go into the interface of the port that you want to configure as a trunk (I would recommend the first or last port for important stuff), from there, you can issue the following commands on the interface:

```
S1(config-if)# switchport mode access
```

```
S1(config-if)# switchport mode trunk
```

This will set the port into trunking mode. In more detail, the port enables Dynamic Trunking Protocol (DTP). The next step in configuring a trunk port is specifying the native VLAN that is *not* VLAN 1 (if one is configured in the first place).

 **Reminder!** Don't forget what a native VLAN is! A native VLAN is VLAN assigned to trunking ports! For both switches, they should be the *same VLAN*! Also, VLAN 1 is always the default native VLAN, but it should be changed!

You can configure the native VLAN with the following command on the trunk interface:

```
S1(config-if)# switchport trunk native vlan vlan-id
```

From there, respectively, you should configure the VLANs that are allowed on the trunk link. If you do not specify the VLANs allowed on that network, then there will *not* be communication between switches for the VLANs. For configuring the VLANs allowed on that switch, issue the following command in the interface:


```
S1(config-if)# switchport trunk allowed vlan vlan-id's
```

What I mean by *vlan-id's* is that you can either add one VLAN, or multiple VLANs, here is an example:

```
S1(config-if)# switchport trunk allowed vlan 10
```

or

```
S1(config-if)# switchport trunk allowed vlan 10,20,30,99
```

 **Reminder!** Remember that you need to add the native VLAN to the list of allowed VLANs! Don't forget that this configuration should be the same for the trunk port on the other end, or, at least to a certain extent.

If you want to reset the configuration of that port, simply putting the following on the interface will revert the port back to its original state:

```
S1(config-if)# no switchport trunk allowed vlan
```

```
S1(config-if)# no switchport trunk native vlan
```

If you need to verify the configuration if it works, you can issue a command to view the interface like this:

```
S1(config)# show int interface-id switchport
```

Introduction to Dynamic Trunk Protocol (DTP)

All devices are different to various extends, especially when they're actually different devices. In the event that ports need to be configured dynamically, we use DTP, Dynamic Trunk Protocol. We mentioned DTP when we talked about setting a port into trunking mode, where the port is set into that mode when we set it into trunking mode. DTP is a protocol automatically enabled on Catalyst 2960 and 3560 series switches. Being a Cisco exclusive protocol will result in other switches not being able to work with it.

DTP manages trunk negotiations only if the communicating switch is configure in a trunk mode that supports DTP. **DTP should only be enabled when you're connected with devices that use it as well**, failure to do this will result in improper frames! To enable trunking from a Cisco switch to a device that does *not* use DTP, use the following commands on the interface being configured:

```
S1(config-if)# switchport mode trunk  
S1(config-if)# switchport nonegotiate
```

Interfaces have different negotiated interface modes, this includes dynamic auto, dynamic desirable, trunk and access. The chart below will tell you what type of result you will get based on the end-to-end configurations on both interfaces:


	Dynamic Auto	Dynamic Desirable	Trunk	Access
Dynamic Auto	Access	Trunk	Trunk	Access
Dynamic Desirable	Trunk	Trunk	Trunk	Access
Trunk	Trunk	Trunk	Trunk	Limited Connectivity
Access	Access	Access	Limited Connectivity	Access

Let me explain first as to what each mode is exactly:

- **switchport mode access** – This is a limited nontrunking interface. If a different type of interface (such as trunk) is connected to an Access interface, the result will be limited connectivity. If an interface configured with Dynamic Auto or Desirable is connected, it will result in the interface defaulting to Access, meaning no trunking.
- **switchport mode dynamic auto** – This will allow the interface to switch into trunk mode. If the interface on the other end is set to Trunk or Dynamic Desirable, the connection between them will be a trunk link.
- **switchport mode dynamic desirable** – If this configuration is set, the interface will automatically attempt to convert the link into a trunk link. If anything except for an Access interface connects, the link will be a trunk link.
- **switchport mode trunk** – This is a permanent trunking mode; If anything except for an Access interface attempts to connect, it will result in the link becoming a trunk link.
- **Switchport nonegotiate** – This can be added to your interface to prevent the generation of DTP frames. This can only be set when the interface type is access or trunk.

Understand that while you are in an interfaces configuration, *you can set any of these configurations!* If you would like to view the current DTP mode, you can execute the following command:

```
S1(config)# show dtp interface interface-id
```

 **Note:** It's a good practice to set the interface to **trunk** and **nonegotiate** when you need to have a trunk link. On links that do not need trunking, DTP should be turned off with **switchport nonegotiate**.

VLAN Troubleshooting

One of the biggest issues you can face with VLANs is the incorrect configuration of IP addresses; ensure that VLANs correspond to a unique IP subnet to ensure organization! If there is still no connection between devices in a VLAN, you can take a chance to review your switch configurations.

In the event of no connection, there are some steps that you can take to ensure that configuration is correct:

1. Review if the assigned VLAN on the port is correct, if not; assign the port to the correct VLAN.
2. Once that has been done, review the VLAN database with commands such as **show vlan** or **show interfaces**, if the VLAN does not exist, create the VLAN within the VLAN database.

From there, you can review your connections, but there is more in case of trunk troubleshoot:

1. Check to ensure that the native VLAN on both ends are the same, if not, change the native VLANs to ensure that both ends are matching. You can check the interfaces with **show interfaces interface-id trunk**.
2. When that is done, ensure that the trunk modes on both ends are correct, if so, the trunk port should be operational. You can review trunk settings with **show interfaces interface-id trunk**.

Some problems most people run into when working with VLANs would be native VLAN mismatches, trunk mode mismatches and insufficient or unnecessary VLANs in the list of allowed VLANs. For example, both ends of two switches have the same native VLAN; however, they're both set to Dynamic Auto. Two Dynamic autos result in an Access link, which is not the type of link you want for trunk ports.

Another problem would be an incorrect VLAN list. This can be configured though the **switchport mode trunk allowed** command. Remember that the VLANs that need to be allowed would be the native VLAN and any VLANs that would need to communicate through that trunk link.

VLAN Attacks

With new features always comes the possibility of attacks against a modern network. VLAN based infrastructure simplifies networks and improves overall performance, however, also opens the chance for new attacks. There are some attacks you should know about, and how to approach and mitigate them.

VLAN hopping allows traffic from one VLAN to be seen by others. Switch spoofing is a type of VLAN hopping attack that works by taking advantage of trunk ports that were not configured correctly. By default, trunk ports have access to all VLANs and pass traffic for multiple VLANs across the same link between switches.

In a basic spoofing attack, the attacker will take advantage of the fact that the default configuration of the switch port is dynamic auto, resulting in the ability to spoof as a switch, configuring yourself as a trunk or dynamic desirable, leading to the ability to receive DTP messages. This can allow the attacker the ability to access all VLANs on the port. To mitigate this attack, disable trunking on *all* ports except for those that need it, and manually enable trunking.

Double-Tagging attacks is a type of attack that takes advantage of switch hardware. The attacker is on, let's say, VLAN 10, they tag a frame for VLAN 10 and insert an additional tag for VLAN 20. Most switches only perform one level of 802.1Q de-capsulation, meaning the attacker can insert an additional frame because native traffic is *not* retagged, this will allow the frame to go to a VLAN that the original 802.1Q tag did not specify.

The switch will see the first tag, take it off, keep the second tag (VLAN 20) and forward the frame to the next switch. The second switch will see the tag and proceed to forward it. The best way to approach this attack is to ensure that the native VLAN of the trunk ports is different from the VLAN of any other port. It's best to use a fixed native VLAN that are distinct from all user VLANs in a switched network.

PVLAN Edge, known as Private VLAN Edge, is used to protect ports to ensure no exchange of unicast, broadcast or multicast traffic between ports on the switch. Some applications require that no traffic be forwarded between ports on the same switch so that one host does not see the others traffic. This is where PVLAN edge comes in. Once again, *PVLAN can be used to prevent users from sending each other traffic*. Here are more details about the PVLAN feature:

- Protects the port so it does not forward any traffic (unicast, multicast, or broadcast) to any other port that is also a protected port, except for control traffic. This results in no exchange of data traffic between hosts.
- Forwarding behavior between a protected port and nonprotected proceeds as per usual.
- Protected ports are manually configured.

To configure a port to use the PVLAN Edge feature, use the following command in the configuration mode your interface:

```
S1(config-if)# switchport protected
```

VLAN Design Guidelines

The factory settings that come with switches by default are preconfigured with various settings that would be considered insecure. Some of these settings include default DTP, open ports, a default VLAN, etc. A lot of things can go bad if they're not configured correctly. It's a good practice to change your native VLAN, associate unused ports to a black hole VLAN, etc. You must also separate user traffic from management traffic, in other words, create a VLAN solely dedicated to use for SSH.

Ensure that you've followed all tips and whatnot for designing your VLAN, this would include the implementation of port security, voice traffic configuration if you will use VoIP and Cisco IP phones, etc.

You can prohibit certain VLANs on certain trunk interfaces like so:

```
S1(config-if)# switchport trunk allowed vlan remove vlan-id
```

The biggest step for all of this, is to ensure that you've practiced everything. Please ensure that you practice.

RSE 4 – ROUTING CONCEPTS

Preface

We're done talking about switches for now. The next subject is going to be routers. There are various things about routers we can learn, in this section, some of these things include:

- Functions and features of routers
- How to set up a small routed network
- Using the CLI and configuring basic settings between two networks
- Verify connections between two networks
- Explain encapsulation and decapsulation between networks on a layer 3 level
- Explain path determination for a router
- Explain routing table entries for directly connected networks
- Explain how a router builds the routing table through directly connected networks, static routes and dynamic routing

When IP addresses (source and destination) are involved, this stuff becomes level 3. The job of a router is to deliver packets across a multitude of networks over the internet. The router uses a routing table to determine the best path to forward a packet. The effectiveness of the network depends on the router's ability to choose the best paths.

When a host sends a packet to a device on different networks, the packet is forwarded to the default gateway (the router), because the host cannot just send the packet right away to the destination, it must go through the gateway. The default gateway is where traffic is routed.

Network and Router Characteristics

When you begin building networks, they need to be able to work with everything you're doing. From VoIP, video, gaming, web browsing, your networks handle a lot of stress, *especially when it's a heavily populated network*. There are many things that play a very important role in the design of your network.

- **Topology** – The topology can be both physical and logical. The topology in the first place, is how everything is organized. Where are the cables? How is the network set up? The topology needs to be designed well physically and must work logically.
- **Speed** – Speed is measured in bits per second. We work with B/s, MB/s, GB/s, KB/s and many others.
- **Cost** – This is how much will be spent on components and other important features.
- **Security** – The network must have security and security practices must evolve as the world evolves.
- **Availability** – Availability is simple. Will the network resources be available anytime?
- **Scalability** – This determines how much growth can a network handle? Failure to account for this may result in problems in the future, such as growth being too expensive.
- **Reliability** – This indicates the uptime of the network; will routers, switches and servers be able to service all the time, or will there be times where it's down? It's measured in mean time between failures, or, MTBF.

So what's the purpose of a router? A router is there to point packets in the direction they must go. They determine the best, fastest, and most reliable path a packet can take to reach its destination. The router checks into its routing table to see the best path first, this may seem like an easy task, but when a router is working with hundreds of requests in seconds, they need some serious processing power. The destination could be as short as a couple miles away and as far as another continent.

Routers are like any other device, like switches. They have a CPU, OS, and Memory and Storage like RAM, ROM, NVRAM, Flash, and hard drives. A router is a special type of computer to execute various tasks in fast, real time. Routers usually operate their own version of the Cisco Internetwork Operating System. Here are some important details:

- **Random Access Memory (RAM)** – We know what RAM is, its temporary memory. This temporary memory for routers will be used to store the running IOS, running configuration, IP routing tables, Ethernet ARP tables. It's also used to keep the packets in buffers. RAM is *volatile* memory because the moment the power is gone, it loses what it had.
- **Read-Only Memory (ROM)** – This memory provides permanent storage for boot up instructions, basic diagnostic software and a limited IOS in the event of failure. ROM is firmware and *non-volatile*, so the contents are always there.
- **Non-Volatile Random Access Memory (NVRAM)** – This is RAM, but non-volatile, so the moment the switch no longer has power, the contents stay. Things that would stay in the NVRAM would be startup configuration.
- **Flash** – This is a permanent storage for the router. Things such as the latest IOS are copied from the flash into the RAM. Flash is non-volatile.

Routers come with specialized ports and other interfaces to connect to items, for example, it may come with flash memory cards, auxiliary ports, console ports, USB ports, etc. Usually a pale blue background behind a USB Icon signifies that it is a console connection.

Most people don't really understand how routers work. When you make a google search, your packet could make 30 hops, or, go through 30 routers. Your router is likely to be connected to a WAN, Wide Area Network, transferring data from router to router, landline to landline, etc. Your packets go a long distance, and for all these things to be so quick and reliable, it requires serious planning.

How do routers connect and send packets so quickly? Routers choose best paths. The primary function of a router is to select the best path to send the traffic, and the way they can do this is by using a routing table. A routing table also includes the currently connected interfaces, when the match is found, the router will encapsulate the packet into a data link frame and sends it to the egress port. Sometimes, a router may receive a packet with a different data link frame, only to send it back out with a different one again.

If you would like to view your current routing table, you can execute the following command:

R1# show ip route

Back to another concept earlier, it's possible for routers to receive a packet encapsulated in one type of data link frame, only to forward it with a different type of data link frame. For example, a router may receive a packet on Ethernet, but most forward the packet out with a different interface, such as Point-to-Point Protocol (PPP). The data depends on the type of interface on the router and the medium it's connected through. Some of those mediums include Ethernet, PPP, Frame Relay, DSL, cable, and wireless connections like 802.11 and Bluetooth.



Note: By default, Cisco Routers will use HDLC as their encapsulation for serial interfaces.

Packet Forwarding Mechanisms

Cisco routers will support three different types of packet-forwarding mechanisms:

- **Process Switching** – An old packet forwarding mechanism. When the packet arrives on the ingress interface, it is forwarded to the Control Plane (which is where the CPU is), this is where the CPU matches the destination address with an entry in the routing table, and then proceeds to send it to the egress interface. This process is done with *all* packets, so, you can tell that this can be considered a slow and steady process, hardly implemented. Here is a model of how it works.



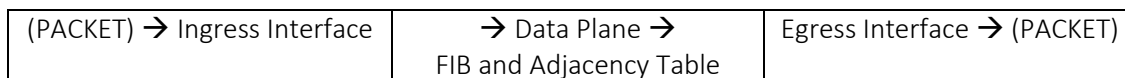
- **Fast Switching** – This is a more common method of packet forwarding. When the packet arrives in the ingress interface, it is forwarded to the Control Plane where the CPU searches for a match in the fast-switching cache. If it does not exist within the fast-switching cache, it is process-switched and forwarded to the egress interface. The flow of information for the packet is stored in the fast-switching cache, so, if a packet is going to the same destination, the packet is immediately sent to the egress interface without CPU intervention. Here is a model if the packet is *not* in the fast-forward cache.



Here is a model if the packet *is* found in the fast-forward cache.



- **Cisco Express Forwarding (CEF)** – CEF is a recently developed Cisco proprietary protocol. CEF builds a similar fast-switching cache like Fast Switching called the Forwarding Information Base (FIB) and an adjacency table. These tables are not packet triggered however. These tables change in the event of a change of the network topology. In this case, the information of the FIB and adjacency tables will be there. The FIB contains pre-computed reverse lookups, next hop information for routes with layer 2 information and interfaces. This is the fastest mechanism and is recommended by Cisco. The model should look like this:



In a more simple explanation, this is how each packet-forwarding mechanism works:

- Process switching figures out where to send the packet by handling each packet, even if others have the same destination.
- Fast switching will check if it knows where to go, if it does not, it will handle the packet slowly, and handle the next ones more quickly in the case that it encounters another with the same destination.
- CEF has every possible path figured out.

The Assembly of a Network

The assembly of a network will require many things, some of these things include the defining of a default gateway, routers, switches, and other items. Network devices and end users are connected through wired or wireless connections. The average office will usually use some wiring and mostly wireless. They usually have a cable model that will connect them to their Internet Service Provider (ISP).

Larger networks will usually have switches and other items involved in the process. Near the edge of the network, they will likely have an edge router. An edge router is a device that sits at the edge of the network, being the separator between a LAN and a WAN.

Very large networks will more than likely host various different servers, switches, etc. They will also host Layer 3 switches. The edge router performs the same duties it original has.

To enable network access, devices must be configured with the following information:

- **IP address** – Identifies unique hosts on local network.
- **Subnet mask** – Identifies which network subnet the host can communicate with.
- **Default gateway** – Identifies the router to send the packets to when the destination is *not* local.

A router is also usually configure with its own default gateway, it is known as the gateway of last resort. When network communication begins, you'll have two different types of communication, *local* and *remote*. Local communication happens on your local network, this will not require the use of a default gateway.

The other type of communication is known as remote communication. Remote communication happens when a packet on a different IP network needs to be transmitted. Direct communication cannot happen, as a result, the default gateway is required. It is the destination that routes traffic from the local network to devices on remote networks.

When designing a network, documentation of devices and many others pieces of information should be invested into. Some of the information you should record are things such as:

- Device names (R1, R2, PC1, PC2, S1, S2)
- Interfaces in design (Fa0/0, Fa0/2)
- IP addresses and subnet masks (192.168.1.1, 192.168.1.2, 255.255.255.0)
- Default Gateway (if Applicable)

It should be placed in a table, sorting out all of these categories. Another idea is creating a topology to provide a visual reference of the network. With designing a network, you need to be able to allocate IP addresses to *all* hosts. This can be done *statically* or *dynamically*. Static is when a host is manually assigned the information needed to function on the network, such as an IP address, subnet mask, and default gateway. This can be used for important things that need a static IP address, such as printers and server.

You can also have IP addresses and other information given dynamically. Information like this can be distributed by the means of DHCP, or, Dynamic Host Configuration Protocol. The DHCP server will provide an IP address, subnet mask, and default gateway. Some devices such as Cisco switches and routers can provide DHCP services.

When it comes to accessing your switches or routers, you will need to do this through SSH, HTTPS clients, or through a console cable. Terminal emulation can be done using the program PuTTY, which is an SSH client.

Within a network infrastructure, you should set IP addresses on switches and other items to allow remote access. A switch does not have a dedicated interface for IP addresses, much rather, you can configure IP address information on a switched virtual interface, or, SVI.

The first step would be going into the interface of a VLAN, which can be done like this:

```
S1(config)# int vlan vlan-id
```

The next step from there would be adding the IP address with both the IP and the subnet mask. Along with that, you should turn on the virtual interface by doing a no shutdown:

```
S1(config-if)# ip add ip-address subnet-mask  
S1(config-if)# no shutdown
```

From there, you will need to go *back* into global configuration mode and configure the default-gateway (router IP):

```
S1(config)# ip default-gateway ip-address
```

Configuration of a Router

Some of the steps taken for Cisco routers and switches are similar in the way of configuration. Some of the things that should be done are naming the device, securing privileged EXEC, user EXEC, and remote access. Another one is the configuration of a banner.

To configure a hostname, you can issue the following command:

```
R1(config)# hostname hostname
```

To configure the banner, you will execute the following command and place the banner within \$ symbols like so:

```
R1(config)# banner motd $ insert text here $
```

Before we configure, I need to explain `enable secret`. `enable secret` is a command that can be enabled over the `enable password` as an additional layer of security, *it puts a layer of security over privileged EXEC mode*. This is how we enable secret:

```
R1(config)# enabled secret password
```

The next step would be configuring the line console. The line console is what a computer connects to whenever you connect to a router via console cable. This is how to secure the line console:

```
R1(config)# line console 0  
R1(config-line)# password password  
R1(config-line)# login
```

 **Note:** Switches use vty 0 to 14. Routers have vty lines 0 to 4. Routers can hold a maximum of five remote sessions.

Configuration of vty lines is the next step. vty lines are configured for remote access. Here is how to secure the vty lines:

```
R1(config)# line vty 0 4  
R1(config-line)# password password  
R1(config-line)# login
```

Once you have configured *all passwords*, you must initiate this next step. These passwords are *not* encrypted until you enable the service password encryption. Execute this command in global configuration mode:

```
R1(config)# service password-encryption
```



Reminder! Don't forget to do a **copy run start** to save all configurations!

One of the most important steps for the process of configuring a router is configuration of interfaces. You are able to configure a router interface with an IP address. The first step of it is to jump into the configuration of the interface. You may choose to add a description with the description command. From there, you can add an IP address and execute the no shutdown command to activate the interface:

```
R1(config)# int g0/0
R1(config-if)# description insert description here
R1(config-if)# ip add ip-address subnet-mask
R1(config-if)# no shutdown
```

Depending on the interface, some additional commands will be required. For example, the serial interface connecting to the serial cable end labeled with DCE must be configured with the clock rate command. For a serial interface, I would recommend that you execute this command, ensure you use the correct clockrate:

```
R1(config-if)# clock rate 128000
```

An IPv6 interface would be configured the same way, the only difference would be the command used to add the IP address. The IP address can be in full format or compressed, regardless, here is an example:

```
R1(config-if)# ipv6 add 2001:db8:acad:1::1/64
```

The most important part of adding IPv6 to your interface is enabling IPv6 in the first place. Recall that the

The **clock rate** command still applies if the interface is serial. The **no shutdown** command still applies because the interface needs to be activated. The **description** command is optional, and the **interface** command is required.

If you are going to implement IPv6, the first thing you should do is configure the following in global configuration:

```
R1(config)# ipv6 unicast-routing
```

The next thing we're going to talk about is the **loopback interface**. The loopback interface is a logical interface that exists on the router. It does not exist physically, so it's similar to interfaces such as the VLAN interface. It's useful for testing devices to ensure that at least one interface works. You can consider it the 127.0.0.1 of computers.

To set a loopback interface, go into the interface configuration for loopback and do the following:

```
R1(config)# interface loopback 0
R1(config-if)# ip add ip-address subnet-mask
```

The loopback interface can also be useful when it comes to the routing interface Open Shortest Path First (OSPF). A loopback interface will mean that the router will use the loopback interface to identify itself instead of an IP address assigned to a port that may not have as much availability. There can be **more** than one loopback interface as long as the interfaces for them are unique and contain unique addresses.

There are many show commands that you can use to verify your interface configurations. Here are three commands you can use to identify any information:

```
R1(config)# sh ip route
```

The **show ip route** command will allow you to view the routing table. The routing table is used by the router to make routing decisions. This table is stored in the RAM. We will look more into the codes and other information later.

```
R1(config)# sh int interface-id
```

This command can be used to show the current configuration of said interface.

```
R1(config)# sh ip int brief
```

This command can be used to give a brief review of *all* interfaces on the router.

You can also verify IPv6 using the following commands:

```
R1(config)# sh ipv6 int brief
```

This can be used to gather all information about IPv6 interfaces.

```
R1(config)# sh ipv6 int interface-id
```

This command can be used to review the interface assigned with an IPv6 address.

```
R1(config)# sh ipv6 route
```

This command can be used to review the IPv6 version of the routing table. An additional command called **ping** can be used to test network connections, you can use the command ping followed by an IP address like so:

```
R1(config)# ping ip-address
```

Expressions and Filters

This next area applies to all sorts of commands, both on Cisco Switches and Routers. You can pipe certain command outputs to receive more tuned results based on your requirements, for example, here are the filtering parameters you can use with commands:

The first one we're going to talk about is Section. Section can be used to show a certain section, for example, executing the following command will result in a specific section being shown in any command's results:

```
R1(config)# command | section section  
R1(config)# show running-config | section line vty
```

In the first line for the command, and how it should look. First part will be the command you plan to work with, followed by the pipe symbol (you can make a pipe by pressing shift and hitting the \ key), followed by **section** and what section you'd like to see. For an example, I did **show running-config**, specifying that I want to see the section with line vty.

The next filter is called Include. The include command will make it so that the output will only print out things into the terminal that command a certain item. Here is an example:

```
R1(config)# command | include string  
R1(config)# sh int brief | include up
```

In the first line, I have how the command should look. It begins with the command, followed by the pipe, ending with **include**. What you want specifically should be put where *string* is. This command would only output things with up in it. This could be useful to find what interfaces are currently up.

Another expression can be exclude. Exclude is used to exclude lines with certain items in it. It works the same way as include, but just to give an example, here you go:

```
R1(config)# command | exclude string  
R1(config)# sh int brief | exclude unassigned
```

The final filtering expression is begin. Begin will show all outputs starting with the line that matches the expression. This could follow the same rule as include and exclude, but here is an example:

```
R1(config)# command | begin string  
R1(config)# sh ip route | begin Gateway
```

This would show us the routing table and skip to the part with Gateway, once a part with Gateway is found, it will ignore everything it went through to get there and print the rest of the output into the terminal.

You have the ability to check all commands you recently executed by doing the show history command. You can also set the history with the following command:

```
R1(config)# show history  
R1(config)# terminal history size amount
```

Routing Switching Functions

Before we continue with this, we just need to understand that the process of routing is to forward packets to their destination. This is done by switching. Switching is literally moving packets from source to destination. The most important part is encapsulating and decapsulating the packets in the correct frames with the right directions. After the router has determined the exit interface, the packet is sent into the data link frame of the egress interface.

A step by step process would essentially be going from Layer 7 (application) to Layer 1 (Physical) with the host on PC1. When the packet is sent, it is sent to the router into the ingress port, going from Layer 1 (Physical) to Layer 3 (Network), once the router has determined the egress port, the router will encapsulate it, making the packet go from Layer 3 (Network) to Layer 1 (Physical), sending the packet to another router. In the event of this happening, it will likely continue to happen, eventually reaching the destination. Of course, the host receives the packet, goes from Layer 1 (Physical) all the way to Layer 7 (Application).

We understand that Layer 3 is involved and so are IP addresses, but to send the packet from one destination to another, we need the MAC addresses, which is why I keep saying “Data Link Layer”. Remember that the Data Link layer works with MAC addresses, this is why interfaces are so important for this. The encapsulation of packets is important because some interfaces may be different, going from FastEthernet, to GigabitEthernet and even to Serial.



Note: Serial interfaces are *not* known by MAC addresses. They are Point-to-Point connections, requiring different Layer 2 frames.

Sending a Packet

The process of sending a packet to a different network will require a couple things. **The first action that must be done is the PC determining its own subnet and network address with the AND operation.** If the ANDing operation results in PC1 belonging to same network, the router is not needed, the switch will be the only required device for transferring the packet. The PC will refer to the Address Resolution Protocol (ARP) Cache. In the event that the MAC address of the host on the *same* network is not in the ARP Cache, the PC will create an ARP request to find the destination MAC address, only to send the packet on its way.

If the ANDing operation says otherwise, the default-gateway (Router) will be required. To determine the MAC address of the default-gateway interface, the PC will check its ARP cache, followed by whether it will send it or be required to send an ARP request. This process is very similar to ARP for IPv6.

Next Hop

To elaborate, a Hop is when a packet goes from one router to another. When Router 1 receives the packet, it will examine the destination MAC address of the ingress interface, copying the frame into its buffer. The router will check for a special item in the Ethernet Type field, specifically 0x800, because this signifies that the packet contains IPv4 data.

The router will initiate the decapsulation process and attempt to prepare the packet for the next hop. Since the destination IPv4 address does *not* match any directly connected networks, Router 1 will search the routing table for a network address that would match the destination address of the packet as a host within said network. Let's assume the routing table has an entry for the 192.168.4.0 network, matching the destination address. Router 1 will proceed to prepare the packet for the next hop. It will do this by re-encapsulating the packet with the MAC address of the next interface the packet will need to go (which is, the Fa0/0 on Router 2), if the router does not know the MAC address of that interface, it will use ARP to find that. From there, the packet is sent where it needs to go.

Reaching the Destination

Router 2 will check its routing table to see if it can send to the next destination, if it can, it will. If the interface is different (for instance, the interface is connected via serial), it will change the data link frame. *Remember that the Data Link frame will be the one to change, **not** the network frame.* Once Router 3 receives the frame, it will check the destination IP address and its ARP cache. If there is nothing in the ARP cache, it will send an ARP request. When someone replies, they will do so. Router 3 adds it to the ARP cache and adds the entry and the packet is encapsulated in a new frame and send to the host.

Destination Found

The host will receive the frame and whatever was in there will end up in the application layer with the end user. It sounds like a good story, but there are some things that you should understand about the process. One of the most important pieces to know is that *the network frame never changes*. The next part is its usually host-to-router, router-to-router, that continues until it goes to router-to-host.

The packet will arrive on the interface, next, the router will check the routing table for a match, if the destination IP address matches a subnet, it will check the directly connected interfaces. If it does indeed match, the ARP cache will be checked, if it does not exist, there will be an ARP request and the packet will be forwarded.

If the destination address does match a nearby subnet (remote network), it will encapsulate the packet and send it on it's way, if not, it will check for a Gateway of Last Resort. From there, the packet will be encapsulated and forwarded out of that interface. If not, the packet will be dropped, there will be three routing decisions in a nutshell:

- **Directly Connected Network** – It will be sent to the host.
- **Remote Network** – The packet will be forwarded to the appropriate interface.
- **No Route Determined** – The Gateway of Last Resort will be taken into consideration, if that does not work, then the packet has hit a dead-end, resulting in it being dropped.

To determine the fastest route, the routers will check hop count versus bandwidth as the unit of measurement, or, metric. Dynamic routing protocols will use their own rules to make super-fast decisions over which way the packet will go, some of these dynamic protocols include:

- **Routing Information Protocol (RIP)** – Based on Hop Count.
- **Open Shortest Path First (OSPF)** – A cisco proprietary protocol, based on bandwidth from source to destination.
- **Enhanced Interior Gateway Routing Protocol (EIGRP)** – Bandwidth, delay, load, and reliability is accounted for.

In the event that two paths to the host are the same, the Router will likely initiate **Equal Cost Load Balancing**. Equal Cost Load Balancing is when a router sends packets through both paths. If configured correctly, load balancing can actually increase the efficiency of networks. It can only be configured with dynamic routing protocols and static routes. The only protocol that supports unequal cost load balancing is EIGRP.

It's possible for routers to be configured with multiple routing protocols and static routes; In this case, the routing table may have more than one route source for the same destination network. In the event that two routing protocols figure two different paths that are deemed "the best", Cisco IOS will take over and figure it out. Cisco IOS uses administrative distance (AD) to determine which path can be trusted the most. In the event that a route is static, it will have an Administrative Distance of 1, resulting in it being used first. EIGRP may have a 5, OSPF may have 110, RIP may have 120, and ultimately, the higher the administrative distance (AD), the less trustworthy a path is.

For reference, here is a chart with default administrative distances.

Route Source	Administrative Distance
Connected (Physically)	0
Static	1
EIGRP summary route	5
External BGP (Border Gateway Protocol)	20
Internal EIGRP	90
IGRP	100
OSPF	110
IS-IS (Intermediate System to Intermediate System)	115
RIP	120
External EIGRP	170
Internal BGP	200

The Routing Table

The routing table exclusively stores two types of routes, **directly connected routes** and **remote routes**. Directly connected routes come from interfaces that are directly connected to the router. Remote routes are connected to other routes, these routes can either be statically configured or dynamically configured.

A routing table is stored in RAM that is used to store routing information about these two types of routes. With this information, the router is told of particular destinations that can be used to send information. If you want to look at the routing table, you can execute the following command:

R1# show ip route

This command will provide various amount of information, including how the route was added, how long the route has been on the table, the specified interface for that destination. Some entries in the routing table can be locally connected interfaces, directly connected interfaces, statically configured routes, or, dynamically learned routes.

Within the routing tables, you will find various codes, some of the common codes are:

- **L** – Identifies the address assigned to the router's interface, allowing the router to determine when it receives a packet for the interface instead of being forwarded from another.
- **C** – Identifies a directly connected network.
- **S** – Identifies a static route.
- **D** – Identifies a dynamically learned network from another router using EIGRP.
- **O** – Identifies a dynamically learned network from another router using OSPF.

D	10.1.1.0/24	[90	/2170112	Via 209.165.200.226,	00:00:05,	Serial0/0/0
---	-------------	-----	----------	----------------------	-----------	-------------

There are certain identifiers within routing entries that must be read correctly. Here is how to read a routing entry by going right within this entry as we go from left to right:

Router Source – Identifies how the destination was learned by the router.

Destination Network – Identifies the address of the remote network.

Administrative Distance – Identifies the trustworthiness of the route source. The lower the value, the more preferred.

Metric – Identifies the value assigned to reach the remote network. The lower the value, the more preferred.

Next-hop – Identifies the Ipv4 address of the next router to forward the packet to.

Router Timestamp – Identifies how long the route has been learned.

Outgoing Interface – Indicates the interface you use to forward the packet out.

You can always review the routing table simply by issuing the command **show ip route** in basic configuration. Before an interface is considered active and added to the IPv4 routing table, the interface must be assigned a valid IP address, be activated with the **no shutdown** command, and receive a signal from another device.

To be able to read a routing table, there are three areas of significance to be identified:

C	192.168.10.0/24 is directly connected,	GigabitEthernet0/0
---	--	--------------------

The first column identifies the router source, or, how the route was learned. For example, an interface could have *C*, which identifies a directly connected network, they may also have *L*, which identifies the IPv4 address assigned to that router's interface.

The second column identifies the destination network and how it is connected, or, the address of the remote network.

The final column indicates the interface on the router connected to the destination, or, the egress interface used to forward the packets to their destination.

Assignment of Addresses to Directly Connected Interfaces

When creating interfaces, there are some things you may want to consider. For instance, you may rename an interface's description, for example:

```
R1(config-if)# description sample text
```

Where I put *sample text* is where the description you would like to implement would go.

Router(config)# ip route network-address subnet-mask { ip-address | exit-intf } If configuring interfaces, you would assign IP addresses with the **ip address** command and in the event that there are IPv6 addresses that need to be added, you may use the **ipv6 address** command to configure one. If you need to view the IPv6 routing table, use the command **show ipv6 route**.

Static Routing

Let's talk about static routing. Static routes are manually specified routes that have been determined by the network administrator. Unlike dynamic routes, static routes are manually configured every time. Some benefits include preference for routing, security, and resource efficiency within the router. They use less bandwidth and almost no work is done by the CPU. Some disadvantages are the lack of reconfiguration in the event that the network topology changes.

There are two kinds of static routes you will encounter within the routing table. The first one is a static route to a specified network, and the second one is a default static route.

A static route can be configured to reach specific network by using the following command:

```
R1(config)# ip route network-mask next-hop-ip or exit-intf
```

Once this has been configured, within the routing table, your entry will be labeled with S for static route. There is another type of static route called a default static route. A default static route specifies the exit point to use when the routing table does not contain a path for the destination network. These are useful in the event that a route has only one exit point to another router, such as when the route connects to a service provider.

To set up the default static route, use the following command:

```
R1(config)# ip route 0.0.0.0 0.0.0.0 {next-hop-ip | exit-intf}
```

It's a good idea to configure a default static route, or, gateway of last resort, you may have to modify the input to get the right results for it. Here is an example for configuring a static route, let's say you need to configure a route to 172.16.3.0/24 using an exit interface/next-hop-ip pair, this pair being S0/0/0 and 172.16.2.1, how would it look like?

```
R1(config)# ip route 172.16.3.0 255.255.255.0 s0/0/0 172.16.2.1
```

IPv6 will work the exact same way, the only difference being the way the commands will be entered:

```
R1(config)# ipv6 route ipv6-address {ipv6-address destination | interface destination}
```

Here are some examples. Let's assume you wanted PC1 to be able to use a static IPv6 route to reach a router with the IP 2001:0DB8:ACAD:3::/64, your PC has the IP address of 2001:0DB8:ACAD:1::/64, this is what you would need to do with the router, issue the following command:

```
R1(config)# ipv6 route 2001:0DB8:ACAD:1::/64 2001:0DB8:ACAD:3::1/64
```

The next scenario would be if we had to do the exact same thing, except, we have to use the interface destination, if so:

```
R1(config)# ipv6 route 2001:0DB8:ACAD:1::/64 s0/0/0
```

Dynamic Routing

We've already discussed the word Dynamic before, which is self-configuring. They spread various bits of information such as reachability, status, network discovery and maintaining routing tables. Network discovery is a very important tool in routing due to the fact that it's the way that information of how to work with the network is spread that way. This is a way for the best path to be found and for a network to fix itself in the event of topology change.

If you want to see the dynamic routing protocols a router may support do the **router ?** command. Cisco ISR routers can support a variety of dynamic IPv4 routing protocols such as:

- EIGRP – Enhanced Interior Gateway Routing Protocol
- OSPF – Open Shortest Path First
- IS-IS – Intermediate System-to-Intermediate System
- RIP – Routing Information Protocol

Routers using dynamic routing protocols automatically share routing information with other routes and compensate for any topology changes without involving manual setup. Our main focus will be EIGRP and OSPF. RIP is discussed as being a legacy protocol.

Within the routing table, there are some examples of dynamic entries within the routing table, such as D*EX meaning it is an EIGRP entry. D means EIGRP, * means it is a default route, and *EX means it is an external route forwarded by EIGRP.

That's just an example, you can also use **ipv6 router** to check out the supported IPv6 routing protocols. Within the IPv6 routing table, it will follow the same concept as IPv4 routing tables. The main concept to understand is that dynamic routing is automatic and does not require help from a certain point.

RSE 5 – INTER-VLAN ROUTING

Foreword

From what we know about VLANs, they're able to allow fixed communication between each other safely and simply. VLANs specifically provide performance, manageability, and security. Trunks can be used to carry multiple VLANs traffic, however, because these VLANs have segmented the network, a Layer 3 process is required to allow traffic to move from one networks segment to another.

Layer 3 routing processes can be introduced into a VLAN network (or you can use a Layer 3 switch), it provides a method of controlling the flow of traffic between network segments and VLANs. This section will focus on implementing inter-VLAN routing. You'll learn how to configure for both the use of a router and a Layer 3 switch, along with troubleshoot.

What Is Inter-VLAN Routing

The purpose of a VLAN is to segment a switched network. Switches work at a very basic level as a Layer 2 device, this means they will lack some functionality, like sending Layer 3 packets. With a large number of VLANs and the lack of dynamic routing, a new way of communication is required. A VLAN itself is a broadcast domain, so computer on a different VLANs will not be able to communicate without a routing device.

I asked myself this, and you may as well, doesn't Inter-VLAN routing defeat the purpose of VLANs? Here's the thing. VLANs break up broadcast domains, assuming you have no VLANs, 100 PCs, 10 PCs per 10 different subnets... Within each network are the PCs with their respective addresses, since these hosts are on different Layer 3 networks they can only communicate with other hosts on the same Layer 3 network, or VLAN.

Legacy Inter-VLAN Routing

PC1 in subnet 1 wants to talk to PC2 in subnet 1, that's okay, it can send an ARP request for the IP address of PC2; every PC in that domain will receive that request and process it. Without VLANs, a hacker can get into PC3 on subnet 4 and start sending broadcasts to all PCs on the network. When you divide the PCs into separate VLANs, or, Broadcast Domains, no longer will that be possible.

Within the beginning of Inter-VLAN, we followed a legacy format for it that was a very long process. PC1 sends a unicast packet to S2 through F0/11(VLAN 10), it's headed to PC3. S2 sends that packet to S1 (F0/1 on S1 is a trunk port), S1 sends the packet to R1 (R1 is a router, meaning it is Layer 3), R1 sends it back to S1 through F0/3 (Layer 10), S1 sends it to S2 through F0/1 (F0/1 is a trunk port), and S2 sends it to PC3 through F0/23 (which is assigned to VLAN 30). This seems like a terribly long process for two hosts connected to the *same switch*.

The Inter-VLAN routing is performed by connecting different physical router interfaces to different physical switch ports. Each are connected to the router are placed in access mode and each physical interface is assigned a different VLAN. This is only being explained to you because it is a legacy concept.

Router-on-a-Stick

While legacy inter-VLAN routing requires multiple physical interfaces on routers and switches, these days, such an approach is no longer required. To be able to have one host on one VLAN communicate with another vice versa, you can use the Router-on-a-Stick method. The router-on-a-stick configuration looks to what you would expect, a router on a stick. First, we need to talk about subinterfaces.

Router can create something called a sub-interface. The sub-interface is a virtual port in a nutshell, you can assign it an IP address and it will be able to handle connections. Instead of having to use two ports on a router in the legacy configuration, you only need one and the job will be so much easier.

Imagine we have PC1 (belonging to VLAN 10) and PC 3 (belonging to VLAN 30). We want these two to talk to each other. What the process is going to roll out is PC1 will send a unicast packet to F0/11 on S1 (VLAN 10), it will be tagged and sent out through F0/1 on S1 (Trunk) and go into F0/1 on S2 (Also a Trunk link), from there it will go into a trunk link that the router has as well.

This is the different part. Router 1 will be connected with a trunk-link, meaning it can carry multiple types of VLAN traffic. So the unicast packet is sent out from F0/3 to G0/0 (Trunk), the router recognizes that it is for VLAN 10 and changes the interface from one sub interface to another. It goes from G0/0.10 (VLAN 10) → G0/0.30 (VLAN 30) simply.

The only problem found with this method is that you cannot exceed 50 VLANs, otherwise, this method is far more efficient than the legacy method.

Multilayer Switch

The Router-on-a-Stick method requires one physical interface on a router and one interface on a switch, however, the use of a dedicated Layer 3 switch can be a way to remove the idea that you need a router to perform one function. This process is likely the most efficient.

It begins with our usual scenario; we have PC1 send a packet to PC3, we know their respective VLANs already, so PC1 sends the unicast packet to S2, S2 sends it to through a trunk link to the Layer 3 switch. The Layer 3 switch, S1, has multiple VLAN interfaces, it notices that the packet is destined for VLAN 30, so it switches it from one VLAN to another (VLAN 10 → VLAN 30) and then proceeds to send the packet back to S2. S2 gets the packet and sends it to PC3.

Discussing this process was much shorter. It's much simpler, you are also not limited to a certain amount of VLANs. Layer 3 switches have additional interfaces compared to a router. With a multilayer switch, packets are routed internally to the switch device, meaning, the packets are not filtered down a single trunk line to obtain new VLAN-tagging information.

Multilayer switches do not completely replace to functionality of routers.

Routers are still extremely significant with various features, however, upgrading from a Layer 2 to Layer 3 switch could help your organization. The configuration of inter-VLAN routing on a switch is specifically static, meaning you will need to configure it to work with the router.

Configuration of Legacy Inter-VLAN Routing

Legacy Inter-VLAN routing will require you to spare 2 interface on the router, limiting communication to only 2 VLANs. The process begins with configuring the switch. You will need to configure 4 ports at a minimum for this to work. Two should belong to one VLAN while 2 belong to another VLAN. Make sure you have 2 PCs with both having one different VLAN. This can be done like so:

```
S1(config)# vlan 10
S1(config-vlan)# vlan 30
```

Putting these two commands will create these VLANs. The next step is to configure the interfaces:

```
S1(config)# int fa0/1
S1(config-if)# switchport access vlan 10
S1(config-if)# int fa0/2
S1(config-if)# switchport access vlan 10
S1(config-if)# int fa0/3
S1(config-if)# switchport access vlan 20
S1(config-if)# int fa0/4
S1(config-if)# switchport access vlan 20
S1(config-if)# end
```

Don't forget you need to save your configuration with a copy run start in basic configuration. The next step for this is configuring the interfaces on the router. **The IP addresses will need to be in conjunction with the VLANs.**

For configuring the router, here is what you will need to do on R1:

```
R1(config)# int g0/0
R1(config-if)# ip add 172.17.10.1 255.255.255.0
R1(config-if)# no shutdown
R1(config-if)# int g0/1
R1(config-if)# ip add 172.17.20.1 255.255.255.0
R1(config-if)# no shutdown
```

From here, you should be able to initiate inter-VLAN communication.

Configuration of Router-on-a-Stick Inter-VLAN Routing

The brightside of doing Inter-VLAN routing with Router-on-a-stick is you will only require one interface by the router, you will have a maximum of 50 sub-interfaces, and you have full functionality of a router by your side. To prepare for Router-on-a-Stick configuration, you must configure the switch as you normally do, and configure the router to become a trunk link with sub-interfaces. Each sub-interface is a software-based virtual interface, assigned to physical interfaces. Each one is configured independently with its own IP address and subnet. It will base the VLAN on the IP address used just like with legacy Inter-VLAN routing and the interface on the router.

To begin, you will need to configure the switch the exact same way we did before, only thing is, we only need 3 interfaces, so one interface for VLAN 10, one interface for VLAN 20, and a trunk port. The configuration should look like this:

```
S1(config)# vlan 10
S1(config-vlan)# vlan 20
S1(config-vlan)# int fa0/1
S1(config-if)# switchport access vlan 10
S1(config-if)# int fa0/2
S1(config-if)# switchport access vlan 20
S1(config-if)# int fa0/3
S1(config-if)# switchport mode trunk
S1(config-if)# end
```

 **Note:** Routers do not support the Dynamic Trunking Protocol, or DTP. That means these commands cannot be used on the trunk port for the switch. **Switchport mode dynamic auto** or **switchport mode dynamic desirable**.

Let's talk about configuring the router now. To begin, we need to create and access a new sub-interface, which is derived from one port, we'll use g0/0, the only difference is that it will be followed by a dot and the VLAN you'll use it for:

```
R1(config)# int g0/0.10
```

Because by default, routers do not support trunking, they must be made a trunk port, issue the following command:

```
R1(config-if)# encapsulation dot1q 10
```

The encapsulation will be the VLAN, which corresponds to the sub-interface. Next, assign the IP address to the interface:

```
R1(config-if)# ip add 172.17.10.1 255.255.255.0
```

From there, you will do the same for the next VLAN:

```
R1(config)# int g0/0.30
R1(config-if)# encapsulation dot1q 30
R1(config-if)# ip add 172.17.30.1 255.255.255.0
```

The final step will be opening the *actual* port (g0/0) like so:

```
R1(config)# int g0/0
R1(config-if)# no shutdown
```

From there, just do **copy run start** and your router-on-a-stick has been configured! On your router, you can use **show vlan** and **show ip route** to verify your sub-interfaces. You can use **ping** or **traceroute** to test connectivity.

Troubleshooting Inter-VLAN Routing

There are some problems within Inter-VLAN routing you may run into. Some of these issues can be misconfigurations the user may have done.

On a legacy configuration, one of the issues could be forgetting to change the VLAN on a particular port, for instance, if there are 2 ports associated with VLAN 30 and one associated with VLAN 10, clearly you're missing a port. This can be solved by doing a **do show vlan** on the switch global configuration.

This could be the same problem on a Router-on-a-Stick configuration, where instead of having the port connecting the switch and router as a trunk port, it has been left as a default VLAN. This can be solved by configuring said port as a trunk port. You could also have an issue where the main interface has *not* been turned on.

Imagine on a Router-on-a-Stick configuration where there is a switch separating the PC1 from the other switch and router, in that event, the connection between must be configured as a trunk link.

There are some steps you can take to ensure that the configuration for a switch is valid, for example, you can issue the following command to check the current specifications for a certain port:

```
S1# show interface fa0/1 switchport
```

This will allow you to view the current configuration for that port. Some important parts would probably be the Access Mode VLAN number and the administrative mode. You should check to ensure that links that are trunk links really are trunking, you can check with the command above along with this one:

```
S1# show run
```

There could also be problems with the configuration of the router sub-interfaces. For instance, ensuring that you've configured the encapsulation correctly is one step. Another step would be ensuring that you've assigned the right IP address to the right interface, remember that the following command must correspond with the sub-interface:

```
R1(config-if)# encapsulation dot1q vLan
```

You can do **show interface** and **show run** to check the encapsulation type for certain interfaces within the router. More problems can arise if the IP address or subnet is configured incorrectly. *VLANs correspond to unique subnets on the network*. For inter-VLAN routing to operate, a router must be connected to all VLANs. Each interface must be assigned an IP address that corresponds to the subnet to which it is connected.

You may verify these things with the **show run** command or the **show ip interface** command. In addition, you can check within the command-line of hosts as to what IP address and subnet mask they have, you will need to issue the **ip config** on the host to check.

Introduction to Layer 3 Switching

So let's talk about the magical technology called Layer 3 Switches. Layer 3 Switches are used in large enterprises. While routers are good at switching packets, Layer 3 Switches have very power hardware, allow millions of packet per second. All catalyst multilayer switches support these types of Layer 3 interfaces:

- **Routed Port** – Pure Layer 3 interface similar to a physical interface on a Cisco IOS router.
- **Switch Virtual Interface (SVI)** – A virtual VLAN interface for inter-VLAN routing.

You can tell for Inter-VLAN routing, we configure and use SVIs for that task. All Layer 3 switches support routing protocols and are extremely advanced apart from their counter parts. Based on what switch you use, a **switchport** or **no switchport** may exist on your switch running config or startup config.

We talked about the switched network design before, the Core, Distribution, and Access layer. Beforehand, switching was fast and routing was slow. This made network designers to extend the switched portion of networks as much as possible. Access, Distribution, and Core were configured to communicate at Layer 2. This topology created some issues, as a result, *spanning-tree technologies were used to prevent to loops while enabling flexibility and redundancy*.

Spanning-Tree Protocol (STP) is a network protocol that builds a logical, loop-free topology for networks. The basic function of STP is to prevent bridge loops and the broadcast radiation that results from them. As network technologies have evolved, routing can be performed at wire speed. This resulted in things such as the Core and Distribution layer having Routing introduced.

Many users are in separated VLANs, with each VLAN in different subnets, as a result, it's logical to configure distribution switches as layer 3 gateways for users of each access switch VLAN; meaning, with all the VLANs, it's best to allow communication by having Layer 3 switches configured at the distribution layer.

Let's talk about SVIs. SVIs are virtual interfaces that are configured within a multilayer switch. They are logical due to the fact that no physical port is truly dedicated to the interface. This not only allows VLAN functionality, but also configuration in much the same way as a router interface (meaning you get cool features like IP addresses, inbound/outbound, ACLs, etc.)

By default, you will have a default SVI created for VLAN 1, also known as the default VLAN. For functionality to work, you must created the additional SVIs for the switch. VLAN numbers correspond to the VLAN tag associated with the data found in the frames, or to the VLAN ID (VID) configured for an access port. When creating an SVI as a gateway for VLAN 10, name the SVI interface and assign it an IP address (for example, 10.1.10.1) would be considered a good starter practice.

Whenever an SVI is created, you need to ensure that the VLAN is within the VLAN database. If the VLAN does not exist within the VLAN database, the SVI will not activate. Here are some reasons SVIs should be configured as soon as possible:

- To provide a gateway for VLAN traffic so that it may go where it needs to go
- To provide Layer 3 IP connectivity to the switch
- To support routing protocols and bridging configurations

The only disadvantage that Multilayer switches have is that they are expensive, but they do have benefits such as lower latencies, not being limited to one link, no need for external links from the switch to the router for routing, and the fact that it is **much** faster than router-on-a-stick, which is why it's used in an enterprise environment.

Routed Ports

Now let's talk about routed ports. A routed port is a physical port that acts similarly to an interface on a router. Unlike an access port, a routed port is not associated with particular VLANs. Because Layer 2 functionality has been removed with routed interfaces, Layer 2 protocols such as Spanning-Tree Protocol (STP) will not function, but some protocols, such as Link Aggregation Control Protocol (LACP) and EtherChannel do function at Layer 3.

Unlike Cisco IOS however, routed ports on a Cisco IOS switch do not support sub interfaces. So Routed ports are used for point-to-point links. Connecting WAN routers and security devices are examples of the use of routed ports. In a switched network, the use of these ports are mostly for configuration between switches in the core and distribution layer.

To configure routed ports, use the **no switchport** interface configuration mode command on said ports. For example, the default configuration of the interface on Catalyst 3560 and up are Layer 2 interfaces that must be configured as routed ports. In addition to configuring them as routed interfaces, you must also assign an IP address, subnet mask, and other Layer 3 details. Once you've finished that, verify that IP routing is globally enabled, and you should be good to go.

Configuring Static Routes on Catalyst 2960

The Catalyst 2960 switch can function as a Layer 3 device and route between VLANs and a limited number of static routes. The Cisco Switch Database Manager. First, we need to talk about what the Cisco Switch Database Manager is. SDM simplifies router deployments, and helps troubleshoot complex network issues. For now, router configuration will be what we need to do.

The command **show sdm prefer** is used to view the current template for the switch. The default template is applied when this command is issued. The default template does *not* support static routing. If IPv6 addressing is allowed, the template will state the ability to use it by showing dual-ipv4-and-ipv6.

If you want to view the global configuration and what configurations you can choose from, issue this command:

```
S1(config)# sdm prefer ?
```

This command will output the templates you can choose from. If you want to use, let's say, lanbase-routing, issue this:

```
S1(config)# sdm prefer lanbase-routing
```

After loading in a new SDM, you will need to reboot your switch with the following command in global configuration:

```
S1(config)# do reload
```

It should be showing out a different output in **show sdm prefer** when you choose to execute that command. Within the output it should tell you that you have much more to work with. Some of the things include an entry that details 'number of IPv4 unicast routes: 0.75K' which means, you have 750 routes you can assign. In a nutshell, issuing the commands will enable IPv4 Routing functionality on your switch, you should be able to assign IP addresses on VLAN interfaces now.

```
S1(config)# interface 0/1
S1(config-if)# switchport access vlan 20
S1(config-if)# interface vlan 10
S1(config-if)# ip add 192.168.10.1 255.255.255.0
S1(config-if)# int vlan 20
S1(config-if)# ip add 192.168.20.1 255.255.255.0
S1(config-if)# no shutdown
```

Once you've issued these commands, the interface fa0/1 should be activated along with the static interfaces. You should issue the **ip routing** command (because switches are not automatically routing by default). If you need IPv6, you will need to issue the **ipv6 unicast-routing** command as it is disabled by default on both Routers and Switches.

From there, if you issue the **do show ip route** command, you should be able to view the routing table within the switch that features the VLANs created within SDM.

Issues that you will encounter while working with Layer 3 switches could range from a variety of things, but here are some categories as to which these problems could exist:

- **VLANs** – VLANs must be defined on *all switches*. Ports must also have the correct VLAN assigned and there must be trunk ports to allow the transfer of VLAN data.
- **SVIs** – SVIs must have the correct IP address and subnet mask. SVIs must be turned on and have a corresponding VLAN number within the IP address.
- **Routing** – Routing must be enabled on switches with the **ip routing** command.
- **Hosts** – The hosts must have the correct IP addresses and subnet masks. The hosts must also have a default gateway associated with an SVI or routed port.

The troubleshooting steps you can take with any other configuration will apply within these scenarios.

Prelude

The big thing about routing is that it is the most important concept within networking. It drives data everywhere it goes. As routers learn more about networks dynamically, with routing protocols, manually, or by using static routes, we understand that routes use plenty of processes to move data. Some routers will use both dynamic routing protocols and static routes.

Static routes are very common and can be far more efficient than dynamic routes. In this chapter, we'll be learning about static routing, how to configure it with IPv4 and IPv6, and how to troubleshoot. We'll also be learning about some extremely important IOS commands and how to analyze the outputs.

We'll also be learning about how to examine and understand the routing table in-depth. Finally, we'll cover Classless Inter-Domain Routing (CIDR) and variable-length subnet masks (VLSM) again. Both of these methods help conserve and optimally use IPv4 address space.

Static Routing Basics

Routes can learn about remote networks in two different ways, manually and dynamically. Manually is when someone manually enters a static route entry into the routing table. Dynamically is when the routes are learned using dynamic routing protocols.

Unlike dynamic routes, static routes will need reconfiguration upon a network topology change. Dynamic will automatically reconfigure itself. So the real question is, why should we use static routing? Static routing does have advantages over dynamic routing while with dynamic routing it is vice versa. Here is a chart to talk about it:

	Dynamic Routing	Static Routing
Configuration Complexity	Generally independent of the network size	Increases with network size
Topology Changes	Automatically adapts to topology change	Administrator intervention requires
Scaling	Suitable for simple and complex topologies	Suitable for simple topologies
Security	Less secure	More secure
Resource Usage	Uses CPU, memory, link bandwidth	No extra resources needed
Predictability	Route depends on the current topology	Route to destination is always the same

Static routes are very useful for smaller network with only one path to an outside network. They also provide security in larger networks for certain types of traffic or other important that need certain types of traffic or links. It is important to understand that you can have both static and dynamic within your network.

When configuring routes, *static routes will **always** have an administrative distance of 1*. There are three uses that most network administrators will take into consideration with static routing:

- Providing ease of routing table maintenance for smaller networks
- Routing to and from stub networks. (A stub network is a network accessed by a single route to and back)
- Using a single default route to represent a path to any network that does not have a more specific match with another route in the routing table.

Within a **Stub Network**, you can simply configure the router to ensure that traffic will only go one way. Simply, static routes can connect to specific networks, stub routes, summarize routing table entries, and create backup routes. They can also provide a gateway of last resort and reduce the number of routes advertised by summarizing several contiguous networks as one static route.

In the event that, let's say, Router 2, is connected to the internet, it can be configured to reach a stub network and a stub network only. That stub network with Router 1, can be configured to know that the only route it can take will be to Router 2, thus, static routes are helpful. This is called a **Standard Static Route**.

A **Default Static Route** is a route that matches for all packets. Typically, a default static route is simply a static route with 0.0.0.0/0 as the destination IPv4 address. If configured like this, it will become a default static route, also known as the Gateway of Last Resort. *These are commonly used with edge routes to connect to an ISP.*



Note: All routes that identify a specific destination with a larger subnet mask takes precedence over the default route.

Default static routes are used when no other routes in the routing table match the packet destination IP address (meaning no match), or, when a router has only one other router to which it is connected, known as a stub router.

There is also something known as a **Summary Static Route**. Simply, if there are multiple networks like, 172.20.0.0/16, 172.21.0.0/16, 172.22.0.0/16, and 172.23.0.0/16, it can be summarized as 172.20.0.0/14. This is to reduce the number of routing table entries. They can be summarized into a single static route if the destination networks are contiguous and can be summarized into a single network address **if** the multiple static routes all use the exact same exit interface or next-hop IP address.

My favorite tool for doing this would exist at <http://asecuritysite.com/ip/routesum>.

The next type of static route is called a **Floating Static Route**. This type of static route is a static route used as a backup path to a primary static or dynamic route in the event of a failure. The only time this route will be used is if the other routes fail.

To accomplish this, the floating static route is configured with a much higher administrative distance, remember about how we said that a static route will always have an administrative distance of 1? Well, we're going to have to configure the route with a high administrative distance.

For example, let's say that we have an Administrator has a floating static route as a backup in case an EIGRP –learned route fails. EIGRP has an administrative distance of 90. If the floating static route is configured with a higher administrative distance than the EIGRP route and that route fails, the floating static route will work.

IP ROUTE Command

We learned about the **ip route** command already, it is used to view at the routing table, however, you can also use this command to add static routes. The command is used in global configuration; here is the syntax:

```
Route(config)# ip route network-address subnet-mask { ip-address | interface-type  
interface-number [ ip-address ] } [ distance ] [ name name ] [ permanent ] [ tag tag ]
```

I should make it very clear that if something is **bolded**, then that means that it is how it's typed, if something is *italicized*, that means that you should put something there. Anyway, let's talk all about this command.

The following parts of the command are required to configure a static route:

- ***network-address*** – This will be the destination address of the remote network being added to the routing table, referred to as the prefix.
- ***subnet-mask*** – Subnet mask, or just mask, of the remote network to be added to the routing table. The subnet mask can be modified to summarize a group of networks.

You can add additional parameters to your command right after.

- ***ip-address*** – The IP address of the connecting router to use to forward the packet to the remote network. This can be also referred to as the next hop.
- ***exit-intf*** – The outgoing interface to use to forward the packet to the next hop.

The command syntax commonly used is

```
Router(config)# ip route network-address subnet-mask { ip-address | exit-intf }
```

If you wanted to create a floating static route, you can use the [***distance***] parameter. Any other parameters are not relevant at this time. Now let's talk about **Next-Hop**.

For next-hop, you could notice that within the routing table, each router has entries only for directly connected network and their associated local addresses. None of the routers have any knowledge of any network that is not theirs, for example, Router 1 wouldn't know that Router 2 has their own LAN set up, or, Router 2 and Router 3 have a serial network between them.

The next-hop can be identified by an IP address, exit interface, or both. How the destination is specified is by one of the three types of routes:

- **Next-hop route** – Only the next-hop IP address exists.
- **Directly connected static route** – Only the router exit interface is specified.
- **Fully specified static route** – The next-hop IP address and exit interface are specified.

You should be able to identify all of these things by using the **show ip route** command.

Next-Hop Static Route

The first type of static route we're going to configure is a **Next-hop Static Route**. In a next-hop static route, only the next-hop IP address is specified. The output interface is derived from the next hop. Before any packet is forwarded by a router, the routing table process must determine the exit interface to use to forward the packet, known as **resolvability process**. This varies based on the type of forwarding mechanism used. CEF (Cisco Express Forwarding) is the default forwarding mechanism used.

Let's make up a scenario in which CEF is not active. First, let's say that we add a couple Next-Hop static routes to the routing table, also, this is how a next-hop static route looks like:

```
R1(config)# ip route 172.16.1.0 255.255.255.0 172.16.2.2
R1(config)# ip route 192.168.1.0 255.255.255.0 172.16.2.2
R1(config)# ip route 192.168.2.0 255.255.255.0 172.16.2.2
```

When a packet is destined for a network, let's say, 192.168.2.0/24, the first thing Router 1 will do is look for a match within the routing table. If the route was set statically, it's going to find this router entry:

```
S    192.168.2.0/24 [1/0] via 172.16.2.2
```

Now, any route that does *not* have an exit interface mentioned must have the next-hop IPv4 address resolved using another route in the routing table with an exit interface. So an exit interface *must* be used. When checking the routing table, there appears to be a connected interface with the following entry:

```
C    172.16.2.0/24 is directly connected, Serial0/0/0
```

Now that Router 1 has determined how to reach the next Router, Router 2, with the IP address 172.16.2.2, it must double check for that match. In this case, it will match with the directly connected network 172.16.2.0/24 with the egress interface Serial0/0/0.

It's actually going to require the use of two routing tables to forward the packet to 192.168.2.0/24. When the router performs multiple lookups before forwarding the packet, it's known as **recursive lookup**. Because these lookups will consume router resources, it should be avoided.

A recursive static route is valid only when the specified next hop resolves either directly or indirectly. CEF provides optimized lookup for efficient packet forwarding by using Forwarding Information Base (copy of routing table) and an adjacency table (includes Layer 2 addressing information). They work together to perform these lookups.

Directly Connected Static Route

In the older version of the Cisco IOS, this method was used to avoid the recursive lookup problem. Directly connected static routes will use an exit interface instead of an IP address to exit. So it really is directly connected, this one might work much simpler than next-hop static routes:

```
R1(config)# ip route 172.16.1.0 255.255.255.0 s0/0/0
R1(config)# ip route 192.168.1.0 255.255.255.0 s0/0/0
R1(config)# ip route 192.168.2.0 255.255.255.0 s0/0/0
```

The routing table may look a bit different compared to the other table within the other example. This is because instead of showing that the IP address is directly connected to a certain next-hop IP address, it will look like this:

```
S      192.168.2.0/24 is directly connected, Serial0/0/0
```

What's going to happen is the router is going to notice the interface that is connected via Serial0/0/0, and use that path to reach its destination. Configuring a directly connected static route with an exit interface allows the routing table to resolve the exit interface with a single search as it is directly connected.

Fully Specified Static Route

The last type of static route is a fully specified static route. This means that both the next-hop and interface are configured. This form of static route is used when the output interface is a multi-access interface and it is necessary to identify the next hop. The next hop must be directly connected to the interface.

```
R1(config)# ip route 172.16.1.0 255.255.255.0 s0/0/0 172.16.2.2
```

The only problem with using a fully specified route is that it may cause unexpected or inconsistent results. The difference between Serial and Ethernet is that with Serial, you expect only one device at the end, the router, while with Ethernet, there will be many different devices sharing the same multi-access network, including hosts and even multiple routes. When you designate the Ethernet exit interface within a static route, the router will not have enough information to determine which device is the next-hop device.

Depending on the topology you are working with and the topology of other routers, you will need to fully specify the static route including both the exit interface and next-hop address.

While you may not like the idea of the router still using CPU resources, a fully specified static route is no longer necessary. A static route using a next-hop address should be used

For verifying your static routes, use the following commands:

```
R1(config)# show ip route
R1(config)# show ip route static
R1(config)# show running-config | section ip route
R1(config)# show ip route network
```


Default Static Route

Now let's talk about default routes, or, the default static routes. A **Default Static Route** is a route that matches all packets. All routes can be stored in a single default route to represent any network that is *not* within the routing table. Routers will use routes that are either directly connected or learned through a routing protocol. A default route is used when no other routes can be used to match the destination IP address, it is also known as the **Gateway of Last Resort**.

Default static routes are commonly used when connecting to an edge router to a service provider, or, a stub router. The common syntax for a default static route will be similar to any other static route, the only difference when configuring it is that the network address is 0.0.0.0 and the subnet mask will be 0.0.0.0. The default syntax will be:

```
R1(config)# ip route 0.0.0.0 0.0.0.0
```

Simply configuring one of these routes will just require you to input the command above with a little more modification. For example, let's say Router 1 has the IP address 172.16.2.1, the router next door is 172.16.2.2, if you want that router to be your Gateway of Last Resort, you would have to configure it like so:

```
R1(config)# ip route 0.0.0.0 0.0.0.0 172.16.2.2
```

Whenever you choose to view the routing table, an asterisk will display next to the S code to indicate that it is a candidate default route, which also means that it is a default gateway of last resort. To truly know whether or not it is, review the address and mask, which should be 0.0.0.0/0.

IPv6 Static Routing

Static routing for IPv6 is extremely similar like most things. For first, here is the syntax for configuring static IPv6 routes:


```
Router(config)# ipv6 route ipv6-prefix/prefix-length { ipv6-address | exit-intf }
```

Parameter	Description
ipv6-prefix	Destination network address of the remote network to be added to the routing table.
prefix-length	Prefix length of the remote network to be added to the routing table.
ipv6-address	Referred to as the next-hop router's IP address, through Ethernet, creates recursive lookup.
exit-intf	Uses outgoing interface to forward packets, referred to as directly attached static route. It is typically used in a point-to-point configuration (serial).

Most of the parameters are almost identical to the IPv4 version of the command. Here are the types of IPv6 Static Routes:

- Standard IPv6 static route
- Default IPv6 static route
- Summary IPv6 static route
- Floating IPv6 static route

As with IPv4, these routes can be configured recursive, directly connected, or fully specified.

 **Reminder!** IPv6 is not on by default. Enable IPv6 routing with the **ipv6 unicast-routing** in global configuration.

Next-Hop IPv6 Static Route

As with IPv4, before any packet is forwarded, the routing table must resolve the route to check if the exit interface is used to forward the packet. This will use CEF once again. The way that IPv6 executes next-hop static routes is exactly the same compared to IPv4. We'll look at an example command and check each detail:

```
R1(config)# ipv6 route 2001:DB8:ACAD:2::/64 2001:DB8:ACAD:4:2
```

When you review the output of this command, we're adding an IPv6 route where if any host would like to communicate with 2001:DB8:ACAD:2, they will have to go through the 2001:DB8:ACAD:4::2 network. We don't need the prefix for the exit interface IP address. A recursive static IPv6 route is valid when the specified next hop resolves either directly or indirectly to a valid exit interface.

Directly Connected IPv6 Static Route

When configuring this type of static route on point-to-point networks, an alternative to using the next-hop IPv6 address is to specify the exit interface. This is an alternative if you choose to use an IOS that does not use CEF, or has it disabled. Let's go ahead and examine the command to see what we can find:

```
R1(config)# ipv6 route 2001:DB8:ACAD:2::/64 s0/0/0
```

So clearly this shows that anyone attempting to reach the 2001:DB8:ACAD:2::/64 network will have to go through s0/0/0. With the use of CEF, the recursive lookup is no longer required, so this is almost just like IPv4 configuration.

Fully Specified IPv6 Static Route

Similar to IPv4, this would be used if CEF was not enabled and the exit interface was on a multi-access network. With CEF, a static route using only a next-hop IPv6 address would be preferred even when the exit interface is a multi-access network. Anyway, let's analyze the command.

```
R1(config)# ipv6 route 2001:DB8:ACAD:2::/64 s0/0/0 FE80::2
```

In a nutshell, anyone attempting to connect to 2001:DB8:ACAD:2::/64 will need to go through FE80::2, specifically their interface s0/0/0. The thing is though, with IPv6, there actually is a situation when a fully specified static route must be used. If the IPv6 static route uses an IPv6 link-local address as the next-hop address, a fully specified static route including the exit interface *must* be used.

The reason why we need to use a fully specified address for that is because IPv6 link-local addresses are not contained within the IPv6 routing table. They are unique on a given link or network. A next-hop link-local address may be valid on multiple networks connected to a router, otherwise, IPv6 does have an extra reason why you should use this method.

To verify these routes, you can use the following commands:

- **show ipv6 route**
- **show ipv6 route static**
- **show ipv6 route network**
- **show running-config | section ipv6 route**

Default IPv6 Static Route

So the default route is a static route that matches all packets, yada yada. We should understand this by now that this is the Gateway of Last Resort and whatnot, if you'd like to configure a Gateway of Last Resort on IPv6, use the following:

```
R1(config)# ipv6 route ::/0 { ipv6-address | exit-intf }
```

Of course you should set up a default static route even though, the basic syntax is highlighted above, you can see the use of it within the next line:

```
R1(config)# ipv6 route ::/0 2001:DB8:ACAD:4::2
```

If we get any packets that don't match our routing table, it's going to be sent to 2001:DB8:ACAD:4::2 simply. You can verify your default static route, or, Gateway of Last Resort, using **show ipv6 route static**.

Classful Network Addressing

The information in this section has already been learned to a certain extent, so I'll just put whatever I feel is important. For example, I feel that this is extremely important. RFC 790 and RFC 791 were released in 1981, describing how IPv4 network addresses were initially allocated based on a classification system. The system would provide three different sizes for networks large, medium, and small.

Class	High Order Bits	Start	End
Class A	0xxxxxxx	0.0.0.0	127.255.255.255
Class B	10xxxxxx	128.0.0.0	191.255.255.255
Class C	110xxxxx	192.0.0.0	223.255.255.255
Class D (Multicast)	1110xxxx	224.0.0.0	239.255.255.255
Class E (Reserved)	1111xxxx	240.0.0.0	255.255.255.255

If you can notice, the high order bits can assist you in remembering where and when does the address begin and end. There is a start and end for those.

Class A is the highest addresses, where 0.0.0.0 means all addresses and 127.0.0.0 is reserved for loopback testing. Class B is intended for medium-large organizations. And Class C is intended for small-medium organizations.

The remaining addresses are for multicasting and future use. Within RFC 790, each network class had a default subnet mask associated. For example, Class A had 255.0.0.0, which meant the first portion of the address was the *network portion*, which determined that it was for a certain network. The 0s indicate the *host portion*, which are for unique to hosts and allow identification, Each A class address would have the potential of over 16 million individual host addresses, while C class addresses were built for a maximum of 254 hosts per network.

With classful IP addressing, this meant that subnet masks of a network address could be determined by the value of the first octet, or, the first three bits of the address. Routing protocols like RIPv1 only needed to propagate the network address of known routes and did not need to include the subnet mask. This is because the router receiving the routing update determined the subnet mask by examining the value of the first octet.

The problem with this type of address usage is that it was extremely wasteful. Organizations were assigned any one of these addresses, and they never knew that the amount of addresses needed would expand far more than what was available.

Classless Inter-Domain Routing (CIDR)

This is where Classless Inter-Domain Routing came into play. The Internet Engineering Task Force in 1993 had introduced RFC 1517. CIDR, or, Classless Inter-Domain routing, replaced classful network assignments and allowed network addresses to no longer be determined by the value of the network address. Instead, it was known by a prefix length, or network prefix. (i.e. /8, /16, /24, etc...) **Router updates contain the address and subnet mask.**

ISPs were no longer limited to these subnet masks, they were able to efficiently allocate address space with these prefixes from /8 to larger. CIDR also reduces the size of routing tables and manages IPv4 address space using **Route Summarization, also known as Prefix Aggregation**, where a route is summarized into a single route, or **Supernetting**, where the route summarization mask is a smaller value than the default traditional classful mask.



Note: A supernet is always a route summary, but a router summary is not always a supernet.

With CIDR and Route Summarization, the internet became far more efficient when an ISP chose to split their IP address into multiple addresses, only to list those IP addresses into even more. It sounds complicated at first, but being able to split addresses for many different hosts, summarizing it into one address is far more efficient than the original system developed. Determining the summary route and subnet mask for a group of networks can be done as the following:

1. List all networks in binary format.
2. Count the number of far left matching bits, identifying the prefix length or subnet mask for the summarized route.
3. Copy the matching bits and then add zero bits to the rest of the address to determine the summarized address.

Let's make an example, assume we have a bunch of B class addresses that need to be represented by Router 1 in static. Now, we could do **ip route 172.16.0.0 255.255.0.0 s0/0/0**, **ip route 172.17.0.0 255.255.0.0 s0/0/0**, **ip route 172.18.0.0 255.255.0.0 s0/0/0**, and so on and so on. Or, you can use **ip route 172.16.0.0 255.248.0.0 s/0/0/0** and be done with it.

How did I do that? Assuming the addresses that needed to be summarized were 172.16.0.0/16 – 172.21.0.0/16, I could summarize all of these routes with 172.16.0.0 255.255.248.0, my favorite tool for route summarization is <http://asecuritysite.com/ip/routesum>.

Let's talk about updating these addresses with Routing Protocols. Classful Routing protocols *cannot* send supernet routes. With the use of VLSM and supernets will require classless routing protocols like RIPv2, OSPF, or EIGRP. Classless routing protocols advertise network addresses with their associated subnet masks. With classless routing protocols, routers can summarize networks and advertise them as a supernet summary static route.



Reminder! When a supernet route is in a routing table, for example, as a static route, a classful routing protocol will not add it into the updates.

Fixed-Length Subnet Masking (FLSM)

Fixed-Length Subnet Masking (FLSM), the same number of addresses is allocated for each subnet. This means that we can split a large address into smaller addresses, for example, we can make 192.168.1.0/24 into eight subnets, each of them with the /27 prefix. How we do this is because if we borrow three bits from the host portion, five will be left for hosts.

Simply using two different equations can be used to figure out the amount of subnets and hosts per subnets:

2^N = Amount of Subnets (N is the amount of 1s that have been borrowed from the host portion.)

$2^H - 2$ = Amount of Hosts (H is the amount of 0s in subnet mask, subtract 2 to count the network and broadcast address.)

If we borrowed three bits, we would gain eight subnets and have 32 addresses per subnet and 30 usable hosts per subnet, it would equal 256 addresses if we combined all eight subnets with all addresses, 254 usable hosts if we subtracted the network and broadcast address, but these larger numbers are only if we didn't subnet them.

Variable-Length Subnet Masking (VLSM)

Variable-Length Subnet Masking is a different type of length subnet masking. The reason I say this is because instead of having to work with the same amount of hosts for each subnet, we can make whatever we'd like with the amount of users we had. Imagine if we had a network 192.168.1.0/24, we could split it in half and have 192.168.1.0/26 and 192.168.1.128/26. But could we split up one of these and have one network with /26 and two networks with /27? It sounds theoretical, but it can be done. Variable-Length Subnet Masking allows you to split subnets into subnets.

In action, we could split up multiple networks continuously until a certain point for *all* networks, this can easily be known as subnetting subnets. A good example for subnet would be to have four /27 networks and three /30 networks. You could incorporate the /27 networks to hold their own hosts while the /30 networks can establish the links between each route, because, if you choose to view how many hosts a /30 subnet has, you can see that it will have four addresses and two usable hosts, which is perfect for the links between the routers.

If we chose to see how many hosts we would have in a /27 network, we can tell that 32 addresses and 30 users per subnet. With this, it leaves three unused /27 subnets and five unused /30 subnets. Putting all these subnets back together will give us 256 addresses and 254 usable hosts.

Check out Page 28 to learn more about VLSM.

Route Summarization

Route summarization, also known as route aggregation, is the process of advertising a continuous set of addresses as one single address with a less-specific shorter subnet mask. The use of CIDR is a form of route summarization and is synonymous with the term supernetting. CIDR will ignore the classful boundaries as it is based around a classless system. This type of summarization will allow a simpler routing table and reduce the bandwidth for utilization of routing updates, resulting in faster router lookups.

To calculate a summary route, it can be done with these steps:

1. List the number of networks in binary format. Here we have some networks listed:

172.20.0.0	10101100	00010100	00000000	00000000
172.21.0.0	10101100	00010101	00000000	00000000
172.22.0.0	10101100	00010110	00000000	00000000
172.23.0.0	10101100	00010111	00000000	00000000

2. Count the number of far left matching bits to determine the mask for the summary route, **we have 14 matching bits within** this (highlighted in black above.)
3. Copy the matching bits and then add zero bits to determine the summarize network address. Here, the end result will be 172.20.0.0. Make the prefix the amount of matching bits (14) and here, we have the final summarized route: 172.20.0.0/14!

Multiple static routes can be summarized into a single static route if the destination networks are contiguous and can be summarized into a single network address *or* if multiple static routes all use the same exit interface or next-hop address. Simply, the destination networks have to be able to be summarized or the static interfaces use the same exit or next-hop.

Route Summarization for IPv6 why

IPv6 addresses can be summarized just like IPv4 addresses, the only difference is that it will require additional steps in the process. Multiple static IPv6 routes can be summarized into a single static IPv6 route if the destination networks are contiguous and can be summarized into a single network address *or* the multiple static routes all use the same exit interface or next-hop address.

To calculate the IPv6 summarized address, do the following:

- 1) List the network addresses (prefixes) and identify the parts where the address differ. You will need to expand IPv6 if it is abbreviated.

2001:0BD8:ACAD:0001::/64
2001:0BD8:ACAD:0002::/64
2001:0BD8:ACAD:0003::/64
2001:0BD8:ACAD:0004::/64

- 2) Convert the differing section from hex to binary, the different section here would be the fourth section.
- 3) Count the number of far left matching bits and then add zero bits to determine the network address.
- 4) Convert the binary section back to hex and only count all matching bits. It should be 61, because none of the last three bits in the fourth octet match. Append the prefix to the summary route, it should be 2001:DB8:ACAD::/61.

2001:0BD8:ACAD:0000000000000001::/64
2001:0BD8:ACAD:0000000000000010::/64
2001:0BD8:ACAD:0000000000000011::/64
2001:0BD8:ACAD:0000000000000100::/64

Floating Static Routes

Floating static routes are routes that have an administrative distance that is greater than static and dynamic routes. They are useful in the case where the original route that is normally used is not available. By default, static routes will have an administrative distance of 1, making them a preferable choice over dynamic routes. Dynamic routes will usually have the following administrative distances assigned to them:

- EIGRP = 90
- IGRP = 100
- OSPF = 110
- IS-IS = 115
- RIP = 120

Administrative distance for a static route can be configured so that the route can “float” when not in use, however, is used in the event that the original route configured cannot be used. Floating static routes can be used to provide backup route to multiple interfaces or networks on a router. It is also encapsulation independent, meaning it can be used to forward packets out of any interface.

The importance is that it is affected by convergence time, meaning, in the event that your original route is constantly dropping and re-establishing connections can cause a floating static route to be activated unnecessarily. To configure a floating static route will be similar to assigning any other route, the only problem is that you will add a number at the end of the command that will represent the chosen administrative distance, like so:

```
R1(config)# ip route 0.0.0.0 0.0.0.0 10.10.10.1 95
```

This configuration reads that any packet that does not match the routing table will head out 10.10.10.1, this routing table entry will be activated after entry within the routing table with an administrative distance under 95 failed to operate. If you would like to test a floating static route, set up two different routes, shutdown the interface with the lowest administrative distance and attempt to reach a host, if your floating static route works, it should communicate.

Troubleshooting Static Routes

There may be times when something goes wrong, especially sometimes with static configurations. When there is a change within the network topology, a network administrator is responsible for solving the problem, to find and solve these problems, use the following tools and commands to help test your network:

- ping
- traceroute
- show ip route
- show ip interface brief
- show cdp neighbors detail
- show ip route | begin Gateway

RSE 7 – ROUTING DYNAMICALLY

Introduction

In the previous chapter, we learned about how routes that a router can send packets to can be manually configured through static routing. When networks become large and too large to maintain, this will result in the need of dynamic routing. To begin, dynamic routing is the automatic configuration of routes using data that is gained through protocols.

Dynamic Routing History

The history of dynamic routing can begin in the late 1980s with the creation of Routing Information Protocol, or, RIP. RIP version 1 was released in 1988 using technologies that were derived from ARPANET. As networking became more complex, RIPv1 became obsolete, resulting in the creation of RIPv2. Even if a newer version was created, RIP is not implemented in large networks.

Two advanced routing protocols were created to solve the need for a better routing protocol for large networks. Open Shortest Path First (OSPF) and Intermediate System-to-Intermediate System (IS-IS). Along with that, Cisco created Interior Gateway Routing Protocol (IGRP) and Enhanced IGRP (EIGRP). These protocols were created to be implemented within large networks.

In addition to needing these protocols, a protocol was created to connected different networks and route between them. What modern-day ISPs use is called Border Gateway Protocol (BGP). This protocol is used by ISPs and large networks to exchange routing information. As networks began to develop even more, eventually, IPv6 versions of most modern protocols were created in the 21st century.

The main purpose of these protocols is to allow routers to send and receive routing information to create tables and topologies. A routing protocol is a set of processes and messages between routers to populate the routing table. The main purposes are simply the following:

- Discovery of remote networks.
- Maintaining up-to-date routing information.
- Choosing the best path to destination networks.
- Ability to find a new best path in the event that the original path is no longer available.

Most routing protocols will follow this basis:

- **Data structure** – Meaning that the protocols will create databases to allow seamless flow.
- **Routing protocol messages** – This is when the routing protocols will exchange messages to allow routers to learn routes, exchange information, and maintain accurate information about the network.
- **Algorithms** – This means that routing protocols will use algorithms to find the best paths.

Simply, the role of the routing protocols is to dynamically share information about remote networks and automatically add this information to their own routing tables. The protocols will find the best path, adapt to change, and exchange information with other routes to allow the best flow of information.

Dynamic routing requires less maintenance, however, in exchange, it takes router resources, goes with whatever is provided, and can be considered less efficient in some situations. However, networks will sometimes get so large that it is better to configure a network to work dynamically.

Static VS Dynamic

Before considering the use of dynamic routing, consider that static routing has many benefits. Some of them include:

- Easier use of routing table.
- Routing to and from stub networks with ease.
- A single default route is available.
- Destination is always the same.
- No use of routing algorithms, updates, or no extra resources are used.

Here are some disadvantages:

- Suitable for only small topologies or stub networks.
- Configuration complexity intensifies.
- Manual intervention required to re-route traffic.
- Link-failure will result in non-reroutable traffic.

The use of dynamic routing will remove the complexity of maintaining a network in the event that a large amount of configuration is needed. Dynamic routing can be useful when:

- The network is large in size.
- Routes are automatically adapted in the event where the original route can no longer be used.

While there are some disadvantages:

- Can be even more complex to implement.
- Less secure, will require additional configuration for security.
- Dependent on topology.
- Requires more router resources.

Dynamic Routing Protocol Operation

Ultimately, all routing protocols are designed to learn about remote networks and to quickly adapt in the event of a topology change. This method requires the use of certain protocols to take advantage of their characteristics. The general operation of dynamic routing protocols are usually as followed:

1. The router sends and receives routing messages on its interface.
2. The routing shares routing messages and routing information with other routers using that protocol.
3. Routers exchange routing information to learn about remote networks.
4. When a topology change is detected, the routing protocol can advertise this change to other routers.

Let's talk about **Cold Start**. When a router begins, it knows nothing about the router surrounding it. The only information that it has is located within its saved configuration within the NVRAM. When the router boots, it will apply the configuration. If the IP addressing is configured correctly within the configuration, then the router will successfully configure itself. If it has no original information, the router will collect information from locally connected routers.

The next step of bootup is **Network Discovery**. Network Discovery is the process of a router requesting and exchanging routing updates to learn about any remote routes. This packet is sent to all routers that are connected, and at the same time, the router will gain updates just like those, from that point, the router will check for new network information. Any corrections will be implemented and new routes will be added.

The process, for example, would start with a router adding certain interfaces and networks to their Network Update. This network update information would be sent to various routers, including the network IP address, the interface, and the distance.

From this point onward, routers will have knowledge of directly connected networks and the neighbor routers. The routers will periodically exchange new information to keep up with any topology changes. Distance vector protocols will implement a routing loop prevention technique known as **split horizon**. This prevents information from being sent out the same interface from which it was received. For example, it will not repeat information a router has already gained. After routers within a network have converged, the routers will work with each other to determine the best route available.

Achieving **Convergence** is the one main goal of those who have created a network. Convergence is when all routers have successfully fulfilled their routing tables for calculation of the best path. Because they consistently update based on topology changes, the convergence should be achieved rather quickly and maintained well as long as configurations are correct and the topology does not gain significant changes. As a result, convergence can be achieved with modern protocols such as EIGRP and OSPF.

Routing Protocols

Ultimately, all routing protocols are separated categories that branch off each other like a spanning tree. Protocols will have their characteristics that you can classify them by:

- **Purpose** – Interior Gateway Protocol or Exterior Gateway Protocol.
- **Operation** – Used for Distance Vector routing, link-state routing, or path-vector routing.
- **Behavior** – Is it classful (a legacy protocol) or classless?

There are various types of routing protocols. Here are their characteristics:

- **RIPv1 (legacy)** – IGP, distance vector, classful.
- **RIPv2** – IGP, distance vector, classless.
- **IGRP (legacy)** – IGP, distance vector, classful protocol developed by Cisco.
- **EIGRP** – IGP, distance vector, classless protocol developed by Cisco.
- **OSPF** – IGP, link-state, classless protocol.
- **IS-IS** – IGP, link-state, classless protocol.
- **BGP** – EGP, path-vector, classless protocol.

The classful (legacy) protocols have been deprecated and replaced with classless protocols.

IGP, EGP, and AS

Let's first talk about Autonomous systems (AS). An **Autonomous System** is a collection of routers under control of an organization. An AS is also known as a **Routing Domain**. These types of networks are internal within an ISP or network. You may also consider them WANs, or, Wide Area Networks.

Remember that the entire Internet is made up of Autonomous Systems, or, the Internet is an AS concept, as a result, two routing protocols were created to work with this concept:

Interior Gateway Protocol (IGP) is used for routing *within* an AS. This is also referred to as intra-AS routing. Companies, organizations, and even service providers use an IGP on their internal network. IGPs include protocols such as RIP, EIGRP, OSPF, and IS-IS. So, these protocols rely on intra-AS routing, working within an Autonomous System.

Exterior Gateway Protocol (EGP) is used for routing between AS'. It is also referred to as inter-AS routing. Service providers and large companies may interconnect using an EGP. In other words, to route between Autonomous Systems, you will need to use an Exterior Gateway Protocol, which in the name makes sense.

The **Border Gateway Protocol (BGP)** is the only currently-viable type of Exterior Gateway Protocol, officially used for all internet routing. Due to BGP being the only inter-AS routing protocol, most will refer to EGP as BGP.

Distance Vector Routing Protocols

We've thrown this term around a lot. Distance vector means that routes are characterized using two details; distance and vector. **Distance** is how far, while **Vector** is in what direction, or, what router, will it be reached with. For example:

For Router 1 to reach 172.16.2.0, it is one hop away (the distance). It can be reached through Router 2 (the vector).

The router using a distance vector routing protocol does not have knowledge of the entire path to a destination network. Distance vector protocols essentially use routers as sign posts that point toward their final destination. They consult the routing table by checking the route they will need to take, **how many routers away** the IP address is, etc.

There are four different distance vector protocols within IPv4 Interior Gateway Protocols:

RIPv1 is the deprecated routing protocol that was the one of the first, while **RIPv2** is a much simpler, distance vector routing protocol. **IGRP** was created by Cisco, eventually replaced with **EIGRP**, an advanced distance vector routing protocol created by Cisco. **The protocol uses metric based on delay (packet delivery time), bandwidth, and on the amount of link traffic and the reliability of the link.**

Link-State Routing Protocols

Link-State Routing Protocols, in comparison to Distance Vector Routing Protocols, can be used to create a complete view or topology of the network by gathering information from all of the other routers. How the routing protocol works is at any point when there is a topology change, the router will send an update that one path no longer works to the other routers. The other routers will receive the message and update their database.

An Introduction to Networks and Cisco

Link-state protocols work best in situations where the network design is hierarchical, usually occurring in large networks, when fast convergence of the network is crucial, or, when the administrators have good knowledge of the network where this protocol has been implemented. There are two link-state IPv4 IGP's:

OSPF, which stands for Open Shortest Path First. This protocol is a popular standard based routing protocol. **This protocol uses cost (cumulative link bandwidths) as its metric.** Another protocol called **IS-IS** exists. It stands for Intermediate System to Intermediate System. This protocol is popular with provider networks, simply, the protocol will notify other systems of any changes.

Classful and Classless Routing Protocols

The largest difference between the two protocol types are that classful routing protocols do *not* send subnet masks with their routing updates, while classless routing protocols *need* to send their subnet mask.

The two original protocols developed for networks were originally created for older networks where IP address consumption was not an issue. The lack of a subnet mask would result in no need for variable-length-subnet masks (VLSMs) and classless interdomain routing (CIDR). Classful routing protocols also created problems in the event that they need to connect from a completely different network.

For example, imagine that two routers (router 1 and 3) are separated by router 2. This network, let's say, has implemented VLSM. VLSM, or, Variable Length Subnet Masks, is used to divide subnets. Now, let's assume the links between Router 1 and 2 and 2 and 3 have a different subnet mask (for, let's say, only giving a little bit of IP addresses between the two networks), the network will not be able to communicate due to the fact that the router in the middle (router 2) is using different subnets.

Classless networks come into play since the use of subnetting plays an extremely important role within the convergence of networks. The classless IPv4 routing protocols, (RIPv2, EIGRP, OSPF, and IS-IS) will include all the information to allow communicate and the use of VLSM and CIDR. All IPv6 routing protocols are considered classless because they include the prefix-length within the address. Here is a summary of the routing protocols:

	Distance Vector				Link State	
	RIPv1	RIPv2	IGRP	EIGRP	OSPF	IS-IS
Speed Convergence	Slow	Slow	Slow	Fast	Fast	Fast
Scalability – Size of Network	Small	Small	Large	Large	Large	Large
Use of VLSM	No	Yes	No	Yes	Yes	Yes
Resource Usage	Low	Low	Low	Medium	High	High
Implementation and Maintenance	Simple	Simple	Simple	Complex	Complex	Complex

- **Speed of Convergence** – Speed of convergence defines how quickly the routers in the network topology share routing information and reach a state of consistency. Fast convergence results in the most efficient protocol. Routing loops may occur when inconsistent routing tables cannot update quickly enough.
- **Scalability** – Scalability is how large a network can become when deployed.
- **Use of VLSM** – Simply whether or not the network is classful or classless.
- **Resource usage** – Resource usage indicate how powerful a router must be to support a protocol.
- **Implementation and Maintenance** – This indicate the intensity and requirements for a network administrator to manage and implement the network.

Distance Vector Technologies (RIP and IGRP)

The first routing technology we're going to talk about are **Distance Vector Technologies**. Distance Vector Routing Protocols share updates between neighbors. Neighbors are routers that share a link and are configured to use the same routing protocol. Remember that the router will only be aware of the network addresses of its own interfaces and the remote network addresses that can be reached through checking the other routers, routers using distance vector routing are *not* aware of network topology.

Some distance vector routing protocols will send updates even if there are no topology changes through the network broadcast address. This is considered inefficient because everyone on the network will have to process that update and there are resources consumed to send that update.

At the core of the distance vector protocol is the algorithm that calculates the best paths available to send to neighbor routers. **The algorithm used for the routing protocol** is defined as mechanisms for sending and receiving routing information, calculating the best paths and installing routes in the routing table, and detecting and reacting to topology changes. Simply, it will receive and send update, ensure that the best route is used, and change when the topology is reconfigured. RIP uses the **Bellman-ford algorithm** as its routing algorithm. IGRP and EIGRP use the **Diffusing Update Algorithm (DUAL)**.

So let's talk specifically about **Routing Information Protocol**, or, RIPv1, first. RIPv1 was a first generation routing protocol created and configured for legacy networks. It is still taught due to its use. RIPv1 has the following characteristics:

- Routing updates are broadcasted to 255.255.255.255 every 30 seconds.
- The hop count is used as the metric for path selection.
- A hop count greater than 15 hops is considered too far, meaning, the router would not send the routing update to that next router.

Eventually, RIPv2 was created to surpass RIPv1. RIPv2 introduced classless routing and many other important features, such as the following:

- **Classless Routing** – Support of VLSM and CIDR with the use of subnet masks.
- **Increased Efficiency** – RIPv2 will forward updates to 224.0.0.9 (this is in the Class D range; reserved for multicast groups).
- **Reduced Routing Entries** – It will support manual route summarization on any interface.
- **Security** – It supports authentication mechanisms to secure routing table updates between neighbors.

All RIP updates are encapsulated into UDP port 520. RIP still has the 15 hop limitation with an administrative distance of 120.

Now we can talk about **Interior-Gateway Routing Protocols**, or, IGRP. IGRP was a protocol created by Cisco to abide by the following characteristics:

- Bandwidth, delay, load, and reliability are used to create a composite metric.
- Routing updates are broadcasted by default every 90 seconds.

Eventually, Enhanced IGRP was developed and introduced. Some of the features include the implementation of support for classless routing. It also introduced efficiency, reduced routing updates, and secure message exchange. EIGRP also introduced the following features:

- **Bounded Trigger Updates** – This means that routing updates are no longer sent periodically. Only in the event where the routing table changes occur. This reduces the amount of load the routing protocol places on the network. This can be useful in the event where the need for routing resources is needed.
- **hello keepalive Mechanism** – A small hello message is periodically exchanged to maintain adjacencies with neighboring routers, meaning that the routers keep resource usage to a minimum to ensure convergence.
- **Maintenance of Topology Table** – Maintains all routes received from neighbors (instead of best paths) in the routing table. DUAL can insert backup routes into the EIGRP topology table.
- **Rapid Convergence** – This may be considered the fastest protocol for convergence because it maintains alternate routes, enabling almost instance convergence. Simply meaning, when one route fails, the router will immediately switch to the alternate route.
- **Multiple Network Layer Protocol Support** – EIGRP uses Protocol Dependent Modules (PDM), meaning that it is the only protocol to include support for protocols other than IPv4 and IPv6.

EIGRP can have a set maximum hop count of 255.

RIP Configuration

Even if RIP is no longer used, it is still important to understand how to configure and use RIP in networks. To begin, you will go into global configuration and input the following command:

```
R1(config)# router rip
```

In the event that you want to disable RIP, you can issue the following command:

```
R1(config)# no router rip
```

Now, the first step to configuring RIP is telling the router to run RIP. From there, you will need to configure the router to know which local interfaces it will need to communicate with, along with those locally connected. This is known as **advertising networks**. To enable RIP routing, use the following command:

```
R1(config)# network network-address
```

You will enter each classful network address, this will enable RIP on all interfaces that belong to a specific network. Associated interfaces will now both send and receive RIP updates, and advertise the specified network in RIP routing updates sent to other routes every 30 seconds.

If you would like to view the current configuration for RIP, you will need to issue the following in privileged EXEC mode:

```
R1# show ip protocols
```

From there, you will receive various bits of information about what the current protocols on the router are, one of the protocols that will be configured is RIP.

You can also execute the following command in privileged EXEC mode to view if the RIP routes are configured simply by looking for the entries that begin with R, which will stand for RIP:

```
R1# show ip route
```

To configure RIPv2, begin by going into global configuration and going into RIP configuration with the following command:

```
R1(config)# router rip
R1(config-router)# version 2
```

From there, your router will be configured with RIPv2. The only problem is that you must configure all devices to use RIPv2. You may still be receiving RIPv1 updates while you wait for RIPv2 updates, so; your routers must all be configured.

By default, RIPv2 will automatically summarize routes. To modify this setting, issue the following command in router configuration mode:

```
R1(config-router)# no auto-summary
```

You will need to do this if you want to allow classless routing. Summarization will keep entries from having their appropriate subnets and masks. This is required if you are having problems with RIPv2.

Now let's talk about the configuration of Passive Interfaces. By default, RIP updates are forwarded out all RIP enabled interfaces, however, RIP updates only need to be sent out interfaces connected to other RIP enabled routers. For example, one RIP router will send updates to a route where RIP does not exist on the other side. This is going to result in unneeded updates being sent out to waste bandwidth, resources, and risk the security of the network.

The **passive-interface** router configuration command will allow transmission of router updates through a router interface while still allowing that network to be advertised to other routers. Simply, you can stop routing updates from going through a certain interface by issuing the following command in router configuration mode:

```
R1(config-router)# passive-interface fa0/0
```

You can check to see if the interface is considered passive or not based on issuing the **show ip protocols** command in privileged EXEC mode and examining section where RIP is configured.

You can make *all* interfaces a passive-interface by issuing the following command:

```
R1(config-router)# passive-interface default
```

This can be a good idea in the event where you need to only have certain interfaces as non-passive. You can make certain interfaces non-passive with the following command after issuing the **passive-interface default** command:

```
R1(config-router)# no passive-interface fa0/0
```

Now, let's assume that you have an edge router connected to the internet. In the event of this, RIP needs to know where the default route is. To propagate a default route, the edge router must be configured with the default static route; you can do that with the following command:

```
R1(config)# ip route 0.0.0.0 0.0.0.0 interface network-address
```

Next, you will have to use the following command within the routing configuration:

```
R1(config-router)# default-information originate
```

That command will tell the router to originate default information with the default static route in RIP updates. If you check your routing table, you should be able to find the gateway of last resort configured. *It is important that you do this only with the edge router.*

What about the event where we need to use IPv6? Sure, RIP is pretty much resting in peace, but, that doesn't mean we can't configure IPv6 with it, and that we can. IPv6 must first be enabled on the router with the following command:

```
R1(config)# ipv6 unicast-routing
```

The next step would be to jump to the interface that you plan to configure; from there you should be in interface configuration mode. The next step would be to enable RIP on that certain interface with this command:

```
R1(config-if)# ipv6 rip RIP-AS enable  
R1(config-if)# no shutdown
```

The process of propagating a default route for RIP is similar in IPv6 compared to IPv4, for example, the commands would go in this order:

```
R1(config)# ip route 0::/0 network-address  
R1(config)# ipv6 rip domain-name default-information originate
```

You can examine your IPv6 RIP configuration with the following commands in privileged EXEC mode:

```
R1# show ipv6 protocols  
R1# show ipv6 route  
R1# show ipv6 route rip
```


Link-State Technologies (OSPF and IS-IS)

Link-state routing protocols are built around Edsger Dijkstra's shortest path first (SPF) algorithm. We will go into SPF later, for now, we need to understand what the current link-state routing protocols exist, Open Shortest Path First (OSPF) and Intermediate System-to-Intermediate System (IS-IS).

Link-state routing protocols are known as the more complex routing protocols, but when implemented within a simple network, the basic functionality and configuration is relatively simple.

Dijkstra's Algorithm is the algorithm that the link-state protocols are built around. It's easier to refer to it as the Shortest Path First (SPF) algorithm. The use of this algorithm involves calculating the total cost of each path from source to destination. The path with the shortest cost is used first.

Each router determines its own cost to each destination in the topology. For example, we want to get to a host on Router 4. We are on Router 1. We can take two different paths, path 1 (4 hops) and path 2 (3 hops).

Path 1 has 4 hops, each with the same cost. Each hop within Path 1 has a cost of 5. Path 2 has 3 hops, each with a cost of 10. Which path would be more cost effective? Path 1. Even if Path 1 has 4 hops, its total cost is 20 while Path 2 has a cost of 30.

Now the next question is how does link-state routing work? With link-state routing protocols, a link is an interface on a router. Information about the current state of that link is known as link-states. All routers in an OSPF area will complete the following process:

1. Each router learns about its own links and its own directly connected networks. This is done by checking the current connection state of the interface.
2. Each router must learn about their neighbors by exchanging hello packets.
3. Each router builds a Link-State Packet (LSP) containing the state of each directly connected link. This is done by recording all information each neighbor, including their ID, link type, and bandwidth.
4. Each router floods the LSP to all neighbors. The neighbors store the LSPs received in a database, then, they proceed to flood *their* neighbors with *their* LSPs.
5. Eventually, each router will use the database to construct a complete map of the topology and compute the best path to each network.

In a nutshell, the SPF algorithm will construct a map of the topology to determine the best paths for all networks. OSPF for IPv4 and IPv6 use the same process. Now, we know that each router learns about the directly connected networks, or, links, the interface that is connected becomes a part of the network. Once it has learned the interfaces, IP addresses, subnet masks, and all of the other details, the second step is meeting the neighbor routers.

Routers will need to use a hello to keep the connection between them alive, when the routers have both exchanged a hello, this forms an adjacency. This is going to keep going to ensure that the connection is always alive.

From there, the router will build the link-state packet to send to all other routers, this will assist them in building the topology table. All the routers will flood these LSP packets throughout the area, allowing all the routers form their topology table. Whichever LSP packet is the latest is accepted to allow the most current information.

From there, the best paths and the topology table will be built, creating an SPF tree. Each router in the routing area that uses a link-state database and the SPF algorithm to create an SPF tree. An SPF tree is all of the destinations, paths, and costs compiled together into one table. SPF will calculate the shortest paths to reach each individual network, resulting in the SPF tree. Each router will construct their own SPF tree. To ensure proper routing, the link-state databases used to construct those tree must be identical on *all* routers.

Using the shortest path information determined by the SPF algorithm, it can be added into the routing table and identified as OSPF paths. **Ultimately, the main goal is collecting the *shortest path first!***

Now, what's are the main advantage of using link-state protocols?

- Each router builds its own topology of the network.
- Immediate flooding of LSPs achieves faster convergence.
- LSPs are sent only when there is a topology change.
- Hierarchical design used when implementing in multiple areas.
- Builds the shortest path *FIRST!*

And disadvantages?

- Link-state protocols require more resources compared to distance vector technologies.
- The requirement of bandwidth.

The use of a hierarchical design is important in this case. Modern link-state routing protocols require more memory and CPU usage compared to other protocols, so, the implementation of a hierarchical design is important. In this case, all areas will need to be configured for certain areas. When we begin configuring OSPF in chapter 8, we will learn more about this.

The Routing Table

So let's talk about the routing table again. The routing table was built in the beginning to allow a router to send packets to the best route available. The route with the smallest administrative distance will be picked first. Static routes can also be implemented.

Check [Chapter 4, Routing Concepts](#), I refuse to rewrite a bunch of information I've already put.

There are some terms I have yet to discuss for the routing table. A dynamically built routing table has various amounts of information. Because of the routing table being a hierarchical structure that is used to speed up the process of routing, we will need some terms to describe these routes.


- **Ultimate Route** – Routing table entries that contain either a next-hop IPv4 address or an exit interface. Directly connected, dynamically learned, and local routes are ultimate routes.
- **Level 1 route** – A level 1 route is a route with a subnet mask equal to or less than the classful mask of the network address. The ultimate route can also be a level 1 route. Level 1 routes can also be:
 - **A network route** – A route that has a subnet mask equal to that of the classful mask.
 - **Supernet route** – A route that is a network address with a mask less than the classful mask, like, a summary address.

- **A default route** – A default route is a static route with the address 0.0.0.0/0.
- **Level 1 parent route** – A level 1 parent route is a level 1 network route that has been *subnetted*. A parent route can *never* be an ultimate route. Usually, if one network address says that it has been variably subnetted within the routing table, then, it is a level 1 parent route.
- **Level 2 child routes** – A level 2 child route is a route that is a subnet of a classful network address. Basically, child routes are derived from a parent route, being subnetted. You'll find many of these.

Now, let's talk about the **Route Lookup Process**.

When a packet arrives on a router interface, the router examines the IPv4 header, identifies the destination IPv4 address, and proceeds through the router lookup process. The router will follow this set of rules for picking a route:

1. If the best match is a level 1 ultimate route, then this route is used to forward the packet and be done.
2. If the best match is a level 1 parent route, then it will go to step 3.
3. If there is a match with a level 2 child route, that subnet is used to forward the packet and be done.
4. If there is no match with any of the level 2 child routes, then it will go to step 5.
5. If there is now a lesser match with a level 1 supernet or default route, the router uses that route to forward the packet and be done with it.
6. If there is not a match with any route in the routing table, the router drops the packet.

 **Note:** route referencing only a next-hop IP address and not an exit interface must be resolved to a route with an exit interface, if Cisco Express Forwarding is not being used. Without CEF a recursive lookup is performed on the next-hop IP address until the route is resolved to an exit interface.

Now, learn this term. **The longest match to IP Packet Destinations** is the **best match**.

What is it mean when the router finds the best match in the routing table? The best match will be the longest match. For there to be a match between destination IPv4 address of a packet and a route in the routing table, a minimum number of far left bits *must* match between the destination IP Packet Destination and the route in the routing table, here is an example:

IP Packet Destination	172.16.0.10	10101100.00010000.00000000.00001010
Route 1	172.16.0.0/12	10101100.00010000.00000000.00000000
Route 2	172.16.0.0/18	10101100.00010000.00000000.00000000
Route 3	172.16.0.0/26	10101100.00010000.00000000.00000000

For example, a packet is destined for 172.16.0.10. The router will have three possible routes that match this packet. Of the three routes, 172.16.0.0/26 has the longest match, and thus, will be chosen to forward the packet. **Keep in mind**, to be considered a match, there must be at least the number of matching bits indicated by the subnet mask of the route.

At last, we need to talk about **IPv6**. For IPv6, the components are the exact same. There is no level 1 parent or level 2 child routes. Ultimately, we've already learned most of the skills we've gained from previous

Introduction to Open Shortest Path First

Previously, we had discussed Dynamic routing and link-state protocols. One of the most important ones was **Open Shortest Path First**. It is a link-state routing protocol that was developed as a replacement for the distance vector routing protocol, RIP. Originally, RIP was an acceptable routing protocol, however, RIP relying on hop count as the only metric resulted in it *not* choosing the shortest path first.

OSPF has significant advantages compared to RIP. It offers faster convergence and scales for much larger networks. It is classless and uses the concept of scalability. This chapter will cover the basics of single-area OSPF.

So, we already named them, however, OSPF has specific features that make it a choice protocol for implementation:

- **Classless** – Classless by design, supporting VLSM and CIDR.
- **Efficient** – Routing changes based on triggers.
- **Fast Convergence** – The network will quickly adapt to change.
- **Scalable** – Meaning that it works best with small, large, and hierarchical networks.
- **Secure** – It supports MD5 authentication, meaning, OSPF routers will only accept encrypted routing updates with peers using the same pre-shared password.

Administrative distance (AD) is the trustworthiness of a route source. By default, OSPF has an administrative distance of 110, being beaten by only a few protocols. It stands out as one of the best dynamic protocols. There are **Data Structures** within OSPF that help build the protocol:

Database	Table	Description
Adjacency Table	Neighbor Table	<ul style="list-style-type: none">• List of all neighbor routers to which a router has established bidirectional communication.• This table is unique for each router.• Can be viewed using the show ip ospf neighbor command.
Link-state Database (LSDB)	Topology Table	<ul style="list-style-type: none">• Lists information about all other routers in the network.• The database represents the network topology.• All routers within an area have identical LSDBs.• Can be viewed using the show ip ospf database command.
Forwarding Database	Routing Table	<ul style="list-style-type: none">• Lists of routes generated when an algorithm runs on the link-state database• Each router's information is unique and contains information about how and where to send packets to other routers.• Can be viewed using the show ip route command.

Then, there are important **routing protocol messages**:

- Hello packet
- Database description packet
- Link-state request packet
- Link-state update packet
- Link-state acknowledgement packet

And then, **the algorithm**. The CPU processes the neighbor and topology tables using Dijkstra's SPF algorithm. The SPF algorithm is based on the cumulative cost to reach a destination. The SPF algorithm creates an SPF tree by placing each router at the root of the tree, calculating the shortest path of each node, placing the best routes into the forwarding database.

Link-State Operation

We learned about this previously, but, let's go into detail. To maintain routing information, the tables, the topology, and all of the other important information, OSPF initiates various routing protocol messages to construct everything, we'll start from the beginning:

1. **Establishment of Neighbor Adjacencies (Development of Adjacency Table)** – This is the first step. OSPF routers must recognize all OSPF enabled routers on the network. An OSPF router will send **Hello packets** out all OSPF interfaces to determine if neighbors are present on those links. This develops the **Adjacency table**.
2. **Exchange Link-State Advertisements** – Once all routers have met each other, they exchange **link-state advertisements, or, LSAs**. LSAs contain the state and cost of each directly connected link. The routers flood their LSAs to neighbors. Neighbors receiving these advertisements flood it to other neighbors until all routers have it.
3. **Building the Topology Table** – Once LSAs are received, OSPF routers will build the topology table (**LSDB, or, Link-State Database**) based on their LSAs. The LSDB will hold all information about the topology.
4. **SPF Algorithm** – The SPF Algorithm is executed, thus, creating the SPF tree. **SPF will apply all the best routes into the routing table.**

Now, let's talk about **Single-Area** and **Multiarea OSPF**. To make OSPF more efficient and scalable, OSPF supports hierarchical routing using areas. An OSPF area is a group of routers that share the same link-state info within LSDBs. OSPF can be implemented in two ways.

- **Single-Area OSPF** – All routers are connected in one area called the backbone area (area 0).
- **Multiarea OSPF** – OSPF is implemented using multiple areas in a hierarchal fashion. All areas must connect to the backbone area.

Multiarea OSPF can divide one large autonomous system (AS) into smaller areas for hierarchical routing. Hierarchical routing allows routing to occur between the areas, while many calculations of routing are kept within the area. The thing about having one large OSPF area is every time there is a change in the topology, the router will rerun the SPF algorithm, taking resources away from anything at that time. Therefore, interconnecting routers into **Area Border Routers (ABRs)** can suit your network for these reasons:

- **Smaller Routing tables** – Fewer routing table entries because network addresses can be summarized between areas.
- **Reduced link-state update overhead** – Minimizes processing and memory requirements.
- **Reduced frequency of SPF calculations** – Localizes the impact of a topology change within an area.

For this chapter, we will only focus on Single-Area OSPF.

OSPF Messages

OSPF messages transmitted over an Ethernet link contain this information:

OSPF IPv4 Header Fields	
Data Link Frame Header	Identifies the destination multicast MAC address (01-00-5E-00-00-05 or 01-00-5E-00-00-06) and source MAC address of the sending interface.
IP Packet Header	Identifies the IPv4 protocol field 89 (which indicates OSPF). Also contains the destination multicast address 224.0.0.5 or 224.0.0.6. Finally, contains the source address of the sending interface.
OSPF Packet Header	Contains the type code for OSPF packet type, Router ID, and Area ID.
OSPF Packet Type-Specific Database	Contains different OSPF packet types: 0x01 Hello – Discovers neighbors and builds adjacencies. 0x02 Database Description – (DBD) Checks for database synchronization between routers. 0x03 Link State Request – (LSR) Requests specific link-state records from router-to-router. 0x04 Link State Update – (LSU) Sends specifically requested link-state records. 0x05 Link State Acknowledgement – (LSAck) Acknowledges the other packet types.

From the fourth column, you can identify that there are 5 different packet types that can belong to OSPF

Let's talk specifically about the **Hello packet** first. The Hello packet is used to discover OSPF neighbors for establishing neighbor adjacencies, advertise parameters on which two routers must agree to become neighbors, and elect the **Designated Router (DR)** and **Backup Designated Router (BDR)** on multi-access networks. Point-to-point doesn't need a DR or BDR.

This is what is inside the Hello Packet:

- **Type** – Identifies the type of packet. We've look at this above, a hello back will have a 1.
- **Router ID** – A 32-bit value expressed in dotted decimal notation to identify the original router.
- **Area ID** – From what area the packet came from.
- **Network Mask** – The subnet mask of the sending interface.
- **Hello interval** – The frequency of sent hello packets. By default, it is 10.
- **Router priority** – The default priority of all routers is 1. The higher the value, the more likely it is the DR.
- **Dead interval** – The time in seconds the router will wait to hear from a neighbor before declaring the link is gone. By default, the dead interval is four times the hello interval.
- **Designated Router (DR)** – The router ID of the DR.
- **Backup Designated Router (BDR)** – The router ID of the BDR.
- **List of Neighbors** – List that identifies the router IDs of all adjacent routers.

The hop count for OSPF is unlimited, meaning that never will a route be unreachable. Hello packets are transmitted to multicast address 224.0.0.5 in IPv4 and FF02::5 in IPv6, all OSPF routers will send Hello packets with these times:

- 10 seconds (default on multi-access and point-to-point networks)
- 30 seconds (default on non-broadcast multi-access networks [NBMA], like, Frame Relay)

The dead interval is four times the hello interval, which is 40 seconds, or 120 seconds.

OSPF Operational States

OSPF progresses through several states when attempting to reach full convergence. Here is each area in detail:

Establish Neighbor Adjacencies	Down State	This means that no hello packets are received. When the router sends the hello packet, it will transition into the next state.
	Init State	The hello packets are received from the neighbor, they contain the sending router ID, and will transition into the next state.
	Two-Way State	The DR and BDR are elected. They will now transition into the next state.
Synchronize OSPF Databases	ExStart State	Negotiation of master / slave relationship and DBD packet sequence number. The master initiates the DBD packet exchange.
	Exchange State	Routers exchange DBD packets. If additional information is required, they will go into loading state, otherwise, full state.
	Loading State	LSRs and LSUs are used to gain additional route information. Routes are processed using the SPF algorithm.
Full State	Full State	Convergence is complete.

Let's review the **Establish Neighbor Adjacencies** area. So, when a router goes from the **Down State** to the **Init State**, it initiates a multicast to find other routers running OSPF. Other routers will reply, when the adjacencies list is built, the routers will transition into the **Two-Way State**, electing DR and BDR.

Before we talk about the next part, why is there a need for a DR and BDR? Multiaccess networks can create some problems with OSPF, including:

- **Creation of multiple adjacencies** – Meaning that Ethernet networks could potentially interconnect many OSPF routers over a common link. Creating adjacencies with every router is not needed, resulting in excessive LSAs exchanged between routers on the same network.
- **Extensive flood of LSAs** – Link-state routers flood their LSAs any time OSPF is initialized, or, when there is a change in the topology.

To understand the problem, there is a formula to learn.

For any number of routers (designated as n) on a multicast network, there are $n(n - 1) / 2$ adjacencies.

At first, it might not seem like a lot, but in reality, just adding another router could form in many more adjacencies. And then there's the flooding of LSAs. Let's assume that a link connected to R2 goes down, R2 must send this to the routers it is connected to, Routers 1, 3, 4 and 5. So R2 sends four LSAs to four routers. What do the four routers do? They send four LSAs each. It could turn into quite the mess. Imagine more routers?

The solution to managing the number of adjacencies and the flooding of LSAs on a multicasts network is by electing a DR to be the collector and distributor of LSAs sent and received. When R1 has to send an update, it sends it only to the DR router, and, the DR router will send it to all Routers (including the BDR). Any router that is not a DR or BDR is a DROTHER. A **DROTHER** is a router that is neither the DR nor the BDR.

Now let's talk about the **Synchronization of OSPF Databases**. After the two-way state, routers transition to database synchronization. In the beginning, a master and slave relationship is created between DROTHER routers and the DR router. Once the **ExStart State** has finished, the **Exchange State** begins with the DR sending its LSDB to the DROTHER router. Once the DROTHER router receives it, the DROTHER router will send its own LSDB. From there, the DROTHER router will either go into a **Full State** or request additional information in the **Loading State**.

Introduction to OSPFv2 and Configuration

So now we can learn how to configure OSPFv2. First of all, when attempting to configure an OSPF area, you will have to follow the following command in global configuration mode:

```
R1(config)# router ospf process-id
```

From there, your prompt will change into router configuration mode. You can check all the available commands you can input by doing the following in router configuration mode:

```
R1(config-router)# ?
```

Now, let's talk about **Router IDs**. Every router needs a router ID to be a part of an OSPF domain. Router IDs can be defined by an administrator, or, automatically assigned. The router ID is used to uniquely identify the router and participate in the election of a DR and BDR.

But how does a router know its router ID? Cisco Router IDs are based on the three following:

- The router ID is explicitly configured using the OSPF **router-id rid** command. *rid* is expressed with 32-bits.
- If there is no router ID, the router chooses the highest IPv4 address of any configured loopback interface.
- If no loopback interfaces are configured, then the router chooses the highest active IPv4 address of any of its physical interfaces. This makes it difficult to distinguish addresses.

If the router uses the highest IPv4 address for the router, the interface does not need to be OSPF enabled. This means that the interface address does not need to be included in one of the OSPF **network** commands. The only requirements is that the interface is active and is in the up state. Here is a way of thinking about it.

- Is the router ID explicitly configured?
 - If yes, use that as the router ID.
 - If no, is the IPv4 loopback interface?
 - If yes, use that as the router ID.
 - If no, use the highest active configured IPv4 address.

So how do you configure an OSPF router ID? First of all, you will need to verify the router ID. You can do this with **show ip protocols**, it will show up within the first couple lines.

As specified before, use the **router-id rid** command while in router configuration mode (to enter OSPF configuration mode, do **router ospf ospf-area**) to configure any 32-bit dotted decimal value for the router ID. For any changes to take effect, execute this command in privileged EXEC mode:

```
R1# clear ip ospf process
```


We mentioned before that you could have your OSPF ID as the loopback interface on the router. Configure it like so:

```
R1(config)# interface loopback 0
R1(config-if)# ip address 1.1.1.1 255.255.255.255
```

Let's continue on the configuration of OSPF. The **network** command determines which interface will take part in the routing process for OSPF. Any interface on a router that matches the network address in the **network** command are enabled to send, and receive OSPF packets. Here is the syntax:

network *network-address wildcard-mask area area-id*

The **area area-id** syntax refers to the OSPF area. When configuring single-area OSPF, the **network** command must be configured with the same *area-id* value on all routers. Although any area ID can be used, it's best to use an area ID of 0 with single area OSPF, making it the backbone OSPF in the event of scaling into a larger network.

Now, **let's talk about wildcard masks** real quick. OSPFv2 uses the argument of *network-address wildcard-mask* to enable OSPF on interfaces. OSPF is classless, thus, a wildcard mask is required. When identifying interfaces, the wildcard mask is typically the inverse of the subnet mask.

Within a wildcard mask, we look at bits. It's known as inverse because the 0 matches a corresponding bit value while the 1 bit ignores the corresponding bit value. The best way to calculate the wildcard mask is to subtract the network subnet mask from 255.255.255.255.

There are several ways to identify the interfaces that will be a part of the OSPF routing process. In fact, there are two different ways.

First of all, let's assume you have a router, Router 1. Router 1 has three different links, the first link goes to its actual network, 172.16.1.0/24. The other two networks are 172.16.3.0/30 and 192.168.10.4/30. Clearly, two networks are just connected to other routers, while one of the actual network. Regardless, all networks will be added to the network. First, we need to understand that the area for this will be 10 (this is an example), next, we need to find the wildcard masks based on the subnets:

For networks 172.16.3.0 and 192.168.10.4: $255.255.255.255 - 255.255.255.252 = 0.0.0.3$

For network 172.16.1.0: $255.255.255.255 - 255.255.255.0 = 0.0.0.255$

Now, we have our wildcard masks, this is what we would put into the router:

```
R1(config)# router ospf 10
R1(config-router)# network 172.16.1.0 0.0.0.255 area 0
R1(config-router)# network 172.16.3.0 0.0.0.3 area 0
R1(config-router)# network 192.168.10.4 0.0.0.3 area 0
```

Hopefully that makes sense. If you don't want to go through the trouble of calculating the wildcard mask, you can just input the address you need and the wildcard mask 0.0.0.0

Now, **by default**, OSPF messages are forwarded out all OSPF interfaces. The problem with this is that these messages only need to be sent out interfaces connected to other OSPF routers. This can negatively affect the network like so:

- Inefficient Use of Bandwidth
- Inefficient Use of Resources
- Increased Security Risk

As a result, we need to configure these interfaces as **Passive Interfaces**. A passive interface will prevent transmission of routing messages, but allow that network to be advertised by other routers. Now, when configuring passive interfaces, it is the *interface* that is set as passive, for example:

```
R1(config)# router ospf 10
```

```
R1(config-router)# passive-interface GigabitEthernet 0/0
```

This would make Gi0/0 a passive interface, meaning, no OSPF messages would be forwarded beyond that interface. If you want to check your passive interfaces (and OSPF paths), use the **show ip protocols** command.

OSPF Cost

Now, let's talk about cost. There is one thing that you will need to know, it's that cost is an important decider for OSPF:

Interface Type	Reference Bandwidth in bps	Default Bandwidth in bps	Cost
10 Gigabit Ethernet 10 Gbps	100,000,000 / 10,000,000,000		1
Gigabit Ethernet 1 Gbps	100,000,000 / 1,000,000,000		1
Fast Ethernet 100 Mbps	100,000,000 / 100,000,000		1
Ethernet 10 Bbps	100,000,000 / 10,000,000		10
Serial 1.544 Mbps	100,000,000 / 1,544,000		64
Serial 128 kbps	100,000,000 / 128,000		781
Serial 64 kbps	100,000,000 / 64,000		1562

The cost of an interface is inversely proportionate to the bandwidth of the interface. A higher bandwidth will indicate a lower cost. For the reference for this example, I use 100,000,000, so, 100 Mbps. The formula to calculate OSPF cost is:

$$\text{Cost} = \frac{\text{reference bandwidth}}{\text{interface bandwidth}}$$

Now, **OSPF Accumulates Cost**. This means that the cost of an OSPF route is based on the destination from one router to the other. For your router, you can go into the configuration mode of your router to adjust the auto-cost reference bandwidth like so:

```
R1(config-router)# auto-cost reference-bandwidth Mb/s
```

All interfaces have default bandwidth values assigned. As with reference bandwidth, interface bandwidth values do not actually affect the speed or capacity of the link. Instead, they are used by OSPF to compute the routing metric, so, it's best to have a bandwidth value that reflects the actual speed of the link.

How do you adjust the bandwidth on an interface? Simply go into interface configuration mode and execute this:

```
R1(config-if)# bandwidth 64
```

This command only changes the bandwidth metric used by EIGRP and OSPF.

Of course, you may also manually set the OSPF cost of a certain interface. The advantage of this is the router will not have to calculate the metric when the cost is manually configured. Use the following command to set OSPF cost on the interface of your choice:

```
R1(config-if)# ip ospf cost 15625
```

Verifying OSPF Neighbors

If two routers do not establish adjacency and recognize each other as neighbors, then it causes inaccuracy and incompleteness within Databases. As an administrator, you will need to ensure that all routers have an active link to each other. You can do this with the following command in privileged EXEC mode:

```
R1# show ip ospf neighbor
```

Two routers may not form an adjacency if:

- The subnet masks do not match, causing routers to be on separate networks.
- OSPF Hello or Dead timers do not match.
- OSPF Network Types do not match.
- There is a missing or incorrect OSPF **network** command.

You can verify an OSPF configuration with the **show ip protocols** command.

The command **show ip ospf** will show you very specific OSPF information, including OSPF process ID and router ID.

One of the fastest ways to verify OSPF interface settings is to use the **show ip ospf interface** command. This command will provide very detailed information for each OSPF interface. To get a summary, use **show ip ospf interface brief**.

OSPFv3 Differences

To begin this section, OSPFv3 was created for IPv6. Recall that in IPv6, the network address was known as the prefix while the subnet mask is called the prefix-length. Just like OSPFv2, OSPFv3 exchanges routing information to populate the IPv6 routing table with remote prefixes. OSPFv3 supports both IPv4 and IPv6.

While OSPFv2 runs over the IPv4 layer, it only communicates with the IPv4 layer. OSPFv3 has the same functionality, the only difference is that it uses the IPv6 layer to communicate with IPv6 users. As with all IPv6 routing protocols, OSPFv3 has a separate process from the IPv4 counterpart. **OSPFv2 and OSPFv3 each have separate adjacency tables, OSPF topology tables, and IP tables.**

These are the similarities between OSPFv2 and OSPFv3:

OSPFv2 and OSPFv3	
Link-State	Yes
Routing Algorithm	SPF
Metric	Cost
Areas	Supports the same two-level hierarchy
Packet Types	Same Hello, DBD, LSR, LSU, and LSAck packets
Neighbor Discovery	Transitions through the same states using Hello packets
DR and BDR	Function and election process is the same
Router ID	32-bit router ID: Determined by the same process in both protocols

There are some differences that need to be differentiated:

	OSPFv2	OSPFv3
Advertises	IPv4 networks	IPv6 prefixes
Source Address	IPv4 source address	IPv6 link-local address
Destination Address	Choice of: <ul style="list-style-type: none">• Neighbor IPv4 unicast address• 224.0.0.5 all-OSPF-routers multicast address• 224.0.0.6 DR/BDR multi-cast addresses	Choice of: <ul style="list-style-type: none">• Neighbor IPv6 link-local address• FF02::5 all-OSPFv3-routers multicast address• FF02::6 DR/BDR multicast address
Advertise Networks	Configured using the network router configuration command	Configure using the ipv6 ospf process-id area area-id interface configuration command
IP Unicast Routing	IPv4 unicast routing is enabled by default.	IPv6 unicast forwarding is not enabled by default. The ipv6 unicast-routing global configuration command must be configured.
Authentication	Plain text and MD5	IPv6 authentication

Routers running a dynamic routing protocol like OSPF, exchanges messages between neighboring routers. They will only need to send and receive routing protocol messages with those directly connected. These messages are always sent from the source IPv4 address of the router doing the forwarding. IPv6 link-local addresses were made to fit this situation. It allows a device to communicate with other devices on that same link and only on that subnet. Packets with a source or destination link-local address cannot go past where the packet originated from.

So in a nutshell, only those in the same network can communicate with each other.

Configuration of OSPFv3 and Troubleshooting

The steps for configuration of OSPFv3 are the same as the configuration for OSPv2, with only some subtle differences like all other protocols.

Step 1:	Enable IPv6 unicast routing: ipv6 unicast-routing
Step 2	(OPTIONAL) Configure link-local addressing
Step 3:	Configure a 32-bit router ID in OSPFv3 router configuration mode using the router-id rid command.
Step 4:	Configure optional routing specifics such as adjusting for reference bandwidth.
Step 5:	(OPTIONAL) Configure OSPFv3 interface specific settings. For example, adjust the interface bandwidth.
Step 6:	Enable IPv6 routing by using the ipv6 ospf area command.

Remember that a network can be configured with IPv4 and IPv6 interfaces, this is called **dual-stack**. A dual-stacked network can have OSPFv2 and OSPFv3 enabled at the same time.

Let's talk about each step. The first step will be enabling IPv6 routing. So, simply inputting that command will allow the configuration of IPv6 routing, the most crucial step.

The second step will be configuring a link-local address. If you do the command **show ipv6 interface brief**, you can check if there is a global IPv6 address assigned and that the interface is enabled. Each interface is automatically generated with a link-local address. They are automatically created when an IPv6 global unicast address is assigned to the interface. Global unicast addresses are not required on an interface, but IPv6 link-local addresses are.

Cisco routers will create the link-local address using FE::/10 prefix and the EUI-64 process. EUI-64 is the process of using a 48-bit Ethernet MAC address, inserting FFFE in the middle, and flipping the seventh bit. The process of assigning the link-local address is relatively simple, though, you should keep in mind that you should make it an easily recognizable address:

In addition to the previous statement, a link-local address that is recognizable and easy to remember should be remembered because the link-local address is **only** required for **that** network. Since the two routers create one network between themselves, you can make the link-local addresses as you see fit.

Link-local addresses can be configured manually with the following command in the interface of your choice:

```
R1(config-if)# ipv6 address ip-address link-local
```



Note! A link-local address has a prefix within range FE80 to FEBF, when an address begins with this hexet (16-bit segment), the words **link-local** must follow it.

The next step we need to talk about is configuring the router ID. You can use the **ipv6 router ospf process-id** command in global configuration mode to enter router configuration mode for IPv6. Now, we need to talk about the process that is done when deciding a Router ID:

1. Is the 32-bit router configured explicitly?
 - a. Use that as the router ID, if multiple interfaces are active, use the highest address.
 - b. If not, go to 2.
2. Is a loopback interface enabled with an IPv4 address?
 - a. Use that as the router ID, if multiple interfaces are active, use the highest address.
 - b. If not, go to 3.
3. Is an interface enabled with an IPv4 address?
 - a. Use that as the router ID, if multiple interfaces are active, use the highest address.
 - b. If not, the CLI will express an error message.

Now, a router ID is a 32-bit value that can be literally anything, anything that is a 32-bit value. So, from 1.1.1.1 to 255.255.255.255, any value that is 32-bits can work. When entering router configuration mode, you will need to use a process ID. The process ID will does not have to match other OSPF routers to establish adjacencies with them, much rather, they are used for consistency.

So, once you are in router configuration mode, your router's prompt will change. Before we continue, it is important to realize that while the process-id can be the same as others, **your router ID cannot be the same as other router IDs**. This is a very significant detail. When changing the router ID, this is the command that you will use.

```
R1(config-rtr)# router-id router-id
```

From there, you can configure a new router-id in the event that you receive an error that requires you to manually configure a router ID. You can check your router-ID by using the **show ipv6 protocols** command in privileged EXEC mode. If you wanted to change the router ID, you'd just have to use the **router-id** command once again in router configuration.

The next step is going to be **how to enable OSPFv3 on a router interface**. This question will simply require you to go into the interface configuration mode of the interface, and proceed to input certain commands. Let's say I wanted to make one of my interfaces an OSPFv3 interface. I would need to jump into (let's say, s0/0/0's interface), and proceed to configure OSPF on it. It should look like this:

```
R1(config)# interface interface-id
R1(config-if)# ipv6 ospf 10 area 0
```

This should be the end of configuring OSPFv3 on a router interface. If you would like to double-check if it works, do the following command **show ipv6 ospf interfaces brief** in privileged EXEC mode to verify all information. There is another type of verification to use in the event that you are configuring OSPFv3, which is execute the following command in privileged EXEC mode **show ipv6 ospf**.

In the event that routers are not talking to each other and establishing an adjacency, you may have to look back on what may be the problem. Things that may be wrong may include link-state information not being passed, incomplete link-state databases, inaccurate SPF trees and routing tables. Routers to destinations may not exist, or even be the "best paths".

You can also verify settings by using the **show ipv6 protocols** command. This can show vital OSPFv3 information. The fastest way to verify OSPF interface settings is to use the **show ipv6 ospf** command. If you want to check on all interfaces *briefly* or in full, either do **show ipv6 ospf interface brief** or **show ipv6 ospf interface interface-id**. If you want to check the routing table for IPv6, use **show ipv6 route ospf**.

RSE 9 – ACCESS CONTROL LISTS

Introduction to Access Control Lists

Network security is a skill that requires mastery when it comes to being a network administrator. For this reason, one of the most important tools are Access Control Lists. At first, Access Control Lists are daunting, however, going at it step by step will enforce and develop a mastery for ACLs. ACLs use permit and deny statements to stop certain traffic and allow certain traffic.

So first of all, what is **ACL**? ACL is a series of commands that control what a router will decide to do with packets. It is one of the most commonly used features within Cisco and IOS software. When configured correctly, ACLs can do the following:

- Limit network traffic for increased performance.
- Provide traffic flow control, such as restricting delivery of routing updates.
- Provide a basic level of security for network access. For example, ACLs can allow one host to access a part of the network, and prevent another from access the same area.
- Filter traffic based on traffic type.
- Screen hosts to permit or deny access to network services. ACLs can permit or deny a user access to file types.

In the beginning, there are no ACL configurations on the router. ACLs can be used for selecting types of traffic to be analyzed, forwarded, or processed in other ways. For example, ACLs can be used to classify traffic for priority processing, making guests limited and those with priority in a different situation.

Before we continue, remember **the three P's**, one ACL per protocol, one ACL per direction, and one ACL per interface.

So, let's talk about TCP communication first. ACLs enable administrators to control traffic right? This control can be simple or complex. First, you will need to understand how **TCP conversations** go.

It begins with **TCP SYN**, which is like a "let's talk" message. When a destination accepts this requests, it will reply with **TCP SYN/ACK**, which is a "sure, let's talk" message. From there, a **TCP ACK** is sent, meaning "we have a connection." From there, the client may continue with some data, such as "I am requesting this data", the server will reply with TCP ACK, sending the data. After, the server will send another TCP ACK, saying "I have received my data." After that, the server, will reply with **TCP FIN**, which means that the server is done talking to the client. From there, the client will reply with **TCP FIN/ACK**, meaning that it is finished as well. At the end, the server replies with a thank you, or, TCP ACK.

The order will usually go:

TCP SYN → TCP SYN/ACK → TCP ACK → TCP FIN → TCP FIN/ACK → TCP ACK

That's usually how it will go. In addition and for future reference, here is a port range list:

Port Number Range	Port Group
0 to 1023	Well Known Ports
1024 to 49151	Registered Ports
491512 to 65535	Private and/or Dynamic Ports

An Introduction to Networks and Cisco

So how do ACLs work? How does the ACL use the information during the TCP/IP conversation to filter traffic? Packet filter, or, static packet filtering, controls access to a network by analyzing the incoming and outgoing packets, passing or dropping them based on given criteria, such as the source IP, destination, or the protocol.

A router will act like a packet filter based on predetermined filtering rules. When a packet arrives, the router will extract information from the header and run the information through the filter. **Packet filtering works at Layer 3 and Layer 4.** The router will make its decision based on that information. The router can filter based on source ports and destination ports of the TCP or UDP segments.

I think it would be a good idea to state that an ACL is a sequential list of permit or deny statements, known as **Access Control Entries (ACEs)**. ACEs are also known as ACL statements. ACEs can be made to filter traffic based on set criteria. When network traffic passes through an interface, the router will check each ACE based on the order the ACEs were created in. When a match was found, the packet is processed with either a permit or deny. ACLs can be configured to control access to a network or subnet.

It's just important to know that ACLs extract the following information from Layer 3 and Layer 4:

Layer 3	Layer 4
Source IP address	TCP/UDP source port
Destination IP address	TCP/UDP destination port
ICMP message type	

For example, when the router scans the packet, it can go through a certain analysis. It goes down the ACEs until a certain match is met, the first thing is the network the packet is from, next, where is the packet going? It could work like this:

- If the packet is a TCP SYN from Network A using Port 80, it is allowed to pass. All other access is denied.
- If the packet is a TCP SYN from Network B using Port 80, it is blocked. All other access is permitted.

The Operation of ACLs

Let's talk about how ACLs work. ACLs once again, are a defined set of rules that give added control for packets that enter inbound interfaces, packets that relay through the router, and packets that exit outbound interfaces. ACLs **do no** act on packets that originate from the router itself. So, we know that ACLs are configured to apply to inbound traffic or to apply to outbound traffic, there are two different types of ACLs:

- **Inbound ACLs** – These are incoming packets that are processed before they are routed to the outbound interface. Inbound ACLs are efficient because it saves the resources used on routing lookups. Inbound ACLs are best used to filter packets via an inbound interface as the only source of the packets needed to be examined.
- **Outbound ACLs** – Incoming packets are going through an outbound interface, processed through the outbound ACL. Outbound ACLs are best used when the same filter will be applied to packets coming from multiple inbound interfaces before exiting the same outbound interface.

Ultimately, within a ACL list, the last statement of an ACL is always an implicit deny statement, meaning that if nothing matches (whether permitted or denied, or, there is no permit statement, it will deny all traffic.

There are two different types of ACLs within Cisco IOS. Standard and Extended. A **Standard ACL** can be used to permit or deny traffic only from source IPv4 addresses. The destination and ports are not evaluated. If there was an access list with this entry:

```
access-list 10 permit 192.168.30.0 0.0.0.255
```

Because there is at least one permit statement, any traffic that is not from 192.168.30.0/24 will be denied access. Standard ACLs are created in global configuration mode. **Extended ACLs** are a filter that examine packets based on the following attributes:

- Protocol type
- Source IPv4 address
- Destination IPv4 address
- Source TCP or UDP ports
- Destination TCP or UDP ports
- Optional protocol type information for additional control

So, assuming that we had an access list configured with this:

```
access-list 103 permit tcp 192.168.30.0 0.0.0.255 any eq 80
```

This access list will permit all TCP traffic from 192.168.30.0/24 going through port 80. Anything else will be denied. Extended ACLs are also created in global configuration mode. Later on, you will learn more about both ACLs.

Let's talk about numbering and naming ACLs. For numbering:

Standard IP ACL	1 to 99, 1300 and 1999
Extended IP ACL	100 to 199, and 2000 to 2699

Anyway, regarding named ACLs, you can assign a name to identify the ACL, try following these tips. You can name with numbers, I suggest you write it in capital letters, names cannot contain spaces or punctuation, and entries can be added or deleted within the ACL.

ACL Wildcard Masking

Let's talk about wildcard masking. IPv4 ACEs include the use of wildcard masks. A wildcard mask is a string of 32 binary digits used by the router to determine which bits of the address to examine for a match. IPv6 ACLs do not use wildcard masks, much rather, the prefix-length is used. Anyway, as with subnet masks, the numbers 1 and 0 are used in the wildcard mask to identify how to treat the corresponding IP address bits. However, in a wildcard mask, they follow a different purpose, serving as an inverse mask. Subnet masks will use binary 1s and 0s to identify the network, subnet, and host portion of an IP address. Wildcard masks use binary 1s and 0s to filter individual or groups of IP addresses to permit or deny access. The wildcard mask 0 bit means match the corresponding bit, while 1 means ignore corresponding bit.

	Decimal Address	Binary Address
IP Address to be Processed	192.168.10.0	11000000.10101000.00001010.00000000
Wildcard Mask	0.0.255.255	00000000.00000000.11111111.11111111
Resulting IP Address	192.168.0.0	11000000.10101000.00000000.00000000

An Introduction to Networks and Cisco

If you went ahead and read that example, the IP address that needed to be processed was reviewed with the wildcard mask. The wildcard mask had 0s in the first two octets. At first, you need to remember that 0 means that the bit is matched. The last two octets are filled with 1s, so that means that entire section is ignored due to no matching bits.

Let's look at more difficult example. Here are two examples to **match ranges**.

	Decimal Address	Binary Address
IP Address to be Processed	192.168.16.0	11000000.10101000.00010000.00000000
Wildcard Mask	0.0.15.255	00000000.00000000.00001111.11111111
Results	192.168.16.0	11000000.10101000.00010000.00000000
	to	to
	192.168.31.255	11000000.10101000.00011111.11111111

Take into consideration that this is **match ranges**. Now, for this example, the first two octets and the last octet can be any valid number, as a result, this is a mask that checks for the range of networks 192.168.16.0 to 192.168.31.0.

	Decimal Address	Binary Address
IP Address to be Processed	192.168.1.0	11000000.10101000.00010001.00000000
Wildcard Mask	0.0.15.255	00000000.00000000.11111110.11111111
Results	192.168.1.0	11000000.10101000.00000001.00000000
	All odd numbered subnets in the 192.168.0.0 major network	

This example shows a wildcard mask that matches the first two octets, and the least significant bit in the third octet. The last octet and the first seven bits of the third octet can be any valid number. The result is a mask that would permit or deny all hosts from odd subnets from the 192.168.0.0 major networks. Now, the big question is, how do I calculate a wildcard mask? One shortcut method is to subtract the subnet mask from 255.255.255.255.

As an easy example, let's assume you only wanted users from 192.168.3.0 accessing the network. The subnet mask would be 255.255.255.0. Subtract that from 255.255.255.255, and what do you get? 0.0.0.255. That highlights the host portion.

A better example. Let's say you only want 14 users from the subnet 192.168.3.32/28. The subnet mask for the IP subnet is 255.255.255.240, so, subtract 255.255.255.240 from the wildcard mask. You get 0.0.0.15.

For an example, let's assume that you want to match only networks 192.168.10.0 and 192.168.11.0. Take 255.255.255.0 and subtract the subnet mask, for this situation, would be 255.255.254.0. Your result would be 0.0.1.255. You could accomplish this:

```
access-list 10 permit 192.168.10.0
access-list 10 permit 192.168.11.0
```

With this:

```
access-list 10 permit 192.168.10.0 0.0.1.255
```

Or configuring the network range 192.168.16.0 – 192.168.31.0 to:

```
access-list 10 permit 192.168.16.0 0.0.15.255
```

How does that one make sense? Let's look into it. The network ranges 192.168.16.0 to 192.168.31.0. How many networks? From 16 to 31, 15 networks. So, if you only wanted access to 15 networks, you would subtract 255.255.240.0 from the wildcard mask.

Now let's talk about **Wildcard Bit Mask Keywords**. Working with decimal representations of binary wildcard mask bits can be tedious. To simplify it, the keywords **host** and **any** help identify the most common uses of wildcard masking. These keywords eliminate entering wildcard masks when identifying a specific host or an entire network. These keywords make it easier to read an ACL by easing the need to understand.

For example of a single IP address, instead of **192.168.10.10 0.0.0.0**, you can use **host 192.168.10.10**. If you're trying to deny or allow a certain host, you will use 0.0.0.0 as their wildcard mask.

For making an ACL to match any IP address, instead of entering **0.0.0.0 255.255.255.255**, you can use the word **any**. If you're trying to deny or allow a certain amount of hosts, use a wildcard mask relevant to their subnet mask.

Guidelines for ACL Creation

I know it sounds cheesy, but, this is very important, and can help you. Writing ACLs can be complicated. ACLs do not have to be configured for both directions. Their direction is applied to the interface will depend on the requirements.

Remember these three requirements:

1. One list per interface (e.g., FastEthernet0/0)
2. One ACL per direction (i.e., IN or OUT)
3. One ACL per protocol (e.g., IPv4 or IPv6)

Now, here are some guidelines on setting up ACLs:

- Use ACLs in firewall routers positioned between your internal network and an external network, like the internet.
- Use ACLs on a router positioned between two parts of your network to control traffic from entering or exiting a specific part of your internal network.
- Configure ACLs on border routers, that is, the routers on the edges of the network.
- Configure ACLs for each network protocol configured on the border router interfaces.

Guideline/Advice	Benefit
Base your ACLs on the security policy of the organization.	This will ensure you implement organization security guidelines.
Prepare a description of what you want your ACLs to do.	This will help avoid inadvertently creating potential access problems.
Use a text editor to create, edit, and save ACLs.	This will help you create a library of reusable ACLs.
Test your ACLs on a development network before implementing them on a product network.	This will help you avoid costly errors.
For inbound ACLs, incoming packets are processed before they are sent to the outside interface.	For outbound ACLs, incoming packets are processed after they are sent to the outside interface.

Standard and Extended ACL Placement

Next piece of information is, **where do I place ACLs?** The proper placement of an ACL can make the network operate more efficiently. An ACL can be placed to reduce unnecessary traffic, for instance, traffic that will be denied at a remote destination should not be forwarded using network resources along the route to that destination. Here's the info:

- **Extended ACLs** – Locate extended ACLs as close as possible to the source of traffic to be filtered. This will allow undesirable traffic to be denied close to the source network without crossing the network.
- **Standard ACLs** – Because standard ACLs do not specify destination address, place them as close to the destination as possible. Placing a standard ACL at the source of traffic will prevent that traffic from reaching any other network.

Placement of the ACL and type of ACL may also depend on:

- **The extent of network admin control** – Placement of the ACL can depend on whether or not the network administrator has control of both source and destination networks.
- **Bandwidth of the networks involved** – Filtering unwanted traffic at source prevents transmission of traffic before it consumes bandwidth.
- **Ease of configuration** – If the network administrator wants to deny traffic coming from several networks, one option is used to a single standard ACL closest to the destination. The disadvantage is that traffic from these networks will use bandwidth unnecessarily.

So let's talk about **Standard ACL** placement if you didn't understand the original description. A standard ACL can only filter traffic based on a source address. The basic rule for placement of a standard ACL is to place it as close as possible to the destination network. This will allow all traffic to reach other networks except the network where packets will be filtered. For example, let's say we have Router 2, which is connected to another router (that hosts 192.168.10.0/24.) Now, if it connects to Router 3. Router 3 has S0/0/1 (connected to router 2), G0/0 (connected to 192.168.30.0/24), and G0/1 (connected to 192.168.31.0/24).

If we wanted to block traffic from 192.168.10.0/24 from going into 192.168.30.0/24, where would be the best place to put a Standard ACL? The best place would be G0/0, which links to 192.168.30.0/24. If we placed it on S0/0/1, it would consume bandwidth and stop *all* traffic from 192.168.10.0/24 (which is not what we wanted.) Basically, **you put standard ACLs as close to destination networks as possible.**

Let's talk about **Extended ACLs** now. Like the standard, it can filter based on source address. The only difference is that it can also filter based off destination, protocol, and port number. Now, let's say you wanted to block FTP and Telnet Access from 192.168.30.0/24 to 192.168.11.0/24, where would be the best place to place this filter? This depends on the topology. From the way to look at it, G0/0 would still be the best. This is because the only network affected by this ACL is 192.168.30.0/24.

Standard IPv4 ACL Configurations

Let's talk about what makes ACLs... ACLs. **Criteria Statements**. When traffic first enters the or exits the network, the traffic is compared to all ACEs in the order that occurs within the ACL. The router will process the ACEs until a match is found. If nothing matches, it is dropped, so, no matter what, all ACLs will have the implied deny at the end.

Let's talk about the logic of a standard ACL. If packets are permitted, they are routed through the router to an output interface. If the packets are denied, they are dropped at the incoming interface. It's simple. ACLs will look actively at the criteria set on the ACL.

The command can be set like so:

```
R1(config)# access-list access-list-number { deny | permit | remark } { host | any }  
source [ source-wildcard ][ log ]
```

You can add **host** or **any** to the statement if you want to make the ACL for certain reasons. Now, you can use following command to create an access list that will allow all hosts from 192.168.10.0 to access whatever you're protecting:

```
R1(config)# access-list 10 permit 192.165.10.0 0.0.0.255
```

You could also add something before the ACL above, like this:

```
R1(config)# access-list 10 remark Permit hosts from the 192.168.10.0 LAN
```

So, if you issued this command:

```
R1(config)# show access-lists
```

You could get back:

```
access-list 10 remark Permit hosts from the 192.168.10.0 LAN  
access-list 10 permit 192.165.10.0 0.0.0.255
```

Keep in mind that ACLs follow specific logic called Internal logic. Meaning. Statements that deny all hosts, and then allow certain hosts will not be allowed to work. It will conflict. Though, host statements can always be configured before range statements. And host statements can be configured after range statements if there is no conflict.

So what's the process of configuring an ACL?

After a standard ACL is configured, it must be linked to an interface using the **ip access-group** command:

```
R1(config-if)# ip access-group { access-list-number | access-list-name } { in | out }
```

If you want to remove an ACL from an interface, you need to simply put **"no"** at the beginning. **Remember that you must be in interface configuration mode to use these.** And once you've done everything except add **in** or **out**, add either one of those to configure the filter as an inbound or outbound filter.

You may also name an interface. This must be done when first creating the ACL:

```
R1(config)# ip access-list [standard | extended] name
```

From there, you configure your interface, then, when on the interface that you would like to apply the access list to, issue the following command on the interface you would like to configure:

```
R1(config-if)# ip access-group name [in | out]
```

Now, **let's talk about verifying ACLs**. Once you've assigned the interfaces with their ACLs, the next step would be using the following commands to verify your ACLs:

```
R1# show ip interface interface
```

```
R1# show access lists
```

This will allow you to view either what access list is set to the interface you're attempting to view, *or*, view your current access lists. Now, once an ACL has been applied and some testing has begun, the **show access-lists** command can give you some statistics on the access list. For instance, some of the statements have been *matched*, meaning, when traffic is generated that matches an ACL statement, the command output from **show access-lists** should increase by defining the matches. By matches, I mean as to how many times the criteria for that statement was met.

You can clear specific ACLs by issuing the following command:

```
R1# clear access-list counter access-list-id
```

For the sake and proficiency of IOS, there is a bit of logic that is processed when creating ACLs. For example, internal sequencing will not allow things such as deny ranges before singular hosts. For instance, what if we made an ACL that denied all hosts from 192.168.10.0, only to put a permit statement that allowed only host 192.168.10.10?

The result would be an error that does not allow the statement to exist, because even if 192.168.10.10 is allowed, at first, you denied all hosts from 192.168.10.0. Now, at times, the IOS will change ACLs. For instance, let's assume you put multiple range network statements to deny, while also putting multiple permit statements at the end. The result would be the IOS placing all the permit statements at the beginning of the ACL, regardless of what was typed first.

Extended ACLs

Now we can talk about Extended ACLs. In the event where you choose to be more specific with what kind of traffic you would like to filter, Extended ACLs hands this to you with the ability to filter based on source address, destination address, protocol, and port number.

To begin, you can use these access lists with keywords or ports. For example, if you wanted to allow all TCP traffic from the network 192.168.10.0 going through port 22 or SSH, you would use the following ACLs:

```
R1(config)# access-list 100 permit tcp 192.168.10.0 0.0.0.255 any eq 22
```

Or

```
R1(config)# access-list 100 permit tcp 192.168.10.0 0.0.0.255 any eq ssh
```

Now, we're going to view the syntax of the command, following that are descriptions of the parameters:

access-list *access-list-number* {deny | permit | remark} *protocol* {source *source-wildcard*} [operator port [port-number or name]] {destination *destination-wildcard*} [operator port [port-number or name]]

Parameter	Description
<i>access-list-number</i>	Identifies the access list using a number in the ranges 100 to 199, and 2000 to 2699.
deny	Denies access if the conditions are matched.
permit	Permits access if the conditions are matched.
remark	Used to enter a remark or comment.
<i>Protocol</i>	Name or number of an Internet protocol. Common keywords include icmp , ip , tcp or udp . To match any Internet protocol (including ICMP, TCP, and UDP) use the ip keyword.
<i>source</i>	Address of the network or host from which the packet is being sent.
<i>source-wildcard</i>	Wildcard bits applied to the source.
<i>destination</i>	Number of the network or host to which the packet is being sent.
<i>destination-wildcard</i>	Wildcard bits to be applied to the destination.
<i>operator</i>	(optional) Compares source or destination ports. Possible operands include lt (less than), gt (greater than), eq (equal), neq (not equal), and range (inclusive range).
<i>port</i>	(optional) The decimal number or name of a TCP or UDP port.
established	(optional) For the TCP protocol only: Indicates an established connection.

You can also create an extended ACL with names instead of numbers. Using the following command will take you into the extended named ACL configuration mode:

```
R1(config)# ip access-list extended named-acl
```

At last, you will need to verify that your Extended ACLs are functioning. You can do this with various commands, such as **show access-lists** or **show ip interface interface**. These will output certain details that can confirm the status of your ACLs. After an ACL has been implemented, you can verify if it works. As a final wrap up to this section, let's say you want to create an ACL that would only allow users on the 10.1.1.0/24 network to have HTTP access to the web server 10.1.3.8, the correct ACL would be:

```
access-list 101 permit tcp 10.1.1.0 0.0.0.255 host 10.1.3.8 eq 80
```

Troubleshooting ACLs

Before we begin this section of this chapter, we need to learn about general ACL logic. ACL logic is the steps that an ACL will take to determine what to do with a packet. Let's talk about **inbound ACL processes** first.

1. The packet arrives through the interface.
 - a. If the packet matches an ACE, go to step 2.
 - b. If there is not match, then the packet will be discarded (implicit deny any).
2. The packet has matched an ACE.
 - a. If it is a permit, the packet is received.
 - b. If it is a deny, the packet is discarded,

That's how the process works for an inbound ACL. It's as simple as that, an **outbound ACL process works** similarly.

1. The packet is getting ready to be sent.
 - a. If the packet matches an ACE, go to step 2.
 - b. If there is not a match, then the packet will be discarded (implicit deny any).
2. The packet has matched an ACE.
 - a. If it is a permit, the packet is sent outbound.
 - b. If it is a deny, the packet is discarded.

The processes are remarkably similar, the only differencing being whether it is sent or if it has arrived.

Introduction to Dynamic Host Configuration Protocol

Within a network, every device that connects to the network has a unique IP address. Network administrators will assign static IP address to routers, servers, printers, and other devices that will need their own unique IP address for the sake of convenience.

In addition, static addresses allow administrators to initiate the role of remote administration. It is easier for network administrators to access a device when everything has been planned out. The problem with this, is that it can be potentially difficult task. Additionally, for mobile employees working in different locations, the static configuration of networks may be a bit of a hassle.

To solve this problem, Dynamic Host Configuration Protocol (DHCP) was created to simplify IP address management for any device attempting to join the network. A centralized DHCP server enables organizations to administer dynamic IP addresses from a pool. This ensure effective and consistent addressing for all.

DHCPv4 assigns IPv4 address and other network configuration information dynamically. This means that information such as IP addresses, subnets, the default gateway, and other information is sent out to all who require it. However, not every site will require a dedicated DHCP server. In this event, Cisco IOS features a special feature called “Easy IP” which allows a router to work as a fully functioning DHCP server.

DHCPv4 includes three different address allocation mechanisms:

- **Manual Allocation** – The administrator assigns a pre-allocated IPv4 address to a client, and DHCPv4 communicates only the IPv4 address to the device.
- **Automatic Allocation** – DHCPv4 automatically assigns a static IPv4 address permanently onto a device. This means that the IP address is *not* leased and permanently unusable.
- **Dynamic Allocation** – DHCPv4 dynamically leases an IPv4 address for a limited time, *or*, until the client disconnects.

Dynamic allocation is the most commonly used DHCPv4 mechanism, allowing the reuse of IP addresses.

DHCPv4 Operation

The process of a DHCPv4 operation is relatively simple. DHCPv4 works in a client/server mode. When it is communicated with, it will initiate a certain process to provide the IP address. Let’s talk about each step:

0. **Lease Origination** – When the client wants to join the network, they will initiate a four step process.
1. **DHCP Discover (DHCPDISCOVER)** – This message is broadcasted by the client to find a DHCP server. Because the client has no Layer 3 information, this message will be sent as a Layer 2/3 broadcast.
2. **DHCP Offer (DHCPOFFER)** – When the DHCPv4 server receives a DHCPDISCOVER message, it will reserve an available IPv4 address for the client. The server will create an ARP entry consisting of the MAC address of the client and the leased IPv4 address.
3. **DHCP Request (DHCPREQUEST)** – When the client receives the DHCPOFFER from the server, it sends back a DHCPREQUEST message. This message is used for lease origination, the DHCPREQUEST serves as a binding acceptance notice to the select server for the parameters it has offered and an implicit decline to any other servers that have offered the client a DHCPOFFER.

4. **DHCP Acknowledgement (DHCPACK)** – Once the server has received the DHCPREQUEST message, the server verifies the lease information with an ICMP ping to that address to ensure it is not being used already, creates a new ARP entry for the client lease, and replies with a unicast DHCPACK message. The DHCPACK message is basically the DHCPOFFER, the only difference is the change in the message field.

There is also lease renewal, which is where a client will send a DHCPREQUEST directly to the DHCPv4 server that offered the IPv4 address. If a DHCPACK is not received, the client will broadcast a DHCPREQUEST so that a different DHCPv4 server can assist. Upon receiving a DHCPREQUEST message, the server will verify with the DHCPACK.

DHCPv4 Message Format

The DHCPv4 message format is used for all DHCPv4 transactions. All DHCPv4 messages are encapsulated within the UDP protocol, sending from UDP source port 68 and destination 67. DHCPv4 messages sent from the server to client use UDP source port 67 and destination port 68.

8	16	24	32
OP Code (1)	Hardware Type (1)	Hardware Address Length	Hops (1)
Transaction Identifier			
Seconds – 2 bytes		Flags – 2 bytes	
Cisco IP Address (CIADDR) – 4 bytes			
Your IP Address (YIADDR) – 4 bytes			
Server IP Address (SIADDR) – 4 bytes			
Gateway IP Address (GIADDR) – 16 bytes			
Client Hardware Address (CHADDR) – 16 bytes			
Server Name (SNAME) – 64 bytes			
Boot Filename – 128 bytes			
DHCP Options - variable			

I'm going to identify the information that will fill each of these categories:

- **Operation (OP) Code** – Specifies the general type of message. 1 will be a request, 2 will be a reply.
- **Hardware Type** – Identifies the hardware. For instance, 1 is Ethernet, 15 is Frame Relay, and 20 is serial line.
- **Hardware Address Length** – Specifies the length of the address.
- **Hops** – Controls the forwarding of messages. Set to 0 by client before request transmission.
- **Transaction Identifier** – Used by the client to match the request with replies received from DHCPv4 servers.
- **Seconds** – Identifies the number of seconds since the client began attempting to acquire or renew a lease.
- **Flags** – Used by a client that does not know its IPv4 address when sent.
- **Client IP Address** – Used by a client during lease renewal when the address of the client is valid and usable.
- **Server IP Address** – Used by the server to identify the address of the server that the client should use next.
- **Gateway IP Address** - Routes DHCPv4 messages when DHCPv4 relay agents are involved.
- **Client Hardware Address** – Specifies the physical layer of the client.
- **Server Name** – Used by server sending a DHCPOFFER or DHCPACK message.
- **Boot Filename** – Optionally used by a client to request a particular boot file in DHCPDISCOVER message.
- **DHCP Options** – Holds DHCP options.

Before we begin with the next reviewing of messages, we will need to understand what certain items mean:

MAC	Media Access Control Address
CIADDR	Client IP Address
GIADDR	Gateway IP Address
CHADDR	Client Hardware Address

Let's create a scenario. In this scenario, our client sends a DHCPDISCOVER packet, from here, the DHCP server is on the same segment and will receive the request. Here is the request. Notice how the GIADDR field is blank, or, has 0.0.0.0, along with many other fields being blank. This is because the client has no third layer information.

In addition to that, the SRC MAC is MAC A to represent the computers MAC address. The destination MAC and IP destination is the maximum value to represent a broadcast address. So this message is being sent through both layer 2 and 3, requesting information.

Ethernet Frame	IP	UDP	DHCPDISCOVER
DST MAC: FF:FF:FF:FF:FF:FF SRC MAC: MAC A	IP SRC: 0.0.0.0 IP DST: 255.255.255.255	UDP 67	CIADDR: 0.0.0.0 GIADDR: 0.0.0.0 Mask: 0.0.0.0 CHADDR: MAC A

Now, this is the DHCPv4 offer message, which looks far more different. First of all, this packet has layer 3 information. Second of all, this is still being sent with a MAC address. Now, this offer contains a client IP, the gateway, and the subnet mask. (GIADDR is empty because why not?). The source address is the DHCPv4 server. Notice how the IP address ends with the last available host address? It was likely statically configured.

Ethernet Frame	IP	UDP	DHCP Reply
DST MAC: MAC A SRC MAC: MAC Server	IP SRC: 192.168.1.254 IP DST: 192.168.1.10	UDP 68	CIADDR: 192.168.1.10 GIADDR: 0.0.0.0 Mask: 255.255.255.0 CHADDR: MAC A

DHCPv4 Server and Client Configuration

So we mentioned that a Cisco router can be configured to run as a fully featured DHCPv4 server with Easy IP. This is true, but, how can we configure it to do so? Some of the first steps begin with planning the process:

1. **Excluding IP Address** – The first step is to ensure that some IP addresses will *not* be assigned from the pool. For instance, printers and servers should have static IP addresses, so, configure your router to exclude IP addresses from the DHCP pool with this command:

```
R1(config)# ip dhcp excluded-address Low-address [high-address]
```

With that command, you may either choose to exclude certain addresses, or, an entire range of addresses.

2. **Configuring a DHCPv4 Pool** – This step will require the defining of a DHCPv4 pool. This means you will need to create a name, to do so, use the following command:

```
R1(config)# ip dhcp pool pool-name
```

That command will allow you to create a pool. From there, it will take you to DHCP configuration mode.

3. **Configure Specific Tasks** – The next step will be to configure particular details for your DHCP server, from here, I will label and describe each command and the syntax:

Required Tasks	Command
Define the address pool.	network network-number [mask / /prefix-length]
Define the default router or gateway.	default-router address [address2...address8]

And the additional commands are optional tasks:

Optional Tasks	Command
Define a DNS server.	dns-server address [address2...address8]
Define the domain name.	domain-name domain
Define the duration of the DHCP lease.	lease { days [hours] [minutes] infinite }
Define the NetBIOS WINS server.	netbios-name-server address [address2...address8]

4. **Having DNS enabled or Disabled** – To enable or disable DNS, issue the following commands:

```
R1(config)# no service dhcp
```

```
R1(config)# service dhcp
```

From there, you should have a fully configured DHCP server. In the event that you need to identify all details and ensure that DHCP should be running correctly, issue the following commands to check your DHCP configuration:

```
R1# show running-config | section dhcp
```

 (allows you to view view the DHCP configuration in the running configuration.)

```
R1# show ip dhcp binding
```

 (this command will allow you to check all leased IP addresses.)

```
R1# show ip dhcp server statistics
```

 (this command will allow you to view general DHCP statistics to verify if the messages are being received or sent by the router.)

Let's talk about a very important subject called **DHCP Relay**. Within a large network, there may be a large amount of servers in one area. This is called a *server farm*. Now, let's say we had a host attempting to communicate with a DHCP server, now, this host cannot go beyond their network, so their network (192.168.10.0/24) cannot communicate with the network that has a DNS server (in 192.168.11.6.)

The solution of this problem, is that an administrator can add DHCPv4 servers on all subnets, the problem? That would be costly. The solution? The use of Cisco IOSs helper address. This will allow a router to forward DHCPv4 broadcasts to the DHCPv4 server. When a router will forward it, it will act like a DHCPv4 relay agent.

You can configure this by issuing the following command with the DHCPv4 server:

```
R1# int interface  
R1(config-if)# ip helper-address ip-address
```

Now, what if we wanted our router to be a DHCP client? Sometimes, a Cisco router within a SOHO and branch of a site need to be configured as DHCPv4 clients. Usually, this depends on the ISP, however, it's good to know this.

To begin, you will need to go into the interface that you will configure to function as a DHCP client:

```
SOHO(config)# interface interface
```

Next, you will need to issue the following commands:

```
SOHO(config-if)# ip address dhcp  
(If the interface needs to be activated) SOHO(config-if)# no shutdown
```

Finally, go back to privileged EXEC mode, and issue the following command to check the configuration:

```
SOHO# show ip interface interface
```

Troubleshooting DHCPv4

There can be moments where there will be problems with DHCP, however, there is a good set of steps you can take to solve any arising problems, take these steps:

1. **Resolve IPv4 Address Conflicts** – Any IPv4 address can expire on a client connected to the network. If the client does not renew, the DHCPv4 server can reassign it to another. When the client requests an IPv4 address, the DHCPv4 server might not respond quickly, the client will use the last IPv4 address, resulting in a conflict. Your DHCPv4 server will record conflicts, to view conflicts, issue the following command:

```
R1# show ip dhcp conflict
```

2. **Verify Physical Interface** – Use the **show interfaces interface** command to confirm that a router interface is acting as the default gateway for the client. If the state of the interface is anything other than up, then traffic will not go.
3. **Test Connectivity using a Static IP Address** – This means that you need to verify network connectivity by configuring a static IPv4 address on a client. If the workstation cannot reach something, then the problem is network connectivity.
4. **Verify Switch Port Configuration** – If the DHCPv4 client cannot obtain an IPv4 address from DHCPv4, attempt to obtain an IPv4 address from the DHCPv4 server by manually forcing the client to send a DHCPv4 request. If there is a switch between the client and DHCPv4 server, switch port configuration issues may be the cause.
5. **Test from the same VLAN or Subnet** – This is the final step to distinguish whether DHCPv4 is function correctly when the client is on the same subnet or VLAN as the DHCPv4 server. If it is working correctly when the client is on the same subnet or VLAN, it may be the DHCP relay agent. If the problem persists on the same subnet or VLAN, it may be the DHCPv4 server.

So what about router configuration? When the DHCPv4 server is located on a separate LAN from the client, the router interface facing the client must be configured with DHCP relay. There are two tips, you can verify that **ip helper-address** is configured. You can also check that the **service dhcp** command has been run. You can just execute that command in the router console to get DHCP up and running.

The last thing you can do is configure an access list specifically for any DHCP packets, follow the following commands:

```
R1(config)# access-list 100 permit udp any any eq 67
R1(config)# access-list 100 permit udp any any eq 68
R1(config)# end
```

Try your DHCP operations again. If there are any failures, then you can begin debugging that ACL with the following:

```
R1# debug ip packet 100
```

From there, you will receive an output of any sent or received packets following that criteria. You can also use the following command, it will report server events, like address assignments and database updates:

```
R1# debug ip dhcp server events
```

RSE 11 – NETWORK ADDRESS TRANSLATION

Introduction to Network Address Translation

Theoretically, there is a maximum of 4.3 billion IPv4 addresses. There are many devices in the world, soon, we will exceed that limit and ultimately run out of IP addresses. To stop this from happening, a long term solution, IPv6, was engineered, however, to adapt to a new system will take time. As a result, a short term solution was developed, one of them includes Network Address Translation (NAT) and RFC 1918 private IPv4 addresses.

This allowed us to continue the use of the internet, to adapt and work with what has been given. In this section, there will be discussion of NAT, NAT characteristics, terminology, and operations. The different versions of NAT, including static, dynamic, and overloaded NAT. The benefits and disadvantages of NAT. The configuration, verification, and overall analysis of static, dynamic, and overloaded NAT. There will be discussion on port forwarding, troubleshooting NAT and the use of NAT for dual stack environments (or, IPv4 to IPv6, and vice versa.)

Network Address Translation Details and Terminology

As said, there are only so many IP addresses and so many device in the world, leading to the eventual depletion of IPv4 addresses. This is a form of scarcity. From this point, there was a need to create a way to preserve IP addresses, thus, private IP addresses and NAT were developed.

Private Internet Addresses are defined in RFC 1918:		
Class	RFC 1918 Internal Address Range	CIDR Prefix
A	10.0.0.0 – 10.255.255.255	10.0.0.0/8
B	172.16.0.0 – 172.31.255.255	172.16.0.0/12
C	192.168.0.0 – 192.168.255.255	192.168.0.0/16

With a NAT router, you are able to translate from an internal network (private IPv4 address space) to the internet (public IPv4 address space). Meaning that a single public IPv4 address can represent even millions of potential addresses. Without NAT, the exhaustion of IPv4 address space would have occurred before the 21st century. The problem with NAT includes the certain limitations and other characteristics that make want to eventually transition to IPv6, though, that is a very far away time.

So what is **NAT**? NAT has various uses. Its main use is to conserve public IPv4 addresses. It does this by allowing networks to use private IPv4 address and represent them to the internet only using one public IP address. There are various benefits to using NAT. Some of these include the added benefit of adding a degree of security to the network (since it hides the internal IPv4 address from outside the network.)

To allow NAT to function, the primary need is to have the router configured with NAT. These can be configured with more than one valid public IPv4 address. Behind each of these public IP addresses is a **NAT pool**. The NAT pool is the internal IP addresses being represented by the one public IP address.

When a NAT enabled router sends traffic, the router will translate the internal IPv4 address of the device to the public address that represents all IP addresses from within the NAT pool. To outside devices, all traffic entering and exiting the network will appear to have a public IPv4 address from the NAT pool.

Normally, NAT routers operate at the border of a stub network. A stub network is a network that has a single connection to its neighboring network, basically, one way in and out. When a device inside the stub network wishes to communicate,

An Introduction to Networks and Cisco

the packet is forwarded to the border router, then, the border router will perform the NAT process, translating the address from an internal private address to a public, outside, and routable address.

With NAT, there are certain terms that need to be learned to work with NAT. When using NAT, IPv4 addresses have different designation based on whether they are on the private network (inside), or, the public network (internet.) Within NAT, there are four types of addresses:

- Inside local address – The address of the source as seen from inside the network.
- Outside local address – The address of the destination as seen from the inside network.
- Inside global address – The address of source as seen from the outside network.
- Outside global address – The address of the destination as seen from the outside network. It is a globally routable IPv4 address assigned to a host on the internet.

Inside Local 192.168.10.10 (Source)	Inside Global 209.165.201.1 (Router)	Outside Global 209.165.200.226 (Link to ISP)	Outside Local 209.165.201.1 (Destination)
---	--	--	---

NAT in Action

That explanation may have been confusing, I get it. There is a better way to explain this (use this routing table with it):

NAT Table			
Inside Local	Inside Global	Outside Local	Outside Global
192.168.10.10	209.165.200.226	209.165.201.1	209.165.201.1

1. Let's say our source, 192.168.10.10, sends their packet to their destination, 209.165.201.1.
2. That packet is sent to Router 1, which is sent to Router 2.
3. Within Router 2's NAT table, it has a defined set of IP address, inside local and global, and outside local and global. In this scenario, the inside local address (192.168.10.10) is translated to 209.165.200.226 and it is sent to its destination.
4. Router 2 will send the packet with the translated source address towards the destination.
5. The destination responds with a packet addressed to the inside global address of PC1 (209.165.200.226.)
6. Router 2 will receive a packet back from the destination with the destination address 209.165.200.226, R2 will check the NAT table and find an entry for that address (209.165.200.226). It finds the match of the inside local address (192.168.10.10) and the packet is forwarded to PC1.

Now that we understand what the NAT table is, we can use this

Static NAT and Dynamic NAT

Now, we need to know that there are three different types of NAT. Static Network Address Translation (static NAT) is the first one. This is a one-to-one address mapping between local and global addresses. Static NAT uses a one-to-one mapping of local and global addresses. These are configured by network administrators to remain constant.

For example, to get one point (let's say, Server 1), 192.168.10.10 will need to be translated through the NAT enabled router. Well, static is when the NAT table is configured with a static NAT entry. So, if PC1 from the outside of your network

attempted to use ssh to access Server 1, your router would be configured so that the inside global address (209.165.200.226) would point directly to 192.168.10.10, which is the server.

This is useful for devices that must have a static address to be accessible. It's also useful for devices that must be accessed by offsite personnel, but not the entire internet. From our example, PC1 can SSH into Server 1 using their global address. Router 2 will translate the inside global address 209.165.200.226 into 192.168.10.10 to access the server. Static NAT requires that enough public addresses are available to satisfy the total number of simultaneous user sessions.

Dynamic NAT uses a pool of public addresses and assigns them on a first come-first serve sort of service. In addition, enough public addresses must be available to satisfy the total number of simultaneous user sessions.

Port Address Translation (PAT)

Port Address Translation (PAT), also known as NAT overload, maps multiple private IPv4 addresses to a single public IPv4 address or a few addresses. This is what ISPs and most places do to. Home routers will do this as well. The ISP will assign one address to the router, and yet, several members of the household can use the internet at the same time.

With PAT, multiple addresses can be mapped to one or few addresses, because each private address is also tracked by a port number. When a device initiates a TCP/IP session, it generates a TCP or UDP session. When the NAT router receives a packet from the client, it uses a source port number to uniquely identify the specific NAT translation.

PAT ensures that devices use a different TCP port number for each session with a server on the internet. When a response comes back from the server, the source port number, which becomes the destination port number coming back from the session, determines which device the router forwards the packets to. The PAT process also validates that the incoming packets were requested, adding security to the sessions.

This is how it goes when using an example. Let's say we have a network with two hosts connecting to two different servers at the same time. The source address for host A is (192.168.10.10, connecting through port 1555) while you have host B (using 192.168.10.11, connecting through port 1331), connecting to two destinations on the internet, those destinations being server 1 (209.165.201.1:80) and server 2 (209.165.202.129:80).

The requests will go through the internet, going through the router with NAT overload configured, this is what the table will look like:

Inside Global IP Address	Inside Local IP Address	Outside Local IP Address	Outside Global IP Address
209.165.200.226:1555	192.168.10.10:1555	209.165.201.1:80	209.165.201.1:80
209.165.200.226:1331	192.168.10.11:1331	209.165.202.129:80	209.165.202.129:80

With this table, you can tell that the NAT table will know where exactly to send all of the information. It's mapped that those IP addresses that are sending the information to the server will be recognized based on the port. In the event where the servers will need to reply, the process of sending the packets will be reversed.

Next Available Port technology was created in the event where an inside local IP address will attempt to use the same port as another inside local IP address. PAT will attempt to preserve the original source port, however, if the original source port is already being used, PAT will assign the next available port number from the following port groups: 0-511, 512-1,023, 1,024-65,535.

To compare NAT and PAT is relatively simple. If you have enough public addresses, you can dedicate those IP addresses to certain items or just set it to be dynamic. If you are operating a home router of some sort, it's better to use PAT.

What about IPv4 packets carrying data other than a TCP or UDP segment? These don't have port numbers, PAT will translate them to the most common protocol carried by IPv4 that does not use either TCP or UDP. These protocols would be things like ICMP.

Benefits and Disadvantages of NAT

NAT comes with various benefits. Here are some of the key point benefits:

- NAT conserves the legally registered addressing scheme by allowing the privatization of intranets. NAT conserves addressing through application port-level multiplexing. With NAT overload, internal hosts can share a single public IPv4 address for all communications.
- NAT increases network flexibility of connections to public networks. Additional NAT pools can be implemented to ensure reliable public network connections.
- NAT provides consistent internal network addressing. On a network not using private IPv4 address and NAT, changing the public IPv4 addressing scheme requires reconfiguring all hosts. With NAT, it will allow the private IPv4 address scheme while allowing easy changes to the public addressing scheme. Meaning, you can change providers, and your network will stay the same.
- NAT provides network security. Private networks will not advertise addresses or topology, resulting in only a limited amount of information being given out. This does *not* replace firewalls.

NAT does come with some drawbacks. For example:

- Performance is degraded. Meaning that some real time protocols, such as VoIP, will experience increase latency due to the translation of each IPv4 address within packets.
- End-to-end functionality is degraded. Many protocols require end-to-end addressing from the source to the destination. Some applications will not work with NAT. For example, some security applications, such as digital signatures, will fail because the source IPv4 address changes before reaching the destination. Applications that use physical address do not reach destinations that are translated across a NAT router.
- End-to-end IPv4 traceability is also lost. It becomes more difficult to trace packets that undergo numerous packet address changes will result in difficult troubleshooting and tracing.
- Tunneling becomes more complicated. This means that tunneling protocols, such as IPsec, will no longer work, simply because NAT modifies the values in the header, interfering with the integrity of IPsec.
- And finally, initiating TCP connections can be disrupted. Services that require the initiation of TCP connections from other networks can be disrupted unless NAT has been configured to support protocols that have to do with initiating TCP connections, those connections will be interrupted and packets cannot reach their destinations.

Configuration of Static and Dynamic NAT

Let's talk about how to configure NAT, first, specifically static NAT. There is some preparation before you can do so:

1. The first task is creating a mapping between the inside local address and the inside global address. For example, the 192.168.10.254 inside local address and the 209.165.201.5 inside global address will need to be configured. Within the inside of the network, the web server will be recognized as the inside local network, while on the outside, client computers will need to communicate to 209.165.201.5.
2. Once the mapping has occurred, the interfaces will need to be configured as either inside or outside. For a router, the Serial 0/0/0 interface is an inside interface while Serial 0/1/0 is an outside interface. Packets arriving on the inside interface of R2 (Serial 0/0/0) from the configured inside local IPv4 address (192.168.10.524) are translated then forwarded towards the outside network. Packets arriving on the outside interface of R2 (Serial 0/1/0), that are addressed to the configured inside global IPv4 address (209.165.201.5) are translated to the inside local address (192.168.10.524), the forwarded into the network.

Let's talk about the steps:

Step	Action	Notes
1	Establish static translation between an inside local address and an outside global address: Router(config)# ip nat inside source static local-ip global-ip	Enter the no ip nat inside source static command in global configuration mode to remove the dynamic source translator.
2	Specify the inside interface. Router(config)# interface interface	Enter the interface command. The CLI prompt will change from (config)# to (config-if)#.
3	Mark the interface as connected to the inside. Router(config-if)# ip nat inside	Basically, whatever interface is the one inside the network, label it.
4	Exit from that configuration mode. Router(config-if)# exit	
5	Specify the outside interface. Router(config)# interface interface	
6	Mark the interface as connected to the outside. Router(config-if)# ip nat outside	Whatever interface is facing out of the network.

From there, you be able to thoroughly analyze static NAT. Static NAT translation processes between the client and the web server. Usually static translations are used when clients on the outside network (Internet) need to reach servers on the inside (internal) network.

Now, what if you need to view the current configurations for NAT? Static translation is always present in the NAT table by issuing the following command:

R1# show ip nat translations

It will show static translations and active sessions. In addition, you can clear and show NAT statistics with the following commands:

```
R1# clear ip nat statistics
R1# show ip nat statistics
```

Now let's learn how to configure **Dynamic NAT**. While static NAT provides permanent mapping, dynamic NAT will allow automatic mapping of inside local addresses to inside global addresses. The most important detail to dynamic NAT is that it requires a pool of inside global addresses to function; meaning, your router will need to represent more than one IP address to allow everyone their own translation.

Another problem with this is if there are no more inside global addresses, devices will have to wait to be accessed or access. Now, follow these instructions to define dynamic NAT:

Step	Action and Notes	Command
1	Define a pool of global addresses to be used for translation. The netmask or prefix-length keyword indicates which address bits belong to the network. Remember that these are the global addresses .	ip nat pool name start-ip end-ip{netmask netmask prefix-length prefix-length} EXAMPLE: ip nat pool NAT-POOL1 209.165.200.226 209.165.200.240 netmask 255.255.255.224
2	Configure a standard access list permitting addresses that should be translated. Remember there is an implicit deny all statement at the end of each ACE. Within the access-list, you need to permit the source (network portion) and then, you need to define what bits belong to the hosts.	access-list access-list-number permit source [source-wildcard] EXAMPLE: access-list 1 permit 192.168.0.0 0.0.255.255
3	Establish dynamic source translation, specifying the access list and pool defined in prior steps. This step is also called binding .	ip nat inside source access-list-number pool name EXAMPLE: ip nat inside source list 1 pool NAT-POOL1
4	Specify the inside interface.	1. interface interface 2. ip nat inside
5	Specify the outside interface.	1. interface interface 2. ip nat outside

Since this is plenty of practice for configuring, you should have plenty of experience on this subject. What about verification? The process for checking how you configure NAT is relatively simple. The command **show ip nat translations** will display all details. Doing a **show ip nat translations verbose** will provide you more information.

Configuration of PAT and Port Forwarding

We've already discussed what, PAT, also called NAT overload, does for us. It conserves addresses inside the global address pool by allowing the router to use one inside global address for many inside local addresses. Meaning, one public IPv4 address can be used for hundreds, even thousands of internal private IPv4 addresses.

When this type of translation is configured, PAT will use TCP/UDP ports to make translations from inside global addresses to inside local addresses. It's usually recommended not to exceed 4,000 when using PAT. There are two ways to configure PAT depending on how the ISP allocates public IPv4 addresses. For instance, ISPs allocate more than one public IPv4 address to the organization, or, they allocate a single public IPv4 address.

The first one is **configuring PAT for a pool of Public IP Addresses**.

If a site has more than one public IPv4 address, these addresses can be a part of a pool that is used by PAT, the configuration is similar to the configuration of dynamic NAT, only, and there are not enough public addresses for one-to-one mapping of inside and outside addresses. Here is a set of instructions on how to configure PAT for a pool:

Step	Action and Notes	Command
1	Define a pool of global addresses to be used for overload translation. So, it's going to be the same thing as Dynamic NAT, using the same global addresses.	ip nat pool name start-ip end-ip{netmask netmask prefix-length prefix-length} EXAMPLE: ip nat pool NAT-POOL1 209.165.200.226 209.165.200.240 netmask 255.255.255.224
2	Configure a standard access list permitting addresses that should be translated. Remember there is an implicit deny all statement at the end of each ACE. Within the access-list, you need to permit the source (network portion) and then, you need to define what bits belong to the hosts.	access-list access-list-number permit source [source-wildcard] EXAMPLE: access-list 1 permit 192.168.0.0 0.0.255.255
3	Establish overload translation, specifying the access list and pool defined in prior steps.	ip nat inside source list access-list-number pool name overload EXAMPLE: ip nat inside source list 1 pool NAT-POOL1 overload
4	Specify the inside interface.	1. interface interface 2. ip nat inside
5	Specify the outside interface.	1. interface interface 2. ip nat outside

An Introduction to Networks and Cisco

The next way to **configure PAT is for a single address**. Typically, routers connected to ISPs (especially home networks and Small Office/Home Offices) will run this configuration. Due to the nature of this type of configuration, you will not need to configure a pool of global addresses.

Step	Action and Notes	Command
1	Configure a standard access list permitting addresses that should be translated. Remember there is an implicit deny all statement at the end of each ACE. Within the access-list, you need to permit the source (network portion) and then, you need to define what bits belong to the hosts.	access-list access-list-number permit source [source-wildcard] EXAMPLE: access-list 1 permit 192.168.0.0 0.0.255.255
2	Establish dynamic source translation, specifying the ACL, exit the interface and overload options. Ensure that it is defined as overload.	ip nat inside source list access-list-number interface interface overload EXAMPLE: ip nat inside source list 1 interface serial 0/1/0 overload
3	Specify the inside interface.	1. interface interface 2. ip nat inside
4	Specify the outside interface.	1. interface interface 2. ip nat outside

So, the configuration of this is fairly simple. The process of NAT overload is the same whether a pool of addresses are used or a single address. We know that addresses are translated into either a predefined list of IP addresses, or, a singular IP address, to use a certain port. We know how PAT works. The use of an interface only comes in when we have a single public IP address.

How do we verify PAT? We can verify it using various commands like **show ip nat translations** or **show ip nat statistics** in privileged EXEC mode. You can also use **show ip nat translations verbose** for more detail.

Now that we've gone through various configurations, let's talk about **Port Forwarding**. Port Forwarding, also known as **Tunneling**, is the act of forwarding traffic addressed to a specific network port from one network to another. This allows an external user to reach a port on a private IPv4 address.

Typically, peer-to-peer file-sharing programs require port forwarding. Because NAT hides internal addresses, peer-to-peer only works from the inside out where NAT can map outgoing requests against incoming replies. A big problem with NAT is that it doesn't allow requests from the outside. Port forwarding allows users on the Internet to access internal server by using the WAN port address of the router and the matched external port number. When a request is sent to the IPv4 address of the WAN port via the Internet, the router forwards the requests to the appropriate server on the LAN.

So how do I Port Forward on IOS? Implementing a port forward with IOS commands is similar to configuring static NAT. You can follow these instructions for that:

Step	Action and Notes	Command
1	Establish a static translation between an inside local address and local port and an inside global address and global port.	ip nat inside source static protocol-tcp-udp internal-ip internal-port external-ip external-port EXAMPLE: ip nat inside source static tcp 192.168.10.254 80 209.165.200.225 8080
2	Specify the inside interface.	1. interface interface 2. ip nat inside
3	Specify the outside interface.	1. interface interface 2. ip nat outside
4	Ensure the configuration is correct.	show ip nat translations

The final subject for this is, what about NAT for IPv6? Since the 1990s, the concern about the depletion of IPv4 addresses was a troubling idea. Since then, we've created NAT and IPv6. With the security and usefulness of NAT, it gives a new meaning to the protocol, possibly halting the need to advance; however, it should not just replace proper network security.

IPv6, with a 128-bit address, provides **a lot** of addresses, making space not an issue. IPv6 was developed with the intention that it can make NAT for IPv4 with its translation between public and private IPv4 addresses obsolete. IPv6 does have its own version of NAT however.

Troubleshooting NAT

There may be moments where you have problems with NAT, I mean, think of the nature of NAT, there's bound to be at least one thing wrong. When there are IPv4 problems, there are various steps you can take to resolve the problem.

1. The first step in checking for issues within NAT is to review what the purpose of NAT is. If it's supposed to translate 192.168.10.10 to 209.165.200.226, then, it will be relatively easy to find the configuration that could be wrong.
2. Verify that correct translations exist in the translation table using the **show ip nat translations** command.
3. Use the **clear** and **debug** command can verify whether NAT is operating correctly or not. Check step 2 for changes.
4. Review in detail as to what is happening with packets and how to router is moving the packet. Using the command **show ip nat statistics** and **show ip nat translations** can provide details that can assist in the reconfiguration of NAT. You can clear all of the previously created statistics and translations with **clear ip nat statistics** and **clear ip nat translations ***
5. Next, there is the debug command. You can run **debug ip nat** in global configuration mode to verify the operation of NAT in detail. It will display all packets going through the router. If you want to view it in detail, do **debug ip nat detailed**.