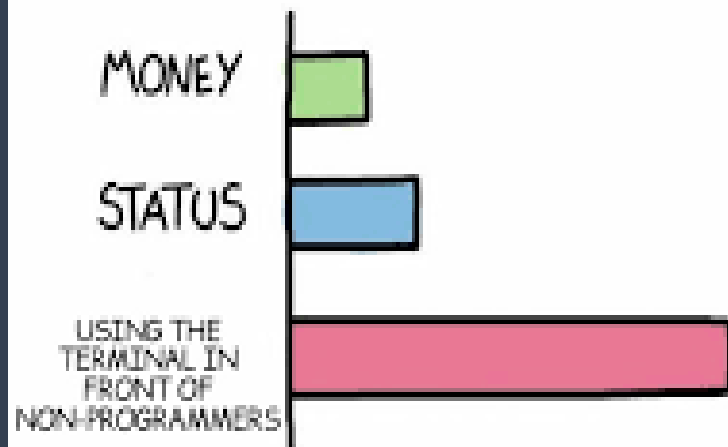


Unit - 2

Scripting Part - 2

WHAT GIVES PEOPLE FEELINGS OF POWER



RM ~/.BASHRC

**INSTEAD OF
~/.BASH_HISTORY**

made on imgur

Package Management

Package Management

- The most important determinant of distribution quality is the *packaging system* and the vitality of the distribution's support community.
- As we spend more time with Linux, we see that its software landscape is extremely dynamic. Things are constantly changing.
- Most of the top-tier Linux distributions release new versions every six months and many individual program updates every day.
- To keep up with this blizzard of software, we need good tools for **package management**.

Package Management

- Package management is a method of installing and maintaining software on the system.
- Today, most people can satisfy all of their software needs by installing packages from their Linux distributor. This contrasts with the early days of Linux when one had to download and compile source code in order to install the software. Not that there is anything wrong with compiling source code; in fact, having access to source code is the great wonder of Linux.
- It gives us (and everybody else) the ability to examine and improve the system. It's just that working with a precompiled package is faster and easier.

Package Management

- Different distributions use different packaging systems, and as a general rule a package intended for one distribution is not compatible with another distribution.
- Most distributions fall into one of two camps of packaging technologies: the Debian `.deb` camp and the Red Hat `.rpm` camp.
- There are some important exceptions, such as Gentoo, Slackware, and Foresight, but most others use one of the two basic systems

| Packaging System | Distributions (partial listing) |
|-------------------------------|--|
| Debian style (<i>.deb</i>) | Debian, Ubuntu, Xandros, Linspire |
| Red Hat style (<i>.rpm</i>) | Fedora, CentOS, Red Hat Enterprise Linux openSUSE, Mandriva, PCLinuxOS |

How a Package System Works

- The method of software distribution found in the proprietary software industry usually entails buying a piece of installation media such as an “install disk” and then running an “installation wizard” to install a new application on the system.
- Linux doesn’t work that way. Virtually all software for a Linux system is found on the Internet.
- Most of it is provided by the distribution vendor in the form of package files, and the rest is available in source code form, which can be installed manually.

Package File

- The basic unit of software in a packaging system is the package file.
- A package file is a compressed collection of files that comprise the software package.
- A package may consist of numerous programs and data files that support the programs.
- In addition to the files to be installed, the package file also includes metadata about the package, such as a text description of the package and its contents.
- Additionally, many packages contain pre- and post-installation scripts that perform configuration tasks before and after the package installation.

Package File

- Package files are created by a person known as a package maintainer, often (but not always) an employee of the distribution vendor.
- The package maintainer gets the software in source code form from the upstream provider (the author of the program), compiles it, and creates the package metadata and any necessary installation scripts.
- Often, the package maintainer will apply modifications to the original source code to improve the program's integration with the other parts of the Linux distribution.

Repositories

- While some software projects choose to perform their own packaging and distribution, most packages today are created by the distribution vendors and interested third parties.
- Packages are made available to the users of a distribution in central repositories, which may contain many thousands of packages, each specially built and maintained for the distribution.

Repositories

- A distribution may maintain several different repositories for different stages of the software development life cycle.
- For example, testing repository, development repository, third-party repositories

Repositories

- Programs seldom stand alone; rather, they rely on the presence of other software components to get their work done.
- Common activities, such as input/output for example, are handled by routines shared by many programs.
- These routines are stored in what are called shared libraries, which provide essential services to more than one program.
- If a package requires a shared resource such as a shared library, it is said to have a dependency.
- Modern package management systems all provide some method of dependency resolution to ensure that when a package is installed, all of its dependencies are installed, too.

Low level and High Level Package Tools

- Package management systems usually consist of two types of tools:
- low-level tools that handle tasks such as installing and removing package files, and
- high-level tools that perform metadata searching and dependency resolution.

Installing a Package from a Package Repository

- High-level tools permit a package to be downloaded from a repository and installed with full dependency resolution

| Style | Command(s) |
|---------|---|
| Debian | <code>apt-get update</code> <code>apt-get install <i>package_name</i></code> |
| Red Hat | <code>yum install <i>package_name</i></code> |

Installing a Package from a Package Files

- If a package file has been downloaded from a source other than a repository, it can be installed directly (though without dependency resolution) using a low-level tool

| Style | Command(s) |
|---------|---|
| Debian | <code>apt-get update</code> <code>apt-get install <i>package_name</i></code> |
| Red Hat | <code>yum install <i>package_name</i></code> |

Removing a Package

- Packages can be uninstalled using either the high-level or low-level tools.

| Style | Command |
|---------|---|
| Debian | <code>apt-get remove <i>package_name</i></code> |
| Red Hat | <code>yum erase <i>package_name</i></code> |

Updating Packages from a Repository

- The most common package management task is keeping the system up-to-date with the latest packages. The high-level tools can perform this vital task in one single step

| Style | Command(s) |
|---------|--|
| Debian | <code>apt-get update; apt-get upgrade</code> |
| Red Hat | <code>yum update</code> |

Updating Packages from a File

- If an updated version of a package has been downloaded from a nonrepository source, it can be installed, replacing the previous version

| Style | Command |
|---------|---|
| Debian | <code>dpkg --install <i>package_file</i></code> |
| Red Hat | <code>rpm -U <i>package_file</i></code> |

Listing Installed Packages

- If an updated version of a package has been downloaded from a nonrepository source, it can be installed, replacing the previous version

| Style | Command |
|---------|--------------------------------------|
| Debian | <code>dpkg --get-architecture</code> |
| Red Hat | <code>rpm -qa</code> |

Determining Whether a Package Is Installed

| Style | Command |
|---------|--|
| Debian | <code>dpkg --status <i>package_name</i></code> |
| Red Hat | <code>rpm -q <i>package_name</i></code> |

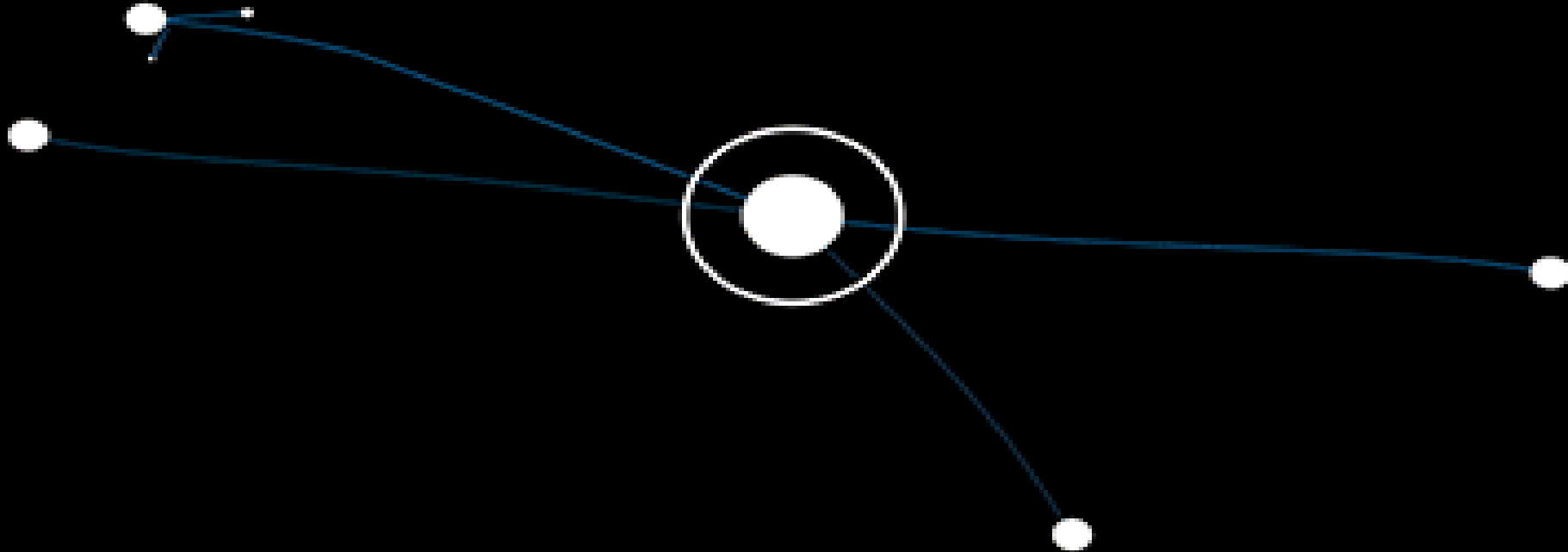
Displaying Information About an Installed Package

| Style | Command |
|---------|---|
| Debian | <code>apt-cache show <i>package_name</i></code> |
| Red Hat | <code>yum info <i>package_name</i></code> |

Finding Which Package Installed a File

| Style | Command |
|---------|---|
| Debian | <code>dpkg --search <i>file_name</i></code> |
| Red Hat | <code>rpm -qf <i>file_name</i></code> |

Networking

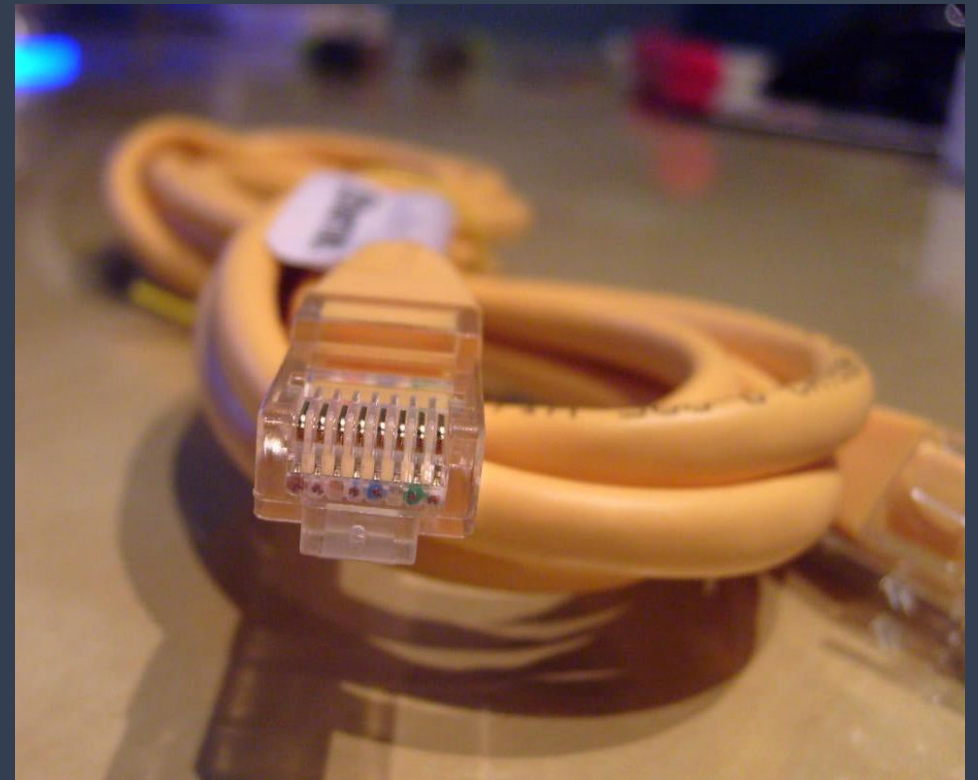


Network Terminology

A **network** is created when two or more computers communicate through some sort of connection. This connection could be created via several different technologies, including **Ethernet**, **fiber optic**, and **wireless**.

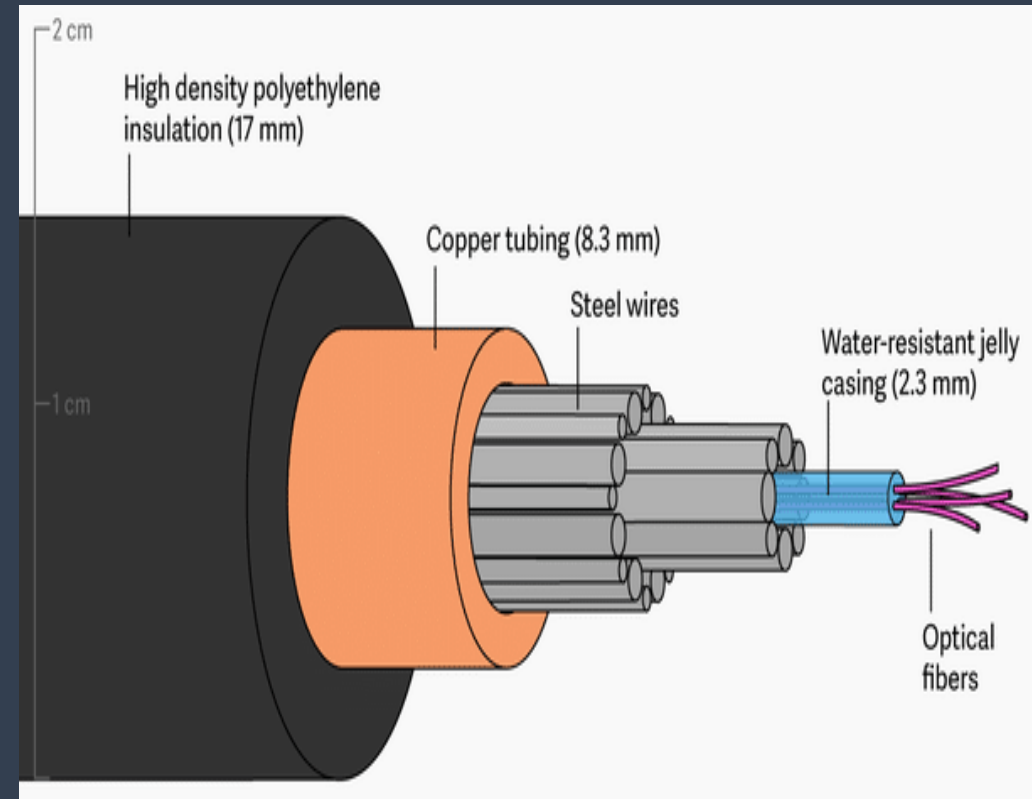
Network Terminology

Ethernet - A **network** is created when two or more computers communicate through some sort of connection. This connection could be created via several different technologies, including **Ethernet**, **fiber optic**, and **wireless**.



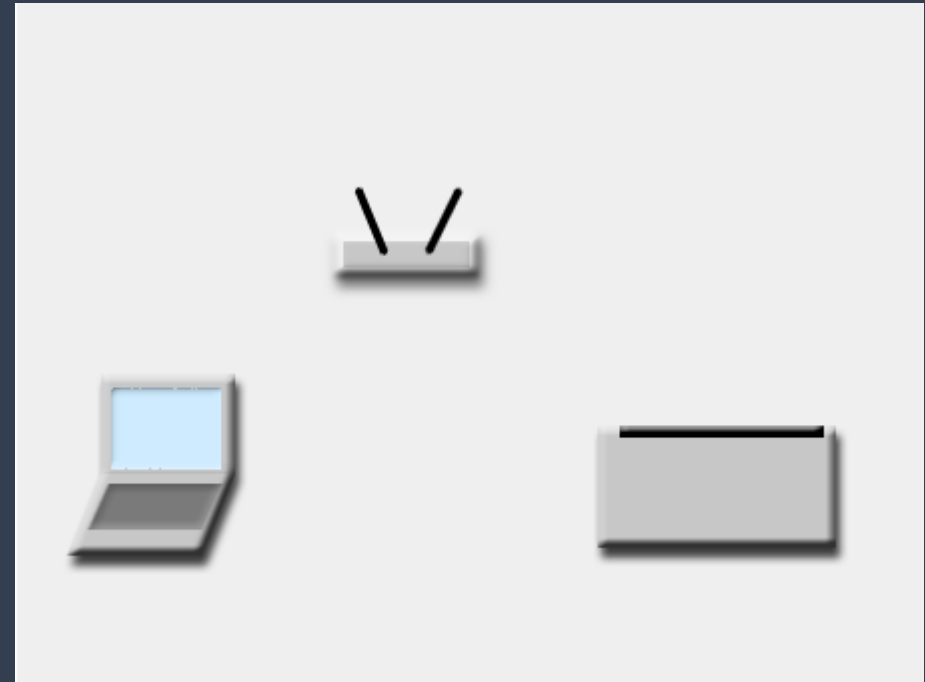
Network Terminology

Fiber optics is the technology used to transmit information as pulses of light through strands of fiber made of glass or plastic over long distances.



Network Terminology

A wireless network refers to a computer network that makes use of Radio Frequency (RF) connections between nodes in the network.



Network Terminology

Each computer on the network is called a **host**, and it can include a large number of different systems, such as desktop and laptop computers, printers, routers, switches, and even cell phones.

Network Terminology

There are two general classes of networks:

LAN (local area network): This network describes all the hosts that **communicate directly** with one another on the same network.

WAN (wide area network): This network describes a collection of LANs that are able to **communicate through a series of routers or switches**. Routers and switches have the ability to transfer network communications from one network to another.

Network Terminology

A **router** is a device that connects two or more packet-switched networks or subnetworks. It serves two primary functions: managing traffic between these networks by forwarding data packets to their intended IP addresses, and allowing multiple devices to use the same Internet connection.



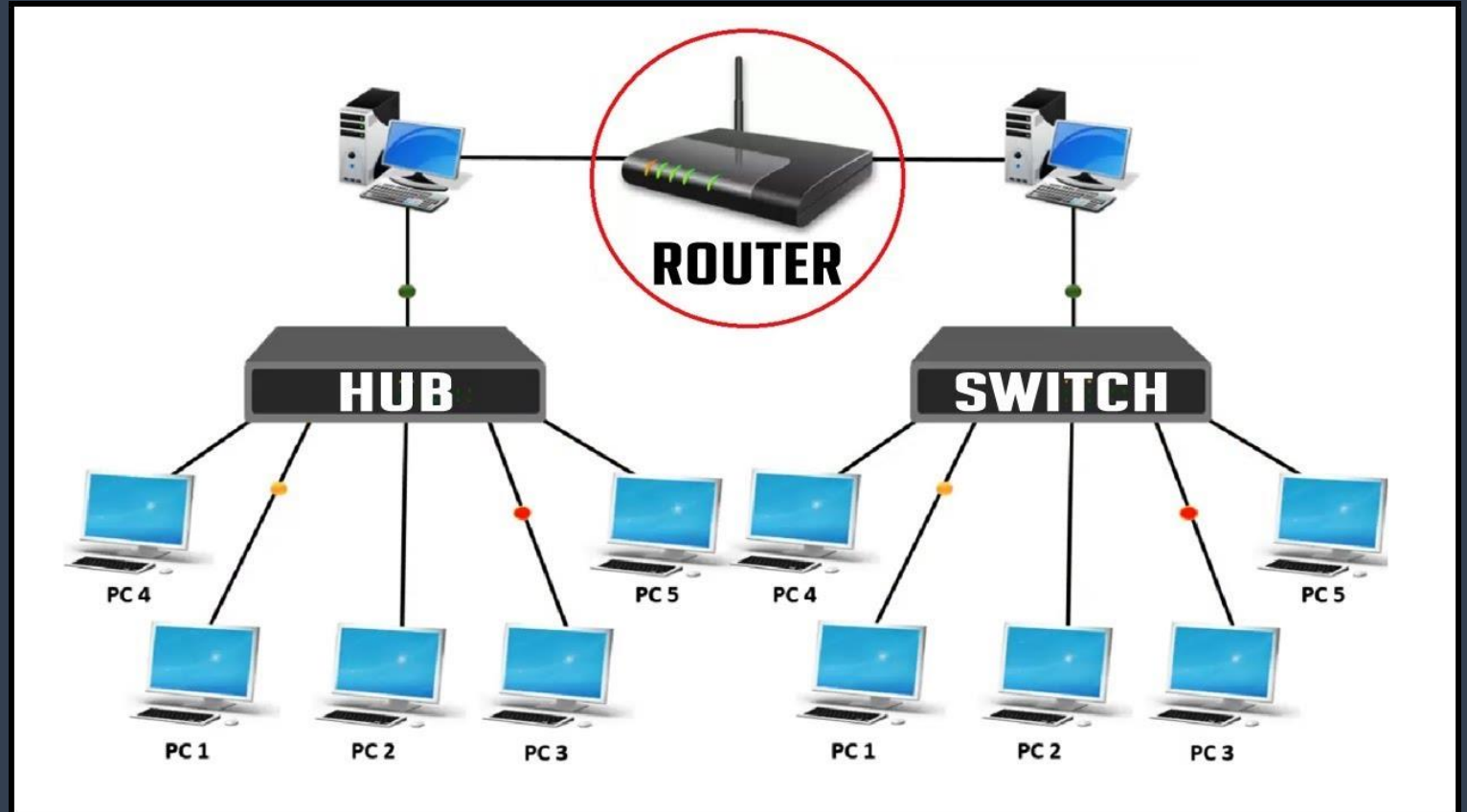
Network Terminology

Switches are key building blocks for any network. They connect multiple devices, such as computers, wireless access points, printers, and servers; on the same network within a building or campus. A switch enables connected devices to share information and talk to each other.

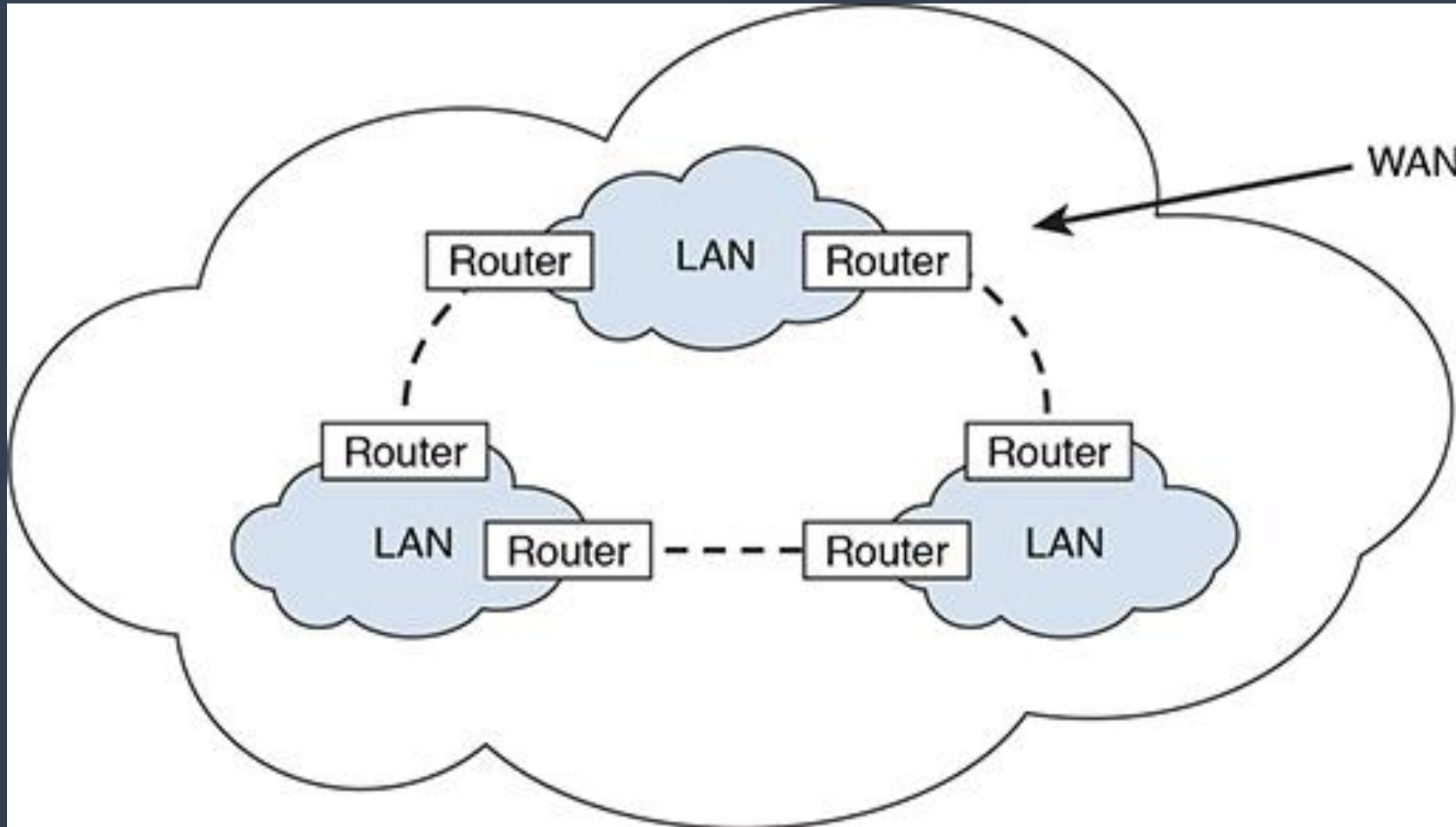


Network Terminology

A network switch forwards data packets between groups of devices in the same network, whereas a router forwards data between different networks.

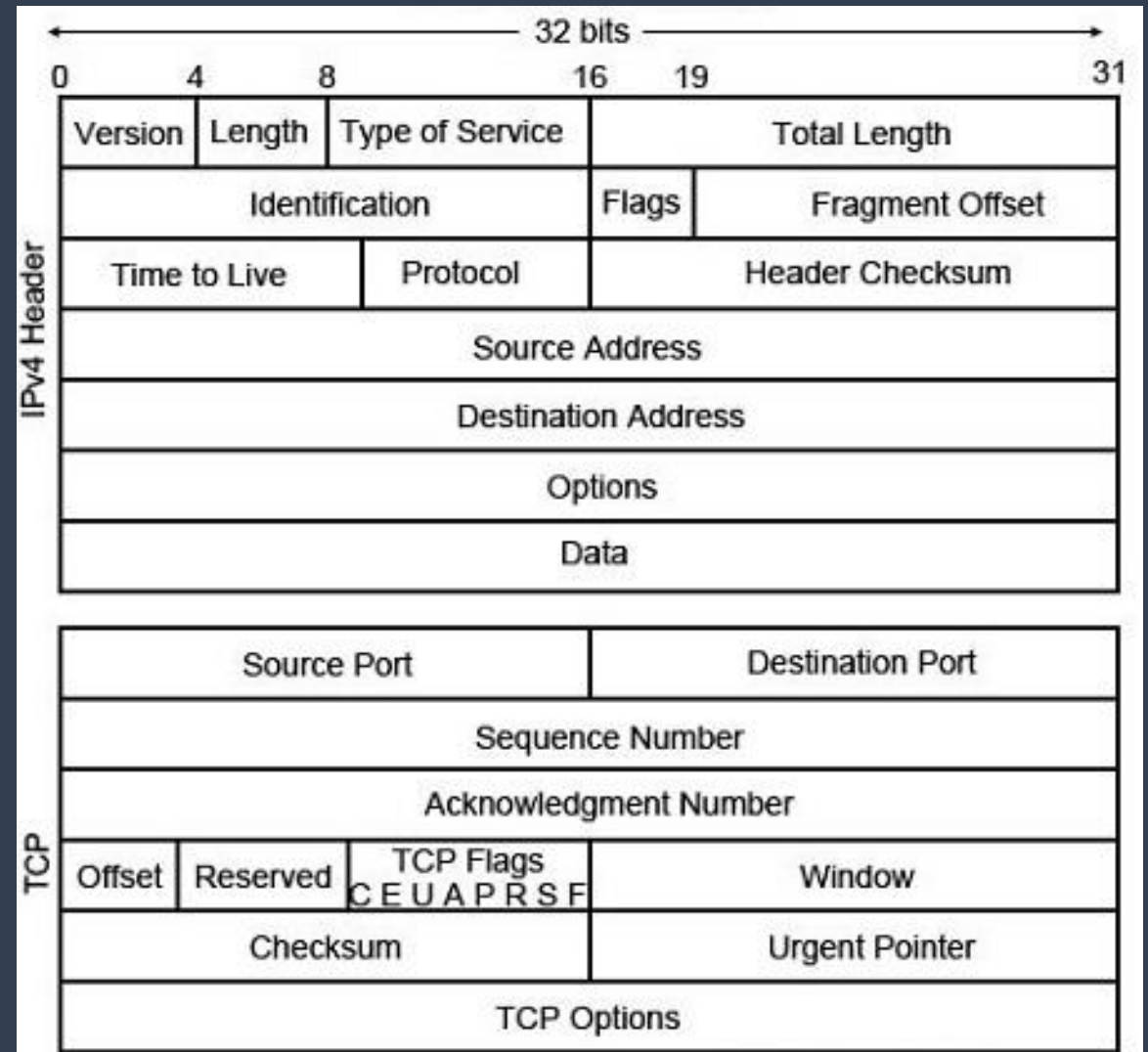


Network Terminology



Network Terminology

Data is sent across networks using a **network packet**. A network packet is a well-defined message that includes the data as well as metadata (called the **packet header**). The packet header provides information about how the network packet is to reach its destination.



Network Terminology

Included in the packet header are two pieces of information that help determine the destination of the packet: the **IP (Internet Protocol)** address of the destination host and the **port** of the destination host.

IP address A unique, numeric-based value used for network communications. Each host has a unique IP address.

The **port** is a numeric value that is associated to a service, which is a program that is running on the host that is listening for incoming messages on a specific port number.

Network Terminology

Within the network packet resides the data. The data must be formatted in a way that the receiving system understands. This is partly handled by **protocols**.

A **protocol** is a **well-defined standard for network communications between two hosts**. For example, web servers often use HTTP (Hypertext Transfer Protocol) as the means of communication between the client host and the server.

A **server** is a host on a network that offers a service; it serves something to the receiver, which is known as a client.

Network Terminology

Internet Protocol

The two different versions of IP are IPv4 and IPv6. IPv4 has been a standard on the Internet for a very long time and is slowly being replaced by IPv6.

Network Terminology

IPv4

To understand this major difference, first consider this: there are 4,294,967,296 total possible unique IPv4 addresses.

That may seem like a lot, and perhaps even more than we could possibly need, but the way in which these IP addresses are categorized and distributed resulted in a shortage of available IP addresses that was first recognized in the 1990s.

Every device that is connected to the Internet needs a unique identifier so that communication with the device.

Network Terminology

IPv6

IPv4 uses a technique called **dotted decimal notation** (also known as dot delimited), a 32-bit number divided into four octets that results in about 4.3 billion possible IP addresses.

IPv6 uses a different technique called **hexadecimal notation**, a 128-bit number.

This technique results in 340,282,366,920,938,463,463,374,607,431,768,211,456 IPv6 addresses

Network Terminology

IPv4 VS IPv6

| | IPv4 | IPv6 |
|------------------------------------|---|---|
| Source and destination addresses | 32 bits (4 bytes) in length | 128 bits (16 bytes) in length |
| IPsec support | Optional | Required |
| Address Resolution Protocol | Broadcast ARP Request frames resolve IPv4 address to link layer address | ARP Request frames replaced with multicast Neighbor Solicitation messages |
| Internet Group Management Protocol | Manages local subnet group membership | IGMP replaced with MLD messages |
| ICMP Router Discovery | Determines IPv4 address of default gateway | Replaced with ICMPv6 Router Solicitation and Router Advertisement messages |
| Broadcast addresses | Sends traffic to all nodes on a subnet | Uses a link-local scope, all-nodes multicast address instead of an IPv6 broadcast address |
| Configuration | Configured manually or through DHCP | Does not require manual configuration or DHCP |
| Resource records | Uses A resource records in DNS to map host names to IPv4 addresses | Uses AAAA resource records in DNS to map host names to IPv6 addresses |

Network Terminology

There are many reasons why the Internet has not switched completely to IPv6, but here are two of the most common reasons:

Switching an entire network from IPv4 to IPv6 is not a trivial task. The protocols are vastly different in many ways, so a lot of care and work must take place to make this switch happen smoothly and transparently.

If your organization did switch to IPv6, at some point to connect to the rest of the Internet, network communication must be converted into IPv4 because most of the Internet is still using IPv4.

Network Terminology

The concern about running out of IPv4 addresses is eliminated by the invention of an IPv4 feature called **NAT (Network Address Translation)**. With NAT, a single host (in this case, a router) can have one IPv4 address that can communicate directly on the network. The LAN that the router is connected to uses a set of IP addresses (called private IP addresses) that cannot be used directly on the Internet.

The **router translates all incoming and outgoing network packets between the Internet and the internal private network**, allowing the hosts in the internal private network indirect access to the Internet through the NAT router.

Almost **all hosts** today, including your cell phone and devices within your home (like many flat screen televisions, and even refrigerators), have **internal private IP addresses**.

With the widespread adoption of NAT, the concern about running out of IPv4 address is eliminated because only a small number of routers with live IP addresses are required to allow hundreds of hosts indirect access to the Internet.

Network Terminology

IPv4 Addresses

An IPv4 address consists of four numbers separated by a dot character (for example, 192.168.100.25). Each number represents an octet, a number that can be represented by a binary value, like so:

11000000.10101000.01100100.00011001

192 can be represented by the binary number 11000000 because each binary value represents a numeric value

Network Terminology

IPv4 Addresses

IPv4 addresses are divided into classes, and there are five classes total. These classes are defined by the first octet

Five Different Classes of IPv4 Addresses

| Class | First Octet decimal (range) | First Octet binary (range) | IP range | Subnet Mask | Hosts per Network ID | # of networks |
|---------------------------|-----------------------------|----------------------------|---------------------------|---------------|----------------------|---------------|
| Class A | 0 — 127 | 0 XXXXXXXX | 0.0.0.0-127.255.255.255 | 255.0.0.0 | $2^{24} - 2$ | 2^7 |
| Class B | 128 — 191 | 10 XXXXXX | 128.0.0.0-191.255.255.255 | 255.255.0.0 | $2^{16} - 2$ | 2^{14} |
| Class C | 192 — 223 | 110 XXXXX | 192.0.0.0-223.255.255.255 | 255.255.255.0 | $2^8 - 2$ | 2^{21} |
| Class D (Multicast) | 224 — 239 | 1110 XXXX | 224.0.0.0-239.255.255.255 | | | |
| Class E (Experimental) | 240 — 255 | 1111 XXXX | 240.0.0.0-255.255.255.255 | | | |

Network Terminology

Private IP Addresses

Private IP addresses are used with routers that utilize NAT. One network per Internet class (A, B, and C) has been set aside for private IP addresses. Any host with an IP address in one of these ranges must connect to the Internet via a router that utilizes NAT.

Most organizations use private IP addresses for most hosts, using public IP for systems like firewalls, web servers, and other hosts that need to be directly available on the Internet. As a result, you should be familiar with the ranges of IP addresses that are for private use only:

10.0.0.0-10.255.255.255

172.16.0.0-172.31.255.255

192.168.0.0-192.168.255.255

Network Terminology

Protocol Suits

IP (Internet Protocol): This protocol is responsible for delivering network packets between hosts. Its functions include routing, or sending packets from one physical network to another. Often network packets are forwarded through several routers before reaching the correct destination.

Network Terminology

Protocol Suits

TCP (Transmission Control Protocol): This protocol compliments IP (Internet Protocol), which is why you will commonly hear the term TCP/IP.

TCP is designed to ensure that the network packages arrive in a reliable and ordered manner. With TCP, data packages are connection based, which means error checking is performed to determine if packages are lost in transmission. If a packet is lost, a replacement packet is created and sent.

TCP is generally slower than UDP (see the definition of UDP) because error-checking each package requires more “work.” Also, the loss of a package requires a request to resend the packages, which can have an impact on subsequent packages. However, it is more reliable than UDP because all packages are verified. One example of the use of TCP is the downloading of a software program.

Network Terminology

Protocol Suits

UDP (User Datagram Protocol): Like TCP, this protocol complements IP (Internet Protocol). It performs a similar function to TCP; however, data packages are connectionless, which means no error checking is performed to determine if packages are lost in transmission.

As a result, it is faster than TCP because connectionless data transfer requires less “work.” It is also less reliable than TCP because of the lack of error checking.

One example of the use of UDP is live streaming of video, in which the loss of an occasional packet does not have any major impact on the overall data flow.

Network Terminology

Protocol Suits

ICMP (Internet Control Message Protocol): This protocol is used primarily to send error messages and for determining the status of network devices.

It is unlike TCP or UDP in that it is designed to send simple messages, not transfer data between devices or establish connections between devices.

One example of the use of ICMP is the ping command (see Chapter 19, “Network Configuration”), which is used to verify that a host can be contacted via the network.

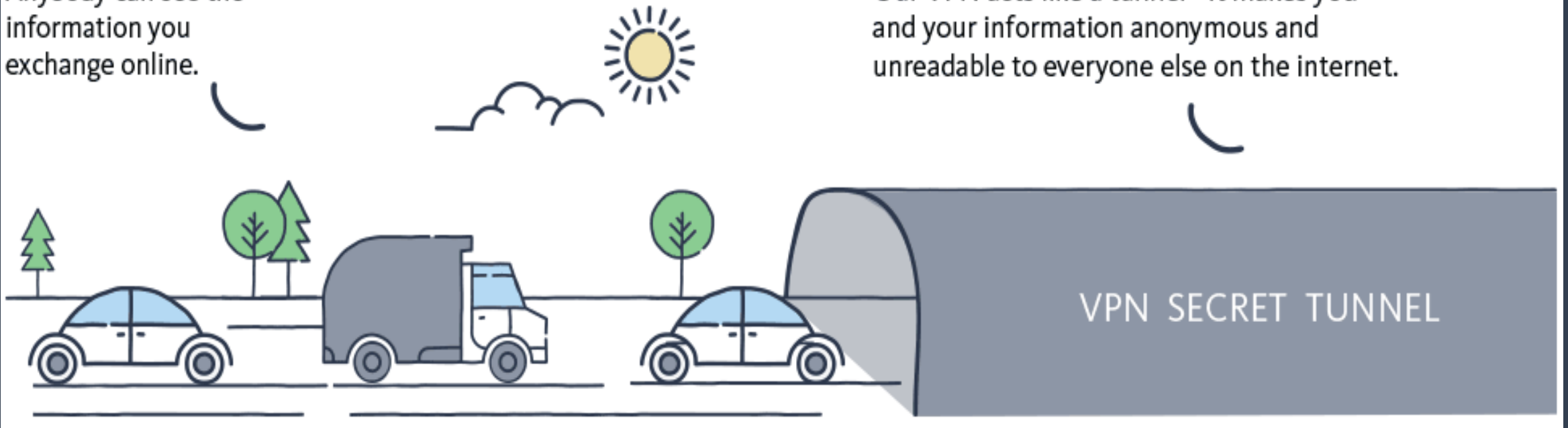
Network Terminology

VPN (Virtual Private Network)

VPN or the Virtual Private Network is a private WAN (Wide Area Network) built on the internet. It allows the creation of a secured tunnel (protected network) between different networks using the internet (public network). By using the VPN, a client can connect to the organization's network remotely.

Anybody can see the information you exchange online.

Our VPN acts like a tunnel - it makes you and your information anonymous and unreadable to everyone else on the internet.



Network Configuration

Displaying Ethernet Port Configurations

One of the commonly used commands to display network information is the **ifconfig** command.

When executed with no arguments, it lists active network devices

Network Configuration

Promiscuous mode

```
ifconfig eth0 promisc
```

Network Configuration

Network Configuration Tools

`nmcli device status`

`systemctl stop NetworkManager`

`systemctl disable NetworkManager`

Network Configuration

arp protocol

How ARP works

- When a new computer joins a LAN, it is assigned a unique IP address to use for identification and communication.
- When an incoming packet destined for a host machine on a particular LAN arrives at a gateway, the gateway asks the ARP program to find a MAC address that matches the IP address.
- A table called the ARP cache maintains a record of each IP address and its corresponding MAC address.

Network Configuration

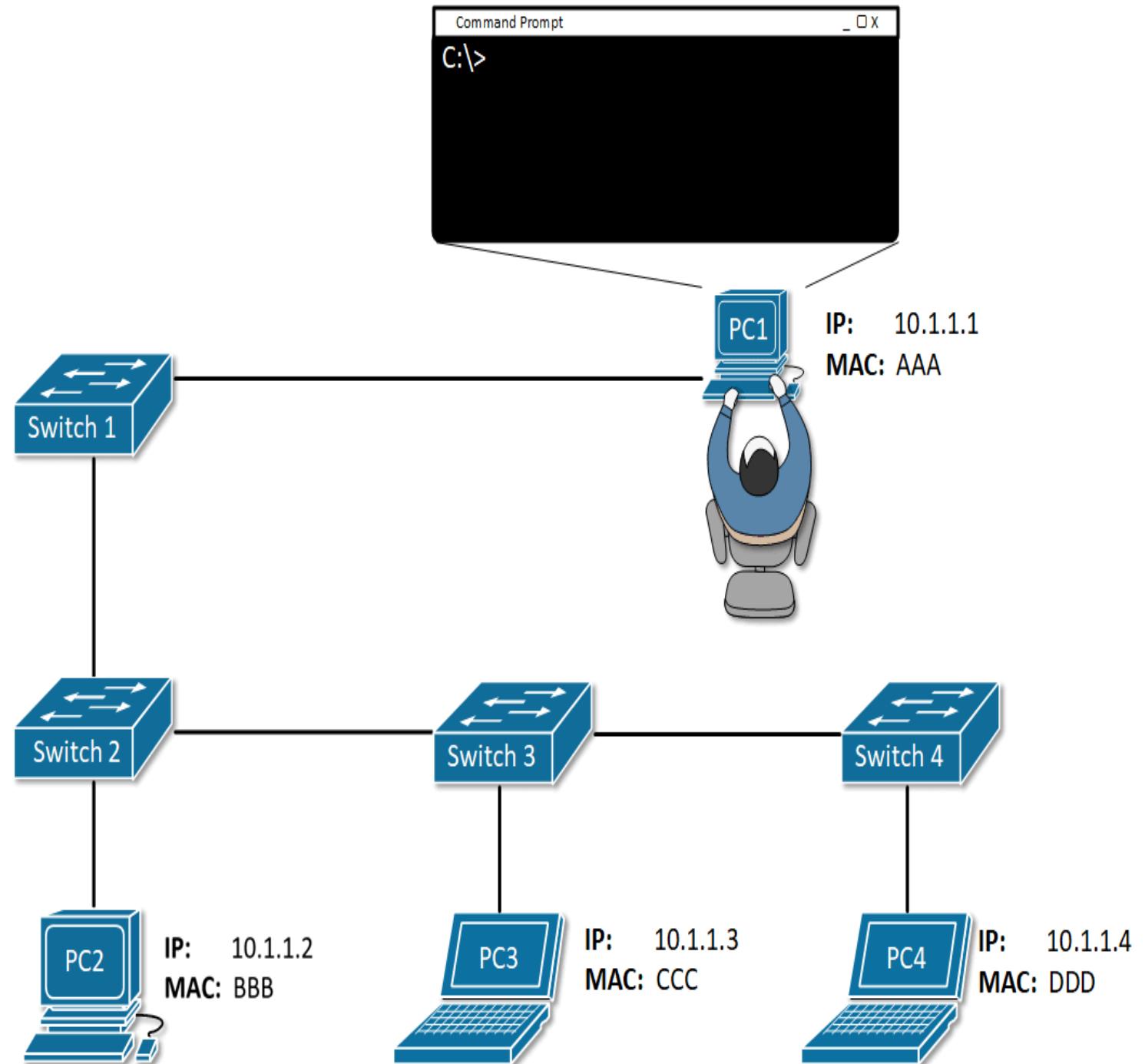
arp protocol

How ARP works

- When a new computer joins a LAN, it is assigned a unique IP address to use for identification and communication.
- When an incoming packet destined for a host machine on a particular LAN arrives at a gateway, the gateway asks the ARP program to find a MAC address that matches the IP address.
- A table called the ARP cache maintains a record of each IP address and its corresponding MAC address.

Network Configuration

arp protocol
How ARP works



Network Configuration

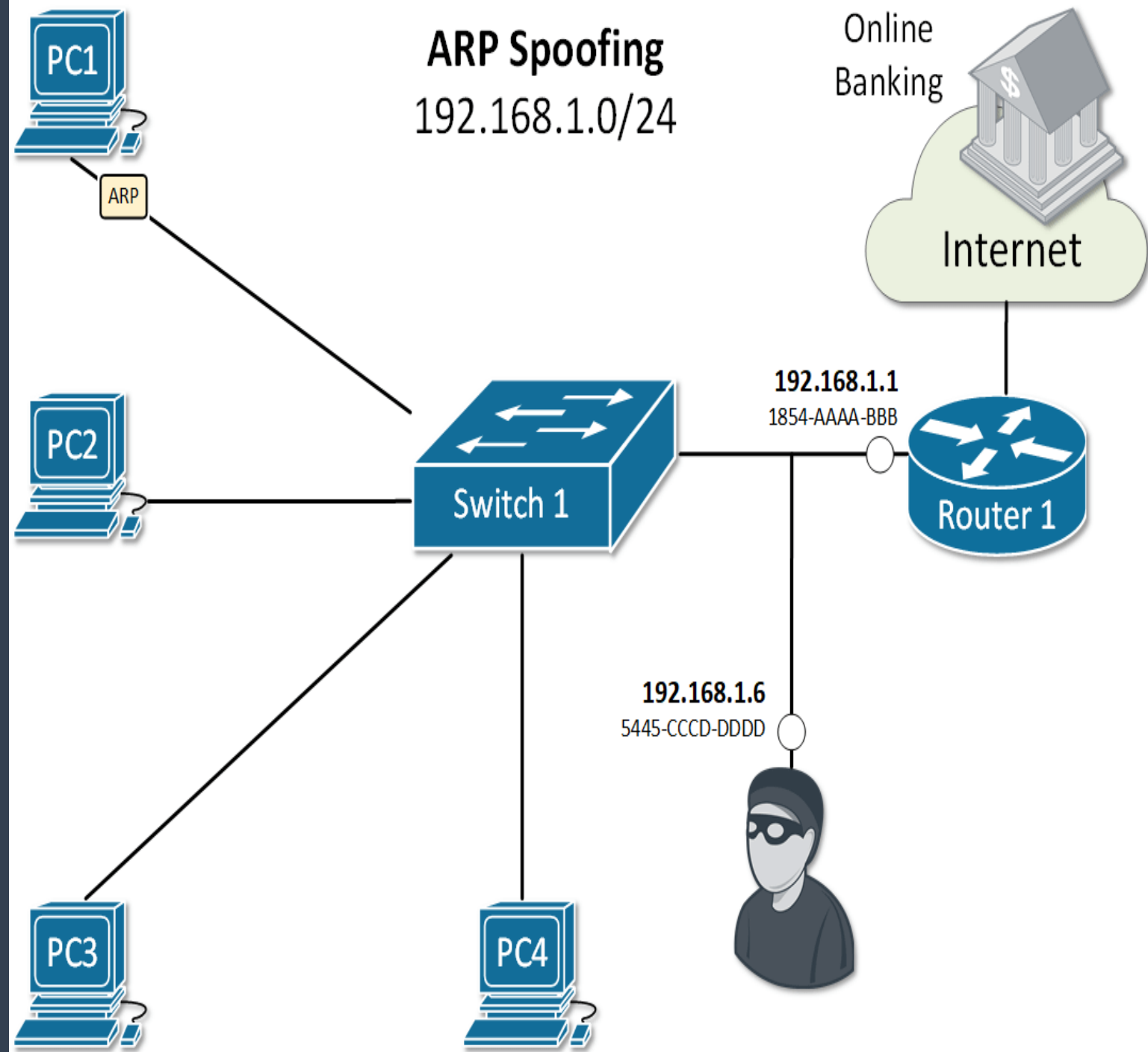
ARP Spoofing

When a rouge host in the local area network sends false ARP messages, it can associate its MAC address with the IP address of the default gateway or any other IP in the LAN.

It then can start receiving data intended for another host in the network.

Network Configuration

ARP Spoofing



Network Configuration

Route command

When a network packet is destined for a host on the local network, it is broadcast on that network and picked up by the appropriate host. When the network packet is destined for a host not on the local network, the packet needs to be passed through a gateway (also called a router). The gateway has a live IP address on both the local network and at least one other network.

The gateway is connected either to the Internet (either directly or through additional gateways) or to an internal, private network. The local system keeps track of the gateways it can directly communicate with by storing information in the routing table.

The **route** command either displays or modifies the routing table. To display the routing table, execute the route command without any arguments

Network Configuration

Route command

Suppose there was a gateway with the IP address of 192.168.1.100 connected to an internal network of 192.168.2.0/255.255.255.0. You could add this gateway by running the following command:

```
route add -net 192.168.2.0 netmask 255.255.255.0 gw 192.168.1.100
```

Network Configuration

ip Command

ip addr show #more information

ip link show #less information

Examining and Monitoring a Network

ping—Send a Special Packet to a Network Host

The most basic network command is ping. The ping command sends a special network packet called an ICMP ECHO_REQUEST to a specified host.

Most network devices receiving this packet will reply to it, allowing the network connection to be verified.

Examining and Monitoring a Network

netstat—Examine Network Settings and Statistics

The netstat program is used to examine various network settings and statistics.

Through the use of its many options, we can look at a variety of features in our network setup.

Transporting Files over a Network

- ftp—Transfer Files with the File Transfer Protocol
- lftp
- wget

Secure Communication with Remote Hosts

- SSH – Securely Log in to Remote Computers
- SSH solves the two basic problems of secure communication with a remote host. First, it authenticates that the remote host is who it says it is (thus preventing man-in-the-middle attacks), and second, it encrypts all of the communications between the local and remote hosts.
- SSH consists of two parts.
- An SSH server runs on the remote host, listening for incoming connections on port 22, while an SSH client is used on the local system to communicate with the remote server. Most Linux distributions ship an implementation of SSH called OpenSSH from the BSD project.

Shell Scripting

Shell Script

- A Shell provides you with an interface to the Unix system. It gathers input from you and executes programs based on that input. When a program finishes executing, it displays that program's output.
- Shell is an environment in which we can run our commands, programs, and shell scripts. There are different flavors of a shell, just as there are different flavors of operating systems. Each flavor of shell has its own set of recognized commands and functions.

Shell Script

- To start a BASH script, enter the following as the first line of the script file in a text editor,

```
#!/bin/bash
```
- This special sequence is called the shebang, and it tells the system to execute this code as a BASH script.

Shell Script

- Writing and executing shell script
- Using commands
- Write a program to print the name of current working directory and list all its content on the terminal screen.

Advanced Usage of echo command

- echo command is used to display string/message or variable value or command result.
- Simple syntax:
 - echo message/string
 - echo "message/string"
 - echo "message/string with some variable \$xyz"
 - echo "message/string/\$variable/\$(command)"
- Advanced usage (to execute escape characters):
 - echo -e "Message/String or variable"
 - Escape Characters:
 - \n New Line
 - \t Horizontal Tab
 - \v Vertical Tab
 - \b Backspace
 - \r Carriage Return
 - \ Escape Character etc...
 - To display message in colors.
 - echo -n "message/string/\$variable/\$(command)"

Rules to Define User Defined Variables

- Variable Name should contain only a-z or A-Z, 0-9 and _ characters.
- Variable Name length should be less than or equal to 20 characters.
- Variable Names are case sensitive. Means x and X are different.
- don't Provide space on either sides of equal symbol
 - Ex: x=4 is valid
 - x =4 or x = 4 or x= 4 are invalid
- No need to declare variable type, Automatically it will take care while executing commands or scripts.
- Use quotes for the data if data consist of spaces
- We can store the output of a command into a variable as follows:
 - anyVariable=\$(command)
 - anyVariable=`command`
- We can assign one variable value/data into another using:
 - Name="Shell Scripting"
 - NewName=\$Name
 - NewName=\${Name}

Comments for Bash Shell Scripting

- A comment is a human-readable explanation that is written in the shell script.
- Why we need comments:
 - Adding comments to your Bash scripts will save you a lot of time and effort when you look at your code in the future.
 - Comments are used to explain the code.
 - The comments also help other developers and system administrators who may need to maintain the script to understand your code and its purpose.
- Here, we have two types of comments. They are:
 - Single line Comments
 - Multi-line Comments
- Note: Comments won't execute while running or executing your script.

Shell Script

- read command : -p and -sp
- Command line argument

Shell Script

- Variables
- Read only variables : readonly variable_name
- Unset variables : unset variable_name

Shell Script

- Special Variables
- \$0 : The filename of the current script.
- \$n : These variables correspond to the arguments with which a script was invoked. Here n is a positive decimal number corresponding to the position of an argument
- \$# : The number of arguments supplied to a script.
- \$* : All the arguments are double quoted. If a script receives two arguments, \$* is equivalent to \$1 \$2.
- @\$: All the arguments are individually double quoted. If a script receives two arguments, @\$ is equivalent to \$1 \$2.
- \$? : The exit status of the last command executed.
- \$\$: The process number of the current shell. For shell scripts, this is the process ID under which they are executing.
- \$! : The process number of the last background command.

Shell Script

- Array
- Array initialization
- Array values access
- `${array_name[*]}`
- `${array_name[@]}`

Shell Script

Conditional Statement

Types of conditional statements

The following types of conditional statements can be used in bash.

- if statement
- if else statement
- if elif statement
- Nested if statement
- case statement

Shell Script

Conditional operator

- eq Returns true if two numbers are equivalent
- lt Returns true if a number is less than another number
- gt Returns true if a number is greater than another number
- == Returns true if two strings are equivalent
- != Returns true if two strings are not equivalent
- ! Returns true if the expression is false
- d Check the existence of a directory
- e Check the existence of a file
- r Check the existence of a file and read permission
- w Check the existence of a file and write permission
- x Check the existence of a file and execute permission

Shell Script

Use of simple if statement –

Syntax –

```
if [ condition ] ; then
```

```
Command(s)
```

```
Fi
```

Example –

```
#!/bin/bash
```

```
echo "Enter a number"
```

```
read n
```

```
if [ $n -lt 100 ]; then
```

```
printf "$n is less than 100\n"
```

```
fi
```

Shell Script

if statement with multiple conditions

Example –

```
#!/bin/bash
```

```
echo "Enter username"
```

```
read un
```

```
echo "Enter password"
```

```
read pw
```

```
if [[ "$un" == "admin" && "$pw" == "superuser" ]]; then
```

```
echo "Login Successful."
```

```
fi
```


Shell Script

Use of if-else statement

Syntax -

```
if [ condition ]; then
```

```
Command(s)
```

```
else
```

```
Command(s)
```

```
fi
```

Example -

```
#!/bin/bash
```

```
echo "Enter your name"
```

```
read name
```

```
if [[ $name == "Neha" || $name == 'Nil' ]]; then
```

```
echo "You have won the prize"
```

```
else
```

```
echo "Try for the next time"
```

```
fi
```

Shell Script

Use of if-elif-else statement

Syntax -

if [condition]; then

Command(s)

elif [condition]; then

Command(s)

.....

else

Command(s)

fi

Example -

```
#!/bin/bash
```

```
echo "Enter the mark"
```

```
read mark
```

```
if (( $mark >= 90 )); then
```

```
echo "Grade - A+"
```

```
elif (( $mark < 90 && $mark >= 80 )); then
```

```
echo "Grade - A"
```

```
elif (( $mark < 80 && $mark >= 70 )); then
```

```
echo "Grade - B+"
```

```
elif (( $mark < 70 && $mark >= 60 )); then
```

```
echo "Grade - C+"
```

```
else
```

```
echo "Grade - F"
```

```
fi
```

Shell Script

Use of nested if

Syntax -

if [condition]; then

Commands

if [condition]; then

Commands

fi

fi

Example -

```
#!/bin/bash
echo "Enter the sales amount"
read amount
echo "Enter the time duration"
read duration
if (( $amount >= 10000 )); then
if (( $duration <= 7 )); then
output="You will get 20% bonus"
else
output="You will get 15% bonus"
fi
else
if (( $duration <= 10 )); then
output="You will get 10% bonus"
else
output="You will get 5% bonus"
fi
fi
echo "$output"
```

Shell Script

Use of case statement

Syntax -

case in

pattern 1) commands;;

pattern n) commands;;

esac

Example -

```
#!/bin/bash
N1=$1
op=$2
N2=$3
case $op in
'+')
((Result=$N1+$N2)) ;;
'-')
((Result=$N1-$N2)) ;;
'x')
((Result=$N1*$N2)) ;;
'/')
((Result=$N1/$N2)) ;;
*)
echo "Wrong numbers of arguments"
exit 0 ;;
esac
echo "$N1 $op $N2 = $Result"
```


Shell Script

'case' statement with a range of values

```
#!/bin/bash
# Print grade based on the mark
name=$1
mark=$2
case $mark in
9[0-9]|100)
grade="A+" ;;
8[0-9])
grade="A" ;;
7[0-9])
grade="B+" ;;
6[0-9])
grade="C+" ;;
0|[0-9]|1[0-9]|2[0-9]|3[0-9]|4[0-9]|5[0-9])
grade="F" ;;
*)
echo "Invalid mark"
exit 0 ;;
esac
echo "$name obtained $grade"
```

Shell Script

Loops in Shell Scripting



```
while [ condition ] do  
  command1 command2 done  
for var in list do  
  command 1 command 2 done  
until [ conditional statement ] do  
  command1 command2 done
```

1

While Loop

2

For Loop

3

Until Loop

Shell Script

While loop

Syntax -

```
while [ condition ]
```

```
do
```

```
command1
```

```
command2
```

```
done
```

Example -

```
number = 1
```

```
while [ $number -lt 11 ]
```

```
do
```

```
echo $number
```

```
((number++))
```

```
Done
```

Shell Script

for loop

Syntax -

for var in list

do

command 1

command 2

done

Example -

for p_name in Stan Kyle Cartman

do

echo \$p_name

done

Shell Script

until loop

Syntax -

until [conditional statement]

do

command1

command2

done

Example -

number = 1

until [\$number -gt 10]

do

echo \$number

((number++))

done

Shell Script

"break" and "continue" statements

Example –

```
COUNT=0
while [ $COUNT -ge 0 ]; do
    echo "Value of COUNT is: $COUNT"
    COUNT=$((COUNT+1))
    if [ $COUNT -ge 5 ]; then
        break
    fi
done
```

```
COUNT=0
while [ $COUNT -lt 10 ]; do
    COUNT=$((COUNT+1))
    # Check if COUNT is even
    if [ $((COUNT % 2)) = 0 ]; then
        continue
    fi
    echo $COUNT
done
```