

Descripción breve del funcionamiento Del programa TCP

Programa 1: Client_TCP

```
import socket
import threading

# Configuración del cliente
server_ip = input("Ingrese la IP del servidor: ")
server_port = 60000

# Crear socket TCP
client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client_socket.connect((server_ip, server_port))

# Función para recibir mensajes del servidor
def receive_messages(client_socket):
    while True:
        try:
            message = client_socket.recv(1024).decode('utf-8')
            if message == 'exit':
                print("Conexión cerrada por el servidor.")
                client_socket.close()
                break
            print(message)
        except:
            print("Error de conexión con el servidor.")
            client_socket.close()
            break

# Crear hilo para recibir mensajes
receive_thread = threading.Thread(target=receive_messages, args=(client_socket,))
receive_thread.start()

# Enviar mensajes al servidor
while True:
    message = input()
    client_socket.send(message.encode('utf-8'))
    if message == 'exit':
        print("Conexión cerrada.")
        client_socket.close()
        break
```

Este programa implementa un cliente de chat que se conecta a un servidor utilizando sockets TCP. El cliente puede enviar y recibir mensajes, permitiendo la comunicación en tiempo real con otros clientes conectados al mismo servidor.

Programa 2: Server_TCP

```
import socket
import threading

# Configuración del servidor
server_ip = "
server_port = 60000

# Crear socket TCP
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_socket.bind((server_ip, server_port))
server_socket.listen(5)

print(f"Servidor escuchando en {server_ip}:{server_port}")

clients=[]
usernames=[]

def disconneted(client_socket):
    index = clients.index(client_socket)
    username = usernames[index]
    message = f"ChatBot: {username} se ha desconectado".encode('utf-8')
    broadcast(message,client_socket)
    clients.remove(client_socket)
    usernames.remove(username)

def broadcast(message,client_socket):
    for client in clients:
        if client != client_socket:
            client.send(message)

# Función para manejar los mensajes de los clientes
def handle_client(client_socket, client_address):
    print(f"{client_address} Cliente conectado.")
    while True:
        try:
            message = client_socket.recv(1024)
            if message.decode('utf-8') == 'exit':
                print(f"{client_address} Cliente se ha desconectado.")
                client_socket.send('exit'.encode('utf-8'))
                disconneted(client_socket)
                client_socket.close()
                break
            print(f"{client_address}:{message.decode('utf-8')}")
            broadcast(message,client_socket)

        except:

            disconneted(client_socket)

    print(f"{client_address} Error de conexión con el cliente.")
```

```
client_socket.close()
break
```

```
def receive_connections():
```

```
# Aceptar conexiones de los clientes
```

```
while True:
```

```
    client_socket, client_address = server_socket.accept()
```

```
    client_socket.send("@Username".encode('utf-8'))
```

```
    username = client_socket.recv(1024).decode('utf-8')
```

```
    usernames.append(username)
```

```
    clients.append(client_socket)
```

```
    message = f"ChatBot : {username} Se a unido al chat".encode('utf-8')
```

```
    broadcast(message, client_socket)
```

```
    client_handler = threading.Thread(target=handle_client, args=(client_socket, client_address))
```

```
    client_handler.start()
```

```
receive_connections()
```

Este programa implementa un servidor de chat que maneja múltiples conexiones de clientes utilizando hilos (threads). El servidor permite que varios clientes se conecten, intercambien mensajes y se comuniquen entre sí en tiempo real. Además, gestiona la desconexión de los clientes y transmite mensajes a todos los clientes conectados.

Instrucciones de ejecución

Para el cliente (Client_Socket):

1. Requisitos previos: Necesitas conocer la dirección IP del servidor al que deseas conectarte.

2. Ejecución:

- Guarda el código del cliente en un archivo, por ejemplo, `Client_socket.py`. Puede ser en el mismo equipo o en un equipo diferente

- Abre una terminal o línea de comandos.

- Ejecuta el script con el comando: `python client_socket.py`.

- Ingresa la IP del servidor cuando se te solicite.

- A partir de aquí, puedes enviar y recibir mensajes. Para desconectarte, escribe `exit`.

Para el servidor (Server_TCP):

1. Requisitos previos: Asegúrate de que el puerto `60000` esté disponible y no bloqueado por un firewall.

2. Ejecución:

- Guarda el código del servidor en un archivo, por ejemplo, `server_socket.py`.

- Abre una terminal o línea de comandos.

- Ejecuta el script con el comando: `python server_socket.py`.

- El servidor estará escuchando en todas las interfaces de red en el puerto `60000`.

- Los clientes podrán conectarse ingresando la IP del servidor (que debe ser la IP de la máquina donde se está ejecutando este script).

Funcionamiento detallado

Cliente (Client_Socket):

1. Configuración inicial:

- Solicita la IP del servidor.
- Configura el puerto del servidor (60000).
- Crea un socket TCP y se conecta al servidor.

2. Recepción de mensajes:

- Se crea un hilo dedicado a recibir mensajes del servidor.
- Si recibe un mensaje de "exit", cierra la conexión.

3. Envío de mensajes:

- En un bucle infinito, se leen los mensajes ingresados por el usuario.
- Se envían los mensajes al servidor.
- Si el mensaje es "exit", cierra la conexión y termina el programa.

Servidor (Server_Socket):

1. Configuración inicial:

- Configura el puerto del servidor (60000).
- Crea un socket TCP y lo vincula a la IP del servidor.
- Comienza a escuchar conexiones entrantes.

2. Gestión de conexiones:

- En un bucle infinito, acepta nuevas conexiones de clientes.
- Solicita y recibe el nombre de usuario del cliente.
- Agrega al cliente y su nombre de usuario a las listas correspondientes.
- Envía un mensaje de bienvenida a todos los clientes.

3. Manejo de clientes:

- Para cada cliente, crea un hilo dedicado a manejar su comunicación.
- Recibe mensajes del cliente y los retransmite a todos los demás clientes.
- Si recibe un mensaje de "exit", cierra la conexión del cliente y notifica a los demás clientes.

Estos programas permiten crear un sistema de chat básico donde múltiples clientes pueden comunicarse a través de un servidor central.

Descripción breve del funcionamiento Del programa UDP

```
import socket
import threading
```

```
# Configuración
BROADCAST_IP = '255.255.255.255'
PORT = 60000
```

```
# Pedir nombre de usuario
username = input("Ingrese su nombre de usuario: ")
```

```

# Crear socket UDP
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.setsockopt(socket.SOL_SOCKET, socket.SO_BROADCAST, 1)
sock.bind(('', PORT))

def receive_messages():
    while True:
        data, addr = sock.recvfrom(1024)
        message = data.decode('utf-8')
        user, msg = message.split(':', 1)
        if msg.strip() == 'exit':
            print(f"El usuario {user} ({addr[0]}) ha abandonado la conversación")
            break
        elif msg.strip() == 'nuevo':
            print(f"El usuario {user} se ha unido a la conversación")
        else:
            print(f"{user} ({addr[0]}) dice: {msg}")

def send_messages():
    while True:
        msg = input()
        if msg.strip().lower() == 'exit':
            message = f"{username}:exit"
            sock.sendto(message.encode('utf-8'), (BROADCAST_IP, PORT))
            print("Has abandonado la conversación")
            break
        else:
            message = f"{username}:{msg}"
            sock.sendto(message.encode('utf-8'), (BROADCAST_IP, PORT))

# Enviar mensaje de unión
sock.sendto(f"{username}:nuevo".encode('utf-8'), (BROADCAST_IP, PORT))

# Crear hilos para enviar y recibir mensajes
receive_thread = threading.Thread(target=receive_messages)
send_thread = threading.Thread(target=send_messages)

# Iniciar hilos
receive_thread.start()
send_thread.start()

# Esperar a que los hilos terminen
send_thread.join()
receive_thread.join()

# Cerrar socket
sock.close()

```

Este programa implementa un chat en red utilizando sockets UDP y transmisión por difusión (broadcast). Cada usuario puede enviar y recibir mensajes, permitiendo la comunicación en tiempo real con otros usuarios conectados en la misma red local.

Instrucciones de ejecución

1. Requisitos previos: Todos los usuarios deben estar conectados a la misma red local.

2. Ejecución:

- Guarda el código en un archivo, por ejemplo, ``udp_chat.py``.
- Abre una terminal o línea de comandos.
- Ejecuta el script con el comando: ``python udp_chat.py``.
- Ingresa un nombre de usuario cuando se te solicite.
- Puedes enviar y recibir mensajes. Para salir del chat, escribe ``exit``.

Funcionamiento detallado

1. Configuración inicial:

- Define la dirección IP de difusión (``BROADCAST_IP``) y el puerto (``PORT``) que se utilizarán para enviar y recibir mensajes.
- Solicita al usuario que ingrese su nombre de usuario.
- Crea un socket UDP y configura la opción de transmisión por difusión.
- Vincula el socket al puerto especificado.

2. Recepción de mensajes:

- Se crea un hilo dedicado a recibir mensajes de otros usuarios.
- Cuando se recibe un mensaje, se analiza para determinar si es un mensaje de unión, de salida o un mensaje normal.
- Si el mensaje es ``exit``, indica que un usuario ha salido de la conversación.
- Si el mensaje es ``nuevo``, indica que un nuevo usuario se ha unido a la conversación.
- Los mensajes normales se muestran junto con el nombre del usuario y la dirección IP desde donde se envió.

3. Envío de mensajes:

- Se crea un hilo dedicado a enviar mensajes.
- Los mensajes se leen desde la entrada del usuario y se envían a la dirección de difusión y el puerto especificados.
- Si el mensaje es ``exit``, se notifica a los otros usuarios y el hilo de envío se detiene.

4. Inicio y gestión de hilos:

- Se envía un mensaje de unión al iniciar el programa para notificar a los demás usuarios.
- Se crean e inician dos hilos: uno para recibir mensajes y otro para enviar mensajes.
- El programa espera a que ambos hilos terminen antes de cerrar el socket.

Este programa permite que múltiples usuarios en la misma red local puedan comunicarse entre sí en tiempo real mediante el uso de sockets UDP y transmisión por difusión.

Actividad 4: Escaneo de puertos

4.4 (actividad con resultados a agregar en el informe) Realice un escaneo buscando host al azar usando `nmap -sS -sU --top-ports 5 -iR 100` (siendo el número entre paréntesis la cantidad de IPs a generar aleatoriamente). Busque algún host con el puerto ssh o http (no https) abierto (debe decir “open”, no debe decir “filtered” o “closed”). Si no encuentra ningún hosts, intente nuevamente con un número de IP aleatorias mayor. Seleccione un objetivo y detecte su ubicación geográfica (en el trabajo práctico N°3 se presentaron varios mecanismos para localizar geográficamente

una IP).

Ejercicio 4.4: Análisis de Todos los Puertos UDP desde el 1 al 60000

Objetivo:

Realizar un análisis exhaustivo de todos los puertos UDP en una computadora de la red local.

```
3 de jun 2015
william_3@ubuntu: ~
You requested a scan type which requires root privileges.
QUITTING!
william_3@ubuntu:~$ sudo nmap -sS -sU --top-ports 5 -iR 500
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-06-03 15:18 PDT
Nmap scan report for ec2-3-248-131-159.eu-west-1.compute.amazonaws.com (3.248.131.159)
Host is up (0.25s latency).

PORT      STATE      SERVICE
21/tcp    filtered  ftp
22/tcp    filtered  ssh
23/tcp    filtered  telnet
80/tcp    filtered  http
443/tcp    filtered  https
123/udp   open|filtered ntp
137/udp   open|filtered netbios-ns
138/udp   open|filtered netbios-dgm
161/udp   open|filtered snmp
631/udp   open|filtered ipp

Nmap scan report for 175.246.81.66
Host is up (0.32s latency).

PORT      STATE      SERVICE
21/tcp    closed    ftp
22/tcp    closed    ssh
23/tcp    closed    telnet
80/tcp    filtered  http
443/tcp    closed    https
123/udp   open|filtered ntp
137/udp   open|filtered netbios-ns
138/udp   open|filtered netbios-dgm
161/udp   open|filtered snmp
631/udp   open|filtered ipp

Nmap scan report for 77.244.176.242
Host is up (0.25s latency).

PORT      STATE      SERVICE
21/tcp    filtered  ftp
22/tcp    open      ssh
23/tcp    filtered  telnet
80/tcp    filtered  http
443/tcp    open      https
123/udp   open|filtered ntp
137/udp   open|filtered netbios-ns
138/udp   open|filtered netbios-dgm
```

```
138/udp   open|filtered netbios-dgm
161/udp   open|filtered snmp
631/udp   open|filtered ipp

Nmap scan report for 77.244.176.242
Host is up (0.25s latency).

PORT      STATE      SERVICE
21/tcp    filtered  ftp
22/tcp    open      ssh
23/tcp    filtered  telnet
80/tcp    filtered  http
443/tcp    open      https
123/udp   open|filtered ntp
137/udp   open|filtered netbios-ns
138/udp   open|filtered netbios-dgm
161/udp   open|filtered snmp
631/udp   open|filtered ipp

Nmap scan report for 105.103.75.227
Host is up (0.29s latency).

PORT      STATE      SERVICE
21/tcp    filtered  ftp
22/tcp    filtered  ssh
23/tcp    filtered  telnet
80/tcp    filtered  http
443/tcp    filtered  https
123/udp   open|filtered ntp
137/udp   open|filtered netbios-ns
138/udp   open|filtered netbios-dgm
161/udp   closed    snmp
631/udp   open|filtered ipp

Nmap done: 500 IP addresses (4 hosts up) scanned in 448.57 seconds
william_3@ubuntu:~$ whois 77.244.176.242
No se ha encontrado la orden «whois», pero se puede instalar con:
sudo apt install whois
william_3@ubuntu:~$ sudo apt install whois
[sudo] contraseña para william_3:
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes NUEVOS:
whois
0 actualizados, 1 nuevos se instalarán, 0 para eliminar y 0 no actualizados
```

Con Whois es un comando de linux que se utiliza para buscar en la base de datos que almacena la información del registrante de un dominio de internet, la dirección IP asignada a un recurso de red y más

Si miramos en la parte de address que dice IT (Italia) es el origen de donde proviene ala direccion IP

```
3 de jun 20:14
william_3@ubuntu: ~
Procesando disparadores para nan-db (2.11.2-3) ...
william_3@ubuntu:~$ whois 77.244.176.242
% This is the RIPE Database query service.
% The objects are in RPSL format.
%
% The RIPE Database is subject to Terms and Conditions.
% See https://apps.db.ripe.net/docs/HTML-Terms-And-Conditions
%
% Note: this output has been filtered.
%       To receive output for a database update, use the "-B" flag.
%
% Information related to '77.244.176.240 - 77.244.176.247'
% Abuse contact for '77.244.176.240 - 77.244.176.247' is 'abuse@planetel.it'

inetnum:        77.244.176.240 - 77.244.176.247
netname:        PLANETEL-TEXCENE
descr:          PLANETEL-CUSTOMERS-RDSL
country:        IT
admin-c:        PLAN2-RIPE
tech-c:         PLAN2-RIPE
status:         ASSIGNED PA
mnt-by:         MNT-PLANETEL
created:        2008-07-21T14:09:24Z
last-modified:  2008-07-21T14:09:24Z
source:         RIPE

role:            Planetel LIR Maintainers
address:         VIA BOFFALORA 4
address:         24048 TREVIOLO
address:         IT
phone:           +39 035 204085
fax-no:          +39 035 204065
admin-c:         GE201-RIPE
abuse-mailbox:   abuse@planetel.it
remarks:         Please send abuse notifications to abuse@planetel.it
tech-c:          GE201-RIPE
tech-c:          GE201-RIPE
tech-c:          GE201-RIPE
tech-c:          GE201-RIPE
tech-c:          GE201-RIPE
nic-hdl:        PLAN2-RIPE
mnt-by:         MNT-PLANETEL
created:        2005-10-03T13:38:18Z
last-modified:  2019-03-06T15:38:02Z
source:         RIPE # Filtered

william_3@ubuntu:~$
```

```
3 de jun 20:14
william_3@ubuntu: ~
inetnum:        77.244.176.240 - 77.244.176.247
netname:        PLANETEL-TEXCENE
descr:          PLANETEL-CUSTOMERS-RDSL
country:        IT
admin-c:        PLAN2-RIPE
tech-c:         PLAN2-RIPE
status:         ASSIGNED PA
mnt-by:         MNT-PLANETEL
created:        2008-07-21T14:09:24Z
last-modified:  2008-07-21T14:09:24Z
source:         RIPE

role:            Planetel LIR Maintainers
address:         VIA BOFFALORA 4
address:         24048 TREVIOLO
address:         IT
phone:           +39 035 204085
fax-no:          +39 035 204065
admin-c:         GE201-RIPE
abuse-mailbox:   abuse@planetel.it
remarks:         Please send abuse notifications to abuse@planetel.it
tech-c:          GE201-RIPE
tech-c:          GE201-RIPE
tech-c:          GE201-RIPE
tech-c:          GE201-RIPE
tech-c:          GE201-RIPE
nic-hdl:        PLAN2-RIPE
mnt-by:         MNT-PLANETEL
created:        2005-10-03T13:38:18Z
last-modified:  2019-03-06T15:38:02Z
source:         RIPE # Filtered

% Information related to '77.244.176.0/24AS47217'

route:          77.244.176.0/24
descr:          PLANETEL-SPA
origin:         AS47217
mnt-by:         MNT-PLANETEL
created:        2023-12-27T13:52:50Z
last-modified:  2023-12-27T13:52:50Z
source:         RIPE

% This query was served by the RIPE Database Query Service version 1.112 (BUSA)

william_3@ubuntu:~$
```

4.5 (actividad con resultados a agregar en el informe) Realice un escaneo a todas las

IPs de Corea del Norte. Indique:

- El número de IPs encontradas.
- Alguna IP con servicio ssh.
- Alguna IP con un servicio web (http o https).

En sistemas Linux, con “cat /etc/services | grep servicio” puede saber el número de puerto de algún servicio (por ejemplo, “cat /etc/services | grep ssh” indicará el puerto utilizado por el servicio ssh).

Para que una máquina ofrezca algún servicio, debe tener un proceso asociado al puerto esperando peticiones. Los procesos usualmente se llaman como el servicio

ofrecido más la letra “d” (por ejemplo, un router ospf ejecutará un proceso llamado `Licenciatura` en Ciencias de la Computación `ospfd` o demonio `ospf` asociado a algún puerto). Nota: este escaneo puede demorar varios minutos.

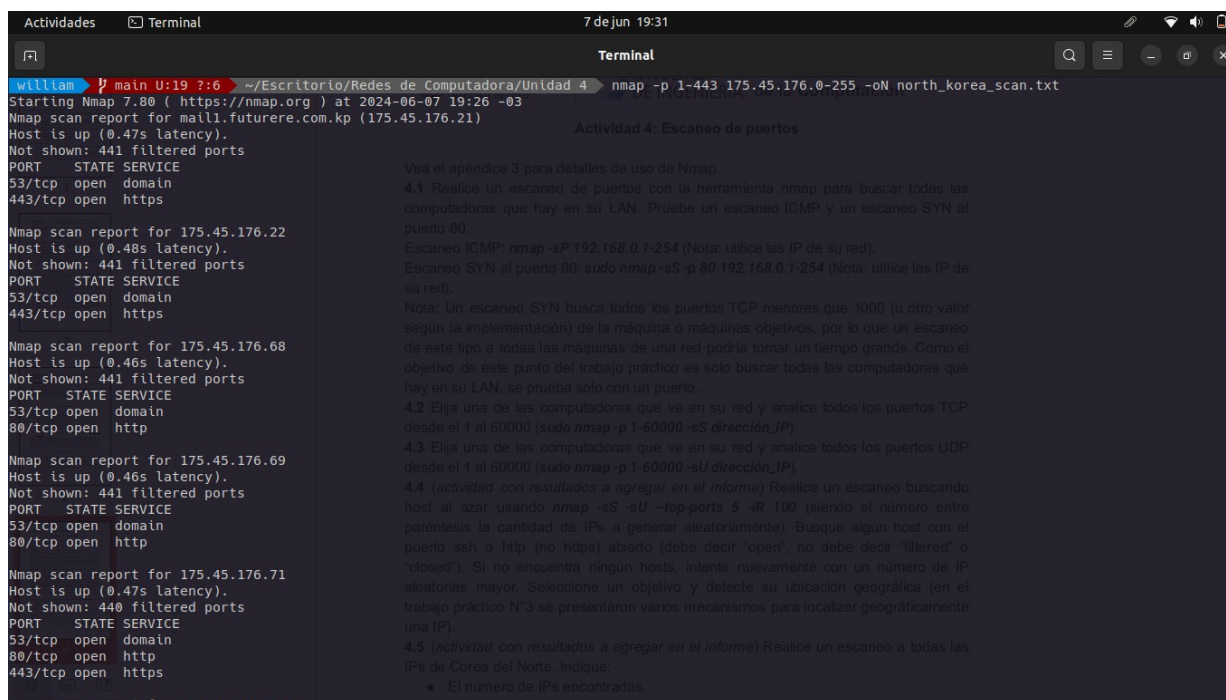
Paso 1: Identificar el Rango de IPs de Corea del Norte

El rango de IPs asignadas a Corea del Norte es limitado y puede ser encontrado a través de diversas bases de datos de IPs. Uno de los rangos conocidos es `175.45.176.0/22`.

Paso 2: Realizar el Escaneo con `nmap`

Usaremos `nmap` para escanear el rango de IPs y buscar servicios específicos. Los comandos que se utilizarán son:

1. Escaneo de todas las IPs en el rango:

A screenshot of a terminal window titled "Terminal" showing the output of an nmap scan. The command executed is `nmap -p 1-443 175.45.176.0-255 -oN north_korea_scan.txt`. The output shows four scan reports for IP addresses 175.45.176.22, 175.45.176.68, 175.45.176.69, and 175.45.176.71. Each report indicates the host is up and lists open ports with their corresponding services. For example, 175.45.176.22 has ports 53/tcp (domain) and 443/tcp (https) open. The terminal also shows a sidebar with "Actividad 4: Escaneo de puertos" and a list of tasks related to nmap scanning.

```
nmap -p 1-443 175.45.176.0/22 -oN north_korea_scan.txt
```

Este comando escaneará todos los puertos (del 1 al 65535) en el rango de IPs y guardará los resultados en un archivo llamado `north_korea_scan.txt`.

Filtrado de resultados para encontrar servicios específicos:

Para encontrar IPs con servicio SSH (puerto 22):

```
grep -B 4 "22/tcp open" north_korea_scan.txt
```

Para encontrar IPs con servicio web HTTP (puerto 80) o HTTPS (puerto 443):

```
grep -B 4 "80/tcp open" north_korea_scan.txt
```

```
grep -B 4 "443/tcp open" north_korea_scan.txt
```

```
1 # Nmap 7.80 scan initiated Fri Jun 7 19:26:22 2024 as: nmap -p 1-443 -oN north_korea_scan.txt 175.45.176.0-255
2 Nmap scan report for mail1.futurere.com.kp (175.45.176.21)
3 Host is up (0.47s latency).
4 Not shown: 441 filtered ports
5 PORT      STATE SERVICE
6 53/tcp    open  domain
7 443/tcp   open  https
8
9 Nmap scan report for 175.45.176.22
10 Host is up (0.48s latency).
11 Not shown: 441 filtered ports
12 PORT      STATE SERVICE
13 53/tcp    open  domain
14 443/tcp   open  https
15
16 Nmap scan report for 175.45.176.68
17 Host is up (0.46s latency).
18 Not shown: 441 filtered ports
19 PORT      STATE SERVICE
20 53/tcp    open  domain
21 80/tcp    open  http
22
23 Nmap scan report for 175.45.176.69
24 Host is up (0.46s latency).
25 Not shown: 441 filtered ports
26 PORT      STATE SERVICE
27 53/tcp    open  domain
28 80/tcp    open  http
29
30 Nmap scan report for 175.45.176.71
31 Host is up (0.47s latency).
32 Not shown: 440 filtered ports
33 PORT      STATE SERVICE
34 53/tcp    open  domain
35 80/tcp    open  http
36 443/tcp   open  https
37
```

Paso 3: Ejecución y Resultados

Después de ejecutar estos comandos, analizamos los resultados para obtener la información solicitada:

1. Número de IPs encontradas: Contamos las líneas que contienen direcciones IP en el archivo de resultados.

```
grep -oP '(?<=Nmap scan report for)[0-9]+\.[0-9]+\.[0-9]+\.[0-9]+' north_korea_scan.txt | wc -l
```

2. Alguna IP con servicio SSH: Filtramos las líneas correspondientes.

```
grep -B 4 "22/tcp open" north_korea_scan.txt | grep "Nmap scan report for"
```

3. Alguna IP con un servicio web (HTTP o HTTPS): Filtramos las líneas correspondientes.

```
grep -B 4 "80/tcp open" north_korea_scan.txt | grep "Nmap scan report for"
```

```
grep -B 4 "443/tcp open" north_korea_scan.txt | grep "Nmap scan report for"
```

```
53/tcp open domain
80/tcp open http
443/tcp open https

Nmap scan report for 175.45.176.85
Host is up (0.49s latency).
Not shown: 440 filtered ports
PORT      STATE SERVICE
53/tcp    open  domain
80/tcp    open  http
443/tcp   open  https

Nmap scan report for 175.45.176.91
Host is up (0.45s latency).
Not shown: 440 filtered ports
PORT      STATE SERVICE
53/tcp    open  domain
80/tcp    open  http
443/tcp   open  https

Nmap done: 256 IP addresses (12 hosts up) scanned in 254.87 seconds
william@main:~$ grep -oP '(?<=Nmap scan report for)[0-9]+\.[0-9]+\.[0-9]+\.[0-9]+' north_korea_scan.txt | wc -l
11
william@main:~$ grep -B 4 "22/tcp open" north_korea_scan.txt | grep "Nmap scan report for"
william@main:~$ grep -B 4 "80/tcp open" north_korea_scan.txt | grep "Nmap scan report for"
william@main:~$ grep -B 4 "443/tcp open" north_korea_scan.txt | grep "Nmap scan report for"
```

para los servicios HTTP en el puerto 80 y 443 se encontraron 11
pero para el puerto 22 no se encontraron disponibles ssh abiertos