UNIVERSIDAD SAN CARLOS DE GUATEMALA

CENTRO UNIVERSITARIO DE OCCIDENTE

DIVISIÓN CIENCIAS DE LA INGENIERÍA

CARRERA DE INGENIERÍA CIENCIAS Y SISTEMAS

LABORATORIO DE REDES DE COMPUTADORAS 2

“OCTAVO SEMESTRE”

Ingeniero: FRANCISCO ROJAS

Estudiante: Mario Moisés Ramírez Tobar - 201830007

Trabajo: “Primer Proyecto”

Fecha de entrega: 21 de septiembre de 2,021

INTRODUCCIÓN

En el siguiente trabajo vamos a hablar sobre lo que es ancho de banda, lo que es iptables, y un programador de tareas, vamos a estar profundizando en estos tres temas, donde por ejemplo el ancho de banda vamos a ver errores comunes al referirnos a este mismo, cómo pensar que es igual a velocidad, esto para entrar en contexto, pero también se van a ver herramientas para el manejo del ancho de banda, como lo son las Colas HTB, y como funcionan, como pequeñas referencias de cómo funciona esta herramienta.

También vamos a hablar sobre protocolos, que es donde entra la antes mencionada herramienta iptables, vamos a ver los tres tipos de protocolos a usar, para que se usan, como se usan y cuáles son sus beneficios, como también cómo usar la herramienta iptables, para la configuración de protocolos a seguir, donde dependiendo del protocolo vamos a tener especificaciones de puertos o no, donde se va a ir viendo su funcionamiento, su escritura, y cómo esto crea un ambiente completo con lo que son las colas HTB, generando así una práctica muy interesante en sus funcionalidades, y su practicidad en llevarlo en un ambiente de trabajo, y hasta personal.

Pero esto no acaba con estos dos grandes temas, si no se va a ver lo que es un programador de tareas, ya que necesitamos como tal programar horarios donde estas tareas se realicen, y terminen de ejecutarse, a una hora determinado, siendo como posibilidad crontab, pero al no funcionar correctamente en este caso particular, se usó at, que como veremos más adelante, es una herramienta super practica, facil de usar, y que no dio ningún tipo de fallo, donde programa una tarea, a la hora y minuto deseado, dando muchas más opciones, y como solo hay comandos para ejecutar una vez en una hora dada, y el proyecto necesitaba que en la hora dada, fuera repitiendo día a día, se realizó una solución que se verá más adelante, que resultó ser muy interesante.

OBJETIVOS

Objetivos Generales

- Limitar el Ancho de banda por tiempo a usuarios que utilicen el internet
- Asignar protocolos con puertos a cada usuario por tiempo

Objetivos Específicos

- Limitar el ancho de banda a tres usuarios
- Limitar en intervalos de tiempo este ancho de banda
- Asignar protocolos, como sus puertos a un usuario
- Limitar en intervalos de tiempo el uso de protocolos, como sus puertos a un usuario
- Utilizar debian como servidor encargado de las configuraciones.

MARCO TEÓRICO

Ancho de banda

El ancho de banda por lo general se confunde con la velocidad de Internet cuando en realidad es el volumen de información que se puede enviar a través de una conexión en una cantidad medida de tiempo, calculado en megabits por segundo (Mbps).

El ancho de banda denota la capacidad de transmisión de una conexión y es un factor importante al determinar la calidad y la velocidad de una red.

Hay varias formas diferentes de medir el ancho de banda. Algunas se utilizan para calcular el flujo de datos en un momento dado, mientras que otras miden el flujo máximo, el flujo típico o lo que se considera un buen flujo.

El ancho de banda también es un concepto clave en muchas otras áreas tecnológicas. Por ejemplo, en el procesamiento de señales se usa para describir la diferencia entre las

frecuencias superior e inferior en una transmisión como una señal de radio, y se mide típicamente en hercios (Hz).

Originalmente, el ancho de banda se medía en bits por segundo y se expresaba como bps. Sin embargo, hoy en día las redes suelen tener un ancho de banda mucho mayor que el que se puede expresar cómodamente utilizando unidades tan pequeñas. Actualmente, es común ver números mayores que se denotan con prefijos métricos como Mbps (megabits por segundo), Gbps (gigabits por segundo) o Tbps (terabits por segundo).

Después de terabit existe el petabit, el extabit, el zettabit y el yottabyte, y cada uno representa una potencia adicional de 10.

El ancho de banda también se puede expresar en bytes por segundo, lo que generalmente se denota con una B mayúscula. Por ejemplo, 10 megabytes por segundo se expresarán como 10 MB/s o 10 MBps.

Un byte son ocho bits.

De ese modo, 10 MB/s = 80 Mb/s.

Se pueden usar los mismos prefijos tanto con bytes como con bits. Por lo tanto, 1 TB/s es un terabyte por segundo.

Velocidad vs Ancho de banda

El ancho de banda es la cantidad de información que recibes cada segundo, mientras que la velocidad es cuán rápido esa información se recibe o descarga. Comprobémoslo con llenar una tina Si el grifo de la tina tiene una abertura ancha, más agua puede correr a una velocidad mayor que si el grifo fuera más angosto. Piensa en el agua como el ancho de banda y la tasa en la cual corre el agua como la velocidad.

Colas HTB

HTB está pensado como un reemplazo más comprensible e intuitivo para la qdisc CBQ en Linux. Tanto CBQ como HTB le ayudan a controlar el uso del ancho de banda de salida en un enlace determinado. Ambos te permiten utilizar un enlace físico para simular varios enlaces más lentos y enviar diferentes tipos de tráfico en diferentes enlaces simulados. En ambos casos, debe especificar cómo dividir el enlace físico, en enlaces simulados y cómo decidir a qué enlace simulado utilizar para enviar un paquete determinado.

A diferencia de CBQ, HTB da forma al tráfico según el filtro de depósito de tokens. Algoritmo que no depende de las características de la interfaz y por lo que no es necesario conocer el ancho de banda subyacente de la salida interfaz.

Clasificación

Dentro de una instancia de HTB pueden existir muchas clases. Cada una de estas clases contiene otra qdisc, por defecto tc-pfifo (8).

Al poner en cola un paquete, HTB comienza en la raíz y usa varios métodos para determinar qué clase debe recibir los datos.

En ausencia de opciones de configuración poco comunes, el proceso es bastante sencillo. En cada nodo buscamos una instrucción y luego vamos a la clase a la que nos refiere la instrucción. Si la clase encontrada es un nodo de hoja estéril (sin hijos), colocamos el paquete en cola allí. Si aún no es un nodo hoja, lo volvemos a hacer desde ese nodo.

Se realizan las siguientes acciones, en orden en cada nodo que visitamos, hasta que uno nos envía a otro nodo, o termina el proceso.

1. Consultar filtros adjuntos a la clase. Si se envía a un nodo de hoja, hemos terminado. De lo contrario, reinicie.
2. Si ninguno de los anteriores regresó con una instrucción, ponga en cola en este nodo.

Este algoritmo se asegura de que un paquete siempre termine en algún lugar, incluso mientras está ocupado construyendo su configuración.

Distribución

En muchas ocasiones los administradores de red experimentan cuellos de botella prolongados debido al mal uso que algunos usuarios dan del acceso a internet. Gran parte de esos abusos son llevados a cabo por las descargas que se realizan ya sea música, vídeo o software a través de programas P2P. Si bien existen alternativas para limitar el ancho de banda como los Delay Pools de Squid, pero solo funciona para limitar el ancho de banda de aquellas descargas vía HTTP y para el FTP.

Para implementar una solución que permita distribuir al ancho de banda y la tasa de transferencia de una forma más justa que incluyan no solo a HTTP y FTP sino inclusive a los programas P2P existen herramientas como CBQ y HTB que realizan la tarea.

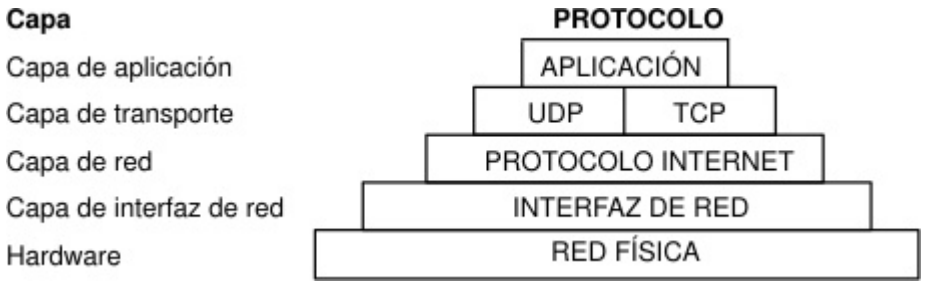
En este documento se presenta una guía básica y funcional para la correcta implementación de HTB+Iptables para una distribución justa del ancho de banda.

Protocolos

TCP

Los protocolos son conjuntos de normas para formatos de mensaje y procedimientos que permiten a las máquinas y los programas de aplicación intercambiar información. Cada máquina implicada en la comunicación debe seguir estas normas para que el sistema principal de recepción pueda interpretar el mensaje. El *conjunto* de protocolos TCP/IP puede interpretarse en términos de capas (o niveles).

Esta figura muestra las capas del protocolo TCP/IP. Empezando por la parte superior son: capa de aplicación, capa de transporte, capa de red, capa de interfaz de red y hardware.



TCP/IP define cuidadosamente cómo se mueve la información desde el remitente hasta el destinatario. En primer lugar, los programas de aplicación envían mensajes o corrientes de datos a uno de los protocolos de la capa de transporte de Internet, UDP (User Datagram Protocol) o TCP (Transmission Control Protocolo). Estos protocolos reciben los datos de la aplicación, los dividen en partes más pequeñas llamadas *paquetes*, añaden una dirección de destino y, a continuación, pasan los paquetes a la siguiente capa de protocolo, la capa de red de Internet.

La capa de red de Internet pone el paquete en un datagrama de IP (Internet Protocol), pone la cabecera y la cola de datagrama, decide dónde enviar el datagrama (directamente a un destino o a una pasarela) y pasa el datagrama a la capa de interfaz de red.

La capa de interfaz de red acepta los datagramas IP y los transmite como *tramas* a través de un hardware de red específico, por ejemplo redes Ethernet o de Red en anillo.

UDP

El protocolo de datagramas de usuario (en inglés: User Datagram Protocol o UDP) es un protocolo del nivel de transporte basado en el intercambio de datagramas (Encapsulado de capa 4 o de Transporte del Modelo OSI). Permite el envío de datagramas a través de la red sin que se haya establecido previamente una conexión, ya que el propio

datagrama incorpora suficiente información de direccionamiento en su cabecera. Tampoco tiene confirmación ni control de flujo, por lo que los paquetes pueden adelantarse unos a otros; y tampoco se sabe si ha llegado correctamente, ya que no hay confirmación de entrega o recepción. Su uso principal es para protocolos como DHCP, BOOTP, DNS y demás protocolos en los que el intercambio de paquetes de la conexión/desconexión son mayores, o no son rentables con respecto a la información transmitida, así como para la transmisión de audio y vídeo en tiempo real, donde no es posible realizar retransmisiones por los estrictos requisitos de retardo que se tiene en estos casos.

Las características principales de este protocolo son:

1. Trabaja sin conexión, es decir que no emplea ninguna sincronización entre el origen y el destino.
2. Trabaja con paquetes o datagramas enteros, no con bytes individuales como TCP. Una aplicación que emplea el protocolo UDP intercambia información en forma de bloques de bytes, de forma que por cada bloque de bytes enviado de la capa de aplicación a la capa de transporte, se envía un paquete UDP.
3. No es fiable. No emplea control del flujo ni ordena los paquetes.
4. Su gran ventaja es que provoca poca carga adicional en la red ya que es sencillo y emplea cabeceras muy simples.

UDP utiliza puertos para permitir la comunicación entre aplicaciones. El campo de puerto tiene una longitud de 16 bits, por lo que el rango de valores válidos va de 0 a 65.535. El puerto 0 está reservado, pero es un valor permitido como puerto origen si el proceso emisor no espera recibir mensajes como respuesta.

Los puertos 1 a 1023 se llaman puertos "bien conocidos" y en sistemas operativos tipo Unix enlazar con uno de estos puertos requiere acceso como superusuario.

Los puertos 1024 a 49.151 son puertos registrados.

Los puertos 49.152 a 65.535 son puertos dinámicos y son utilizados como puertos temporales, sobre todo por los clientes al comunicarse con los servidores.

ICMP

Internet Control Message Protocol (ICMP)) una red de protocolo que es responsable de reportar errores a través de la generación y envío de mensajes a la dirección IP de origen cuando hay problemas de red que son encontrados por el sistema. Los mensajes que genera la ICMP indican que no se puede acceder a un determinado gateway, router, servicio o incluso host que deba conectarse a Internet. Básicamente, el destinatario no puede recibir paquetes durante la transmisión. Cualquier dispositivo de red IP puede enviar, generar, recibir y procesar mensajes de error ICMP.

En otras palabras, usted puede imaginar a ICMP como un equipo que produce piezas mecánicas para vehículos. La mejor manera de ensamblar un vehículo sería fabricar primero cada pieza individual antes de enviar cada una de ellas a la cadena de montaje, que las ensamblará en un producto más complejo. Sin embargo, como todos sabemos, habrá momentos en los que el equipo de producción no podrá enviar algunas de las piezas en un plazo de tiempo programado. Si alguna vez van a faltar piezas, entonces la línea de montaje tendrá que notificar al equipo de producción, por lo tanto, la ICMP funciona de la misma manera. Esencialmente, la ICMP juega el papel de mensajero que transmite datos e información del destinatario al remitente.

Este protocolo es en realidad bastante complejo. Tiene muchas otras funciones además de reportar errores en las transmisiones de paquetes y hosts que no pueden ser alcanzados. También transmite mensajes de eco, así como de respuesta de eco. Esto se utiliza a través de un conocido comando PING que permite a la mayoría de los usuarios re-transmitir un eco a un host receptor. Esto envía una respuesta de eco cada vez que se recibe un eco.

Como tal, los mensajes ICMP proporcionan una forma para que la red y el sistema notifiquen la dirección IP de origen, que es el extremo de re-transmisión, si el host remoto, que es el extremo de recepción, no está recibiendo ningún paquete que se haya transmitido.

Se utiliza un enrutador para transmitir el popular mensaje "Destino inalcanzable" al host de origen. Esto entonces lo enviaría al dispositivo o programa que primero transmitió los paquetes de datos. Estos mensajes de error le harían saber al programa que había un problema con la conectividad de la red. Una vez que el programa fuente haya recibido la información de que algunos de esos paquetes de datos no llegaron al destinatario, entonces retransmitirá esta información al destinatario.

Iptables

Iptables es un módulo del núcleo de Linux que se encarga de filtrar los paquetes de red, es decir, es la parte que se encarga de determinar qué paquetes de datos queremos que lleguen hasta el servidor y cuáles no.

Al igual que ocurre con otros sistemas de cortafuegos, iptables funciona a través de reglas. Es decir, el usuario mediante sencillas instrucciones indica al firewall el tipo de paquetes que debe permitir entrar, los puertos por donde se pueden recibir esos paquetes, el protocolo utilizado para el envío de datos y cualquier otra información relacionada con el intercambio de datos entre redes.

Cuando en el sistema se recibe o se envía un paquete, se recorren en orden las distintas reglas hasta dar con una que cumpla las condiciones. Una vez localizada, esa regla se activa realizando sobre el paquete la acción indicada.

Gracias a su robustez, iptables se ha convertido hoy por hoy en una de las herramientas más utilizadas para el filtrado de tráfico en sistemas Linux.

Formato

Una vez vistas las distintas tablas que trae predefinidas la herramienta iptables, es hora de ver la estructura de las reglas que se pueden crear dentro de cada una de esas tablas.

La estructura general de una regla es la siguiente. Iptables –t [tabla] operación cadena parámetros acción

En ese esquema, los valores de tabla y cadena corresponden con las opciones vistas anteriormente. El resto de datos los veremos a continuación.

Parámetros

Los parámetros son utilizados para definir el paquete al que se le aplicará la regla. A continuación os detallamos los principales parámetros que podemos utilizar tanto en su forma abreviada como completa.

- **-p, --protocol**: Sirve para especificar el protocolo que se utiliza (tcp, udp, icmp, etc. Además si queremos especificar el puerto, se acompaña del parámetro –dport.
- **-s, --source:** Con este parámetro indicamos la dirección IP de origen.
- **-d, --destination:** Especificamos la dirección IP de destino
- **-i, --in-interface:** Especifica la interfaz de red de entrada (eth0, eth1,…)
- **-o, --out-interface:** Especifica la interfaz de red de salida (eth0, eth1,…)

Acciones

Una vez definida la regla, hay que indicar la acción que realizaremos sobre aquellos paquetes que la cumplan. Para indicar esta acción, haremos uso del parámetro –j seguido de alguno de los siguientes valores.

- **ACCEPT**: Mediante esta acción estamos indicando que el paquete sea aceptado.
- **DROP**: Se elimina el paquete y no se le envía al equipo que hizo la petición ningún mensaje de respuesta.
- **REJECT**: Similar al caso anterior, pero en esta ocasión se manda un paquete ICMP al equipo que hizo la petición para indicarle que no está permitida.

- **DNAT**: Esta acción es utilizada en la cadena PREROUTING de la tabla NAT para modificar la IP de destino. Tiene que llevar asociado el parámetro –to.
- **SNAT**: Acción asociada en la cadena POSTROUTING para modificar la IP origen. Al igual que el caso anterior le tiene que acompañar el parámetro –to.
- **MASQUERADE**: Acción equivalente a SNAT pero utilizada cuando tenemos una dirección IP dinámica en la interfaz de salida.
- **REDIRECT**: Se utiliza en la cadena PREROUTING para modificar la dirección IP que tenga la interfaz de red de entrada

Programador de Tareas

Al **programador de tareas** en GNU/Linux se le conoce con el nombre de **CRON**, y al editor de las tareas se le llama **CRONTAB**. Cron permite programar una tarea (por ejemplo un comando, programa o shell script) para ejecutarlos periódicamente o una única vez. Podremos programar las tareas de forma horaria, diaria, semanal, mensual, a una fecha y hora determinada, etc…

La herramienta Cron es muy útil y es usada por multitud de programas para realizar las tareas programadas, como por ejemplo los rotados de logs, monitorización de sistemas, chequeos de filesystem, etc..

At

Tal y como indica el título de la entrada hablamos del comando at, que nos permite programar tareas únicas en nuestro sistemas GNU/Linux No hace falta decir que para programar tareas repetitivas debemos utilizar cron y crontab, de los que ya hemos hablado en la web.

Mientras trabajamos en distribuciones GNU/Linux, generalmente utilizamos crontab para programar trabajos en general. Hay otra utilidad muy útil e interesante para programar tareas únicas. Con at podemos leer comandos de entrada estándar o scripts que deseamos

que se ejecuten más tarde, una única vez. Si queremos que un comando o un script se ejecute más de una vez, debemos programarlo mediante crontab, tal y como hemos indicado en el párrafo anterior.

El comando at, nos puede ser útil para apagar el sistema a una hora especifica, realizar una copia de seguridad única, enviar un correo electrónico como recordatorio a la hora especificada, entre otras muchas cosas.

No suelo utilizar este comando, pero quizás a vosotros os puede ser de utilidad en algún momento. El programa at forma parte del temario del temario de la certificación de LPIC 102, en su versión 500

Los comandos principales son:

- **at**: ejecuta comandos a la hora especificada.
- **atq**: enumera los trabajos pendientes de los usuarios.
- **atrm**: borra trabajos por su número 1.de trabajo.

Ejemplo

```
1 echo "sh copia-seguridad.sh" | at 10:00 PM
```

PRÁCTICA

Archivo enlace.conf

Definirá la velocidad que tendrán de subida y de bajada, su estructura será:

down = <cantidad en Mbps>

up = <cantidad en Mbps>

```
1   down=4
2   up=3
3
```

Archivo modo.conf

Definirá el modo en el que estará trabajando el ancho de banda, su estructura será:

modalidad = <número que identifica al tipo de modalidad>

Los tipos pueden ser:

- 1 = (fijo) de este modo el ancho de banda será únicamente el especificado, no más.
- 2 = (dinámico) de este modo el ancho de banda será como mínimo lo especificado o más.

```
1  modalidad=1
2
```

Archivo usuario_BW.conf

Definirá la configuración por MAC del ancho de banda que tendrá cada equipo conectado, su estructura será:

<MAC>,<BW bajada>,<BW subida>,<hora_inicio>,<hora_fin>

- El formato de hora es siempre de 24 hrs.
- El ancho de banda se maneja en porcentaje un valor de 0 - 100

```
1  08:00:27:7C:9D:D9,40,40,19:00,21:00
2  08:00:27:8A:7A:FA,30,30,19:00,21:00
3  08:00:27:BD:CE:4D,30,30,19:00,21:00
4
```

Archivo usuario_Proto.conf

Este archivo definirá la configuración por MAC del protocolo por el cual se enviará la información así como los puertos, su estructura será

<MAC>,<protocolo>,<puerto>,<hora_inicio>,<hora_fin>

- Protocolo pueden ser: UDP, TCP y ICMP

```
08:00:27:7C:9D:D9,icmp,19:00,21:10
08:00:27:8A:7A:FA,icmp,19:00,21:10
08:00:27:BD:CE:4D,icmp,19:00,21:10
08:00:27:7C:9D:D9,tcp,1:65535,19:00,21:10
08:00:27:8A:7A:FA,tcp,1:65535,19:00,21:10
08:00:27:BD:CE:4D,tcp,1:65535,19:00,21:10
08:00:27:7C:9D:D9,udp,1:65535,19:00,21:10
08:00:27:8A:7A:FA,udp,1:65535,19:00,21:10
08:00:27:BD:CE:4D,udp,1:65535,19:00,21:10

```

Estructura Configuración HTB

- tc qdisc add dev **enp0s8** root handle 1: htb default 0xA
- tc class add dev **enp0s8** parent 1:1 classid 1:101 htb rate 1Kbit
- tc class add dev **enp0s8** parent 1:1 classid 1:102 htb rate 1Kbit
- tc class add dev **enp0s8** parent 1:1 classid 1:103 htb rate 1Kbit
- *arreglo_mac* **MAC_1** 1:101
- *arreglo_mac* **MAC_2** 1:102
- *arreglo_mac* **MAC_3** 1:103

```
#nodo raiz
tc qdisc add dev $INTERFAZ root handle 1: htb default 0xA
```

```

#Creamos los nodos hijos, que serían 3 usuarios (los usuarios centos)

tc class add dev $INTERFAZ parent 1:1 classid 1:101 htb rate 1Kbit
tc class add dev $INTERFAZ parent 1:1 classid 1:102 htb rate 1Kbit
tc class add dev $INTERFAZ parent 1:1 classid 1:103 htb rate 1Kbit


arreglo_mac $MAC_1 1:101
arreglo_mac $MAC_2 1:102
arreglo_mac $MAC_3 1:103

```

Asignación Colas HTB

(*arreglo_mac*)

- tc filter add dev **enp0s8** parent 1: protocol ip prio 5 u32 match u16 0x0800 0xFFFF at -2 match u16 0x**MAC_ULTIMOS_4_DIGITOS** 0xFFFF at -4 match u32 0x**MAC_PRIMEROS_8_DIGITOS** 0xFFFFFFFF at -8 flowid **ID**
- tc filter add dev **enp0s8** parent 1: protocol ip prio 5 u32 match u16 0x0800 0xFFFF at -2 match u32 0x**MAC_ULTIMOS_8_DIGITOS** 0xFFFFFFFF at -12 match u16 0x**MAC_PRIMEROS_4_DIGITOS** 0xFFFF at -14 flowid **ID**

```
arreglo_mac() {
    #split mac
    readarray -d : -t PARTES <<< $1
    P1=${PARTES[0]}${PARTES[1]}
    P2=${PARTES[2]}${PARTES[3]}
    P3=${PARTES[4]}${PARTES[5]}

#asignar mac **.**.**.**.**.**
    #si es origen
    tc filter add dev $INTERFAZ parent 1: protocol ip prio 5 u32 match u16 0x0800 0xFFFF at -2 match u16 0x${P3} 0xFFFF at -4 match u32 0x${P1}${P2} 0xFFFFFFFF at -8 flowid $2
    #$TCF match u16 0x${P3} 0xFFFF at -4 match u32 0x${P1}${P2} 0xFFFFFFFF at -8 flowid $2
    #si es destino
    tc filter add dev $INTERFAZ parent 1: protocol ip prio 5 u32 match u16 0x0800 0xFFFF at -2 match u32 0x${P2}${P3} 0xFFFFFFFF at -12 match u16 0x${P1} 0xFFFF at -14 flowid $2
    #$TCF match u32 0x${P2}${P3} 0xFFFFFFFF at -12 match u16 0x${P1} 0xFFFF at -14 flowid $2
}
```

Configuración inicial iptables

- iptables -F
- iptables -P INPUT DROP
- iptables -P FORWARD DROP
- iptables -P OUTPUT DROP

```
#Eliminar toda conexión de iptables de fabrica
iptables -P INPUT DROP
iptables -P FORWARD DROP
iptables -P OUTPUT DROP
```

Lectura archivo enlace.conf

ANCHOB = down

ANCHOS = up

ANCHO=(**ANCHOB**+**ANCHOS**)*1024

```
#ejecutar y empezar con las lecturas

echo 'Archivo de enlace'
while read -r l
do
    readarray -d = -t ENLACE <<< $l
    #ANCHO DE BAJADA
    if [ ${ENLACE[0]} = 'down' ]; then ANCHOB=${ENLACE[1]}; fi
    #ANCHO DE SUBIDA
    if [ ${ENLACE[0]} = 'up' ]; then ANCHOS=${ENLACE[1]}; fi
done < ../archivos/enlace.conf
ANCHO=$(((ANCHOB+ANCHOS)*1024))
echo "El ancho de banda es de: ${ANCHO}Kbit"
echo
echo
```

Lectura archivo modo.conf

MODO = mode

```
echo 'Archivo de modo'
while read -r l
do
    readarray -d = -t MODO <<< $l
    MODOF=${MODO[1]}
done < ../archivos/modo.conf
echo "El modo de configuración es ${MODOF}"
echo
echo

```

Lectura archivo usuario_BW.conf

Si el modo es dinámico, se agrega esta cadena

CADENA= ‘ceil **ANCHO**Kbit’

Por cada línea hacer:

AB = Porcentaje ancho de bajada (2do parámetro)

AS = Porcentaje ancho de subida (3er parámetro)

BWT = ((**ANCHOB** * **AB**) + (**ANCHOS** * **AS**)) * 1024 / 100)

si es **MAC_1** igual a la mac de la línea

/sbin/tc class change dev **enp0s8** parent 1:1 classid 1:101 htb rate **BWT**Kbit'

CADENA

si es **MAC_2** igual a la mac de la línea

/sbin/tc class change dev **enp0s8** parent 1:1 classid 1:102 htb rate **BWT**Kbit'

CADENA

si es **MAC_3** igual a la mac de la línea

/sbin/tc class change dev **enp0s8** parent 1:1 classid 1:103 htb rate **BWT**Kbit'

CADENA

```
26
27  if [ $MODOF -eq 2 ]; then CADENA="ceil ${ANCHO}Kbit"; fi
28  echo 'Archivo de usuario_BW'
29  while read -r l
30  do
31      readarray -d , -t USUARIOBW <<< $l
32      echo ${USUARIOBW[0]}
33      #ancho de bajada
34      AB=${USUARIOBW[1]}
35      #ancho de subida
36      AS=${USUARIOBW[2]}
37      BWT=$((((ANCHOB*AB)+(ANCHOS*AS))*1024/100))
38      #BWT=$((ANCHO*(AB+AS)/100))
39      #readarray -d : -t H_INICIO <<< ${USUARIOBW[3]}
40      #readarray -d : -t H_FIN <<< ${USUARIOBW[4]}
41      bash i_at.sh ${USUARIOBW[3]} $BWT ${USUARIOBW[0]} $CADENA
42      bash i_at.sh ${USUARIOBW[4]} 1 ${USUARIOBW[0]}
43  done < ../archivos/usuario_BW.conf
44
```

```
#$1 hora:minuto
#$2 ancho de banda
#$3 dirección mac
#$4 CADENA

INTERFAZ=enp0s8
#direcciones mac de clientes
echo $4
i=0
while read -r l
do
    if [ $i -eq 0 ]; then MAC_1=${l}; fi
    if [ $i -eq 1 ]; then MAC_2=${l}; fi
    if [ $i -eq 2 ]; then MAC_3=${l}; fi
    i=$((i+1))
done < ../archivos/macsiniciales.conf

if [ $3 == $MAC_1 ]; then
    echo $MAC_1
    echo "echo '/sbin/tc class change dev $INTERFAZ parent 1:1 classid 1:101 htb rate ${2}Kbit $4' | at $1" >> /home/marito/Documentos/reglas/reglas.sh
elif [ $3 == $MAC_2 ]; then
    echo $MAC_2
    echo "echo '/sbin/tc class change dev $INTERFAZ parent 1:1 classid 1:102 htb rate ${2}Kbit $4' | at $1" >> /home/marito/Documentos/reglas/reglas.sh
elif [ $3 == $MAC_3 ]; then
    echo $MAC_3
    echo "echo '/sbin/tc class change dev $INTERFAZ parent 1:1 classid 1:103 htb rate ${2}Kbit $4' | at $1" >> /home/marito/Documentos/reglas/reglas.sh
fi
echo

```

Lectura archivo usuario_PROTO.conf

```
 echo 'Archivo de usuario-PROTO'
 while read -r l
 do
     readarray -d , -t USUARIOPROTO <<< $l
     if [ ${#USUARIOPROTO[@]} -eq 4 ]; then #parametros: mac,protocolo,horainicio,horafin
       bash i_command.sh 0 icmp ${USUARIOPROTO[0]} ${USUARIOPROTO[2]} ${USUARIOPROTO[3]}
     fi

     if [ ${#USUARIOPROTO[@]} -eq 5 ]; then #parametros: mac,protocolo,puerto(s),horainicio,horafin
       readarray -d : -t PUERTOS <<< "${parametros[2]}"
       if [ ${#PUERTOS[@]} -eq 2 ]; then
         bash i_command.sh 1 ${USUARIOPROTO[1]} ${USUARIOPROTO[0]} ${USUARIOPROTO[3]} ${USUARIOPROTO[4]} ${PUERTOS[0]} ${PUERTOS[1]}
       else
         bash i_command.sh 2 ${USUARIOPROTO[1]} ${USUARIOPROTO[0]} ${USUARIOPROTO[3]} ${USUARIOPROTO[4]} ${USUARIOPROTO[2]}
       fi
     fi
 done < ../archivos/usuario_PROTO.conf
 echo
```

Por cada línea hacer:

- *Si es icmp*

  /sbin/iptables -I FORWARD 1 -p icmp -m mac --mac-source **MAC** -j ACCEPT

  /sbin/iptables -I FORWARD 1 -p icmp -m state --state RELATED,ESTABLISHED -j ACCEPT

  /sbin/iptables -D FORWARD -p icmp -m mac --mac-source **MAC** -j ACCEPT

  /sbin/iptables -D FORWARD -p icmp -m state --state RELATED,ESTABLISHED -j ACCEPT

```
if [ $1 -eq 0 ]; then
    echo "echo '/sbin/iptables -I FORWARD 1 -p icmp -m mac --mac-source $MAC -j ACCEPT' | at $4" >> /home/marito/Documentos/reglas/reglas.sh
    echo "echo '/sbin/iptables -I FORWARD 1 -p icmp -m state --state RELATED,ESTABLISHED -j ACCEPT' | at $4" >> /home/marito/Documentos/reglas/reglas.sh
    echo "echo '/sbin/iptables -D FORWARD -p icmp -m mac --mac-source $MAC -j ACCEPT' | at $5" >> /home/marito/Documentos/reglas/reglas.sh
    echo "echo '/sbin/iptables -D FORWARD -p icmp -m state --state RELATED,ESTABLISHED -j ACCEPT' | at $5" >> /home/marito/Documentos/reglas/reglas.sh
    #iptables -I FORWARD 1 -p icmp -m mac --mac-source $MAC -m time --timestart $4 --timestop $5 -j ACCEPT
    #iptables -I FORWARD 1 -p icmp -m state --state RELATED,ESTABLISHED -m time --timestart $4 --timestop $5 -j ACCEPT
fi
```

- *Si es tcp o udp con dos puertos*

/sbin/iptables -I FORWARD 1 -p **PROTOCOLO** -m mac --mac-source **MAC** -m **PRROTOCOLO** --dport **PUERTOS** -j ACCEPT

/sbin/iptables -I FORWARD 1 -p **PROTOCOLO** -m state --state RELATED,ESTABLISHED -m **PROTOCOLO** --sport **PUERTOS** -j ACCEPT

/sbin/iptables -D FORWARD -p **PROTOCOLO** -m mac --mac-source **MAC** -m **PROTOCOLO** --dport **PUERTOS** -j ACCEPT

/sbin/iptables -D FORWARD -p **PROTOCOLO** -m state --state RELATED,ESTABLISHED -m **PROTOCOLO** --sport **PUERTOS** -j ACCEPT

```
if [ $1 -eq 1 ]; then
    echo "echo '/sbin/iptables -I FORWARD 1 -p $2 -m mac --mac-source $MAC -m $2 --dport $6:$7 -j ACCEPT' | at $4" >> /home/marito/Documentos/reglas/reglas.sh
    echo "echo '/sbin/iptables -I FORWARD 1 -p $2 -m state --state RELATED,ESTABLISHED -m $2 --sport $6:$7 -j ACCEPT' | at $4" >> /home/marito/Documentos/reglas/reglas.sh
    echo "echo '/sbin/iptables -D FORWARD -p $2 -m mac --mac-source $MAC -m $2 --dport $6:$7 -j ACCEPT' | at $5" >> /home/marito/Documentos/reglas/reglas.sh
    echo "echo '/sbin/iptables -D FORWARD -p $2 -m state --state RELATED,ESTABLISHED -m $2 --sport $6:$7 -j ACCEPT' | at $5" >> /home/marito/Documentos/reglas/reglas.sh
    #iptables -I FORWARD 1 -p $2 -m mac --mac-source $MAC -m $2 --dport $6:$7 -m time --timestart $4 --timestop $5 -j ACCEPT
    #iptables -I FORWARD 1 -p $2 -m state --state RELATED,ESTABLISHED -m $2 --sport $6:$7 -m time --timestart $4 --timestop $5 -j ACCEPT
fi
```

- *Si es tcp o udp con un puerto*

/sbin/iptables -I FORWARD 1 -p **PROTOCOLO** -m mac --mac-source $MAC -m **PROTOCOLO** --dport **PUERTO** -j ACCEPT

/sbin/iptables -I FORWARD 1 -p **PROTOCOLO** -m state --state RELATED,ESTABLISHED -m **PROTOCOLO** --sport **PUERTO** -j ACCEPT

/sbin/iptables -D FORWARD -p **PROTOCOLO** -m mac --mac-source $MAC -m **PROTOCOLO** --dport **PUERTO** -j ACCEPT

/sbin/iptables -D FORWARD -p **PROTOCOLO** -m state --state RELATED,ESTABLISHED -m **PROTOCOLO** --sport **PUERTO** -j ACCEPT

```
if [ $1 -eq 2 ]; then
    echo "echo '/sbin/iptables -I FORWARD 1 -p $2 -m mac --mac-source $MAC -m $2 --dport $6 -j ACCEPT' | at $4" >> /home/marito/Documentos/reglas/reglas.sh
    echo "echo '/sbin/iptables -I FORWARD 1 -p $2 -m state --state RELATED,ESTABLISHED -m $2 --sport $6 -j ACCEPT' | at $4" >> /home/marito/Documentos/reglas/reglas.sh
    echo "echo '/sbin/iptables -D FORWARD -p $2 -m mac --mac-source $MAC -m $2 --dport $6 -j ACCEPT' | at $5" >> /home/marito/Documentos/reglas/reglas.sh
    echo "echo '/sbin/iptables -D FORWARD -p $2 -m state --state RELATED,ESTABLISHED -m $2 --sport $6 -j ACCEPT' | at $5" >> /home/marito/Documentos/reglas/reglas.sh
    #iptables -I FORWARD 1 -p $2 -m mac --mac-source $MAC -m $2 --dport $6 -m time --timestart $4 --timestop $5 -j ACCEPT
    #iptables -I FORWARD 1 -p $2 -m state --state RELATED,ESTABLISHED -m $2 --sport $6 -m time --timestart $4 --timestop $5 -j ACCEPT
fi
```

Ejecutar a una hora determinada las instrucciones

Para este paso vamos a usar at, siendo un comando muy util para programar tareas esto haciendo:

echo '**CÓDIGO A EJECUTAR'** | at **HORA**

Pero este **CÓDIGO A EJECUTAR**, es los comandos que hemos visto anteriormente, en el archivo de usuario_BW.conf y usuario_PROTO.conf, donde aparece una hora inicio y una hora final, por eso programamos lo que es un comando para cuando empieza y un comando para cuando finaliza, estos comandos lo vamos a colocar en un archivo de **/tareas.sh** donde la ultima linea es una autoreferencia, de que al siguiente día se ejecute de nuevo ese archivo, esto para que no se pierdan las instrucciones, y queden a diario.

```
#ejecutar las reglas en las horas programadas
echo "echo 'sh /home/marito/Documentos/reglas/reglas.sh' | at now + 1 days" >> /home/marito/Documentos/reglas/reglas.sh
sh /home/marito/Documentos/reglas/reglas.sh
```

CONCLUSIONES

- Es un proyecto muy útil en el ámbito personal, como laboral, ya que se visualizan muchas utilidades de lo que es manejo de redes, de ancho de banda y programación de tareas, donde genera un ecosistema donde se pueden hacer cosas muy interesantes, y provechosas.
- El uso de HTB es bastante práctico, aunque es difícil de comprender como cualquier tecnología al principio, se vuelve muy fácil de entender y de aplicar, donde se vuelve una herramienta muy potente.
- El uso de iptables, es más engorroso, más por los puertos, y protocolos, que generan muchas trabas en lo que es la fluidez del proyecto, ya que son difíciles de comprobar, y aparte funcionan de distintas maneras, y el formato, como los parámetros son varios, y bastante complicados de usar dinámicamente.
- Lo que es el manejador de tareas, se empezó con idea de usar crontab, pero aconsejo mejor usar at, tiene una sintaxis mucho más fácil, cómoda de usar, y aparte no genera tantos problemas, como los puede o no generar crontab, por lo que siento que es una herramienta mucho más eficiente, optimizada y aconsejable en usarse.

- El proyecto final fue un gran ecosistema muy interesante de visualizar como es que cada comando, regla, protocolo, afecta en el ancho de banda del cliente, o de la posibilidad de conectarse o no a internet.

Recomendaciones

- Se recomienda leer muy detenidamente lo que es iptables, para el funcionamiento y agilidad de realizar comandos, para cada tipo de protocolo, o diferente dinámica que se necesita en cada caso.
- Recomiendo lo que es usar **at**, en vez de **crontab**, crontab me resultó demasiado engorroso, aparte de no funcionar en mi caso, aunque preguntando a compañeros, algunos sí lograron hacerlo funcionar, algunos de una manera sencillo, otros de manera más complicada, y otros no les funcionó, por lo que siento que no es una herramienta tan aconsejable de usar.
- Entender el manejo de colas de HTB y cómo funcionan sus nodos, como el árbol generado al configurar estas, logra un conocimiento mucho más amplio, aparte de ser una herramienta sencilla de utilizar, por lo que se aconseja simplemente, entender este concepto de los árboles.

BIBLIOGRAFÍA

- https://espanol.verizon.com/info/definitions/bandwidth/
- https://www.paessler.com/es/it-explained/bandwidth
- https://www.ibm.com/docs/es/aix/7.1?topic=protocol-tcpip-protocols
- https://www.alcancelibre.org/staticpages/index.php/distribucion-ancho-banda-htb-iptables
- https://www.redeszone.net/tutoriales/configuracion-puertos/puertos-tcp-udp/
- https://es.wikipedia.org/wiki/Protocolo_de_datagramas_de_usuario
- https://www.speedcheck.org/es/wiki/icmp/

- https://www.acens.com/wp-content/images/2014/07/wp-acens-iptables.pdf
- https://www.raulprietofernandez.net/blog/gnu-linux/como-programar-tareas-en-gnu-linux
- https://www.ochobitshacenunbyte.com/2019/07/01/comando-at-programacion-de-tareas-unicas-en-linux/