

Administración de Redes Locales

Netkit

Caramutti Ignacio, Gonzalez Rosario, Olivares Ana
5to Informática

Netkit:

Netkit es un entorno para configurar y realizar experimentos de redes a bajo costo y con poco esfuerzo. Permite "crear" varios dispositivos de red virtual (enrutadores completos, conmutadores, computadoras, etc.) que pueden interconectarse fácilmente para formar una red en una sola PC. Los equipos de red son virtuales pero presentan muchas de las características de los reales, incluida la interfaz de configuración.

Las máquinas virtuales utilizan un sistema de archivos personalizado que contiene una instalación estándar de Linux, que incluye software orientado a la red, como routing daemons (RIP, OSPF, etc.), servidores (FTP, HTTP, etc.), herramientas de diagnóstico (ping, traceroute, tcpdump, etc.) y otras aplicaciones.

Iniciando Netkit:

Para iniciar netkit debemos entrar en la terminal de Linux y escribir el siguiente comando:

```
$NETKIT_HOME
```

Una vez ingresado en Netkit, se puede empezar a trabajar. Primero se debe crear un directorio e ingresar a este, utilizando los comandos:

```
mkdir lab  
cd lab/
```

Siendo lab el nombre de nuestro directorio.

Después de ingresar al directorio, "creamos" una carpeta para cada uno de los dispositivos de nuestra red virtual, utilizando el mismo comando `mkdir`

.

Luego, creamos un archivo de texto donde configuraremos nuestra red virtual. Los nombres utilizados aquí deben coincidir con los de las carpetas ya creadas.

```
nano lab.config
```

Por último, iniciamos nuestra red con el comando:

```
lstart
```

Bridge:

Un bridge conecta dos segmentos de red como una sola red, haciendo el pasaje de datos de una red hacia otra, con base en la dirección física de destino de cada paquete. Utiliza un mismo protocolo de establecimiento de red. Opera en la capa 2 del modelo OSI.

Switch:

Un switch es un dispositivo digital lógico de interconexión de equipos que opera en la capa de enlace de datos del modelo OSI. Su función es interconectar dos o más hosts de manera similar a los puentes de red, pasando datos de un segmento a otro de acuerdo con la dirección MAC de destino de las tramas en la red y eliminando la conexión una vez finalizada ésta.

Sniffer:

Un Sniffer es un programa de captura de las tramas de una red de computadoras.

Es algo común que, por topología de red y necesidad material, el medio de transmisión sea compartido por varias computadoras y dispositivos de red, lo que hace posible que un ordenador capture las tramas de información no destinadas a él. Para conseguir esto el analizador pone la tarjeta de red en un estado conocido como

”modo promiscuo” en el cual en la capa de enlace de datos no son descartadas las tramas no destinadas a la dirección MAC de la tarjeta; de esta manera se puede capturar todo el tráfico que viaja por la red.

Los analizadores de paquetes tienen diversos usos, como monitorear redes para detectar y analizar fallos, o para realizar ingeniería inversa en protocolos de red.

Ejemplo:

Creamos cuatro PC's virtuales, un Switch y un Sniffer virtual. Conectamos la PC1 y la PC2 al cable ”A”, la PC3 y PC4 al cable ”B” y el SW y el SNIFF a estos dos cables. Los comandos utilizados son:

```
brctl addbr br0
```

Este comando añade un bridge.

```
brctl addif br0 eth0
```

```
brctl addif br0 eth1
```

Conectan los bridge con el switch.

```
ifconfig eth0 up
```

```
ifconfig eth1 up
```

```
ifconfig br0 up
```

Los últimos comandos comienzan el bridge.

Administración de Redes Locales - Netkit

The image displays two screenshots of a Netkit virtual machine environment, showing the configuration and execution of a network setup across four PCs (PC1, PC2, PC3, PC4).

Top Screenshot:

- PC1:** Shows the lab directory (host) and the Netkit phase 2 initialization terminated. It displays the output of the `ifconfig` command, showing the interface `eth0` is up and running.
- PC2:** Shows the lab directory (host) and the Netkit phase 2 initialization terminated. It displays the output of the `ifconfig` command, showing the interface `eth0` is up and running.
- PC3:** Shows the lab directory (host) and the Netkit phase 2 initialization terminated. It displays the output of the `ifconfig` command, showing the interface `eth0` is up and running.
- PC4:** Shows the lab directory (host) and the Netkit phase 2 initialization terminated. It displays the output of the `ifconfig` command, showing the interface `eth0` is up and running.

Bottom Screenshot:

- PC1:** Shows the lab directory (host) and the Netkit phase 2 initialization terminated. It displays the output of the `ifconfig` command, showing the interface `eth0` is up and running.
- PC2:** Shows the lab directory (host) and the Netkit phase 2 initialization terminated. It displays the output of the `ifconfig` command, showing the interface `eth0` is up and running.
- PC3:** Shows the lab directory (host) and the Netkit phase 2 initialization terminated. It displays the output of the `ifconfig` command, showing the interface `eth0` is up and running.
- PC4:** Shows the lab directory (host) and the Netkit phase 2 initialization terminated. It displays the output of the `ifconfig` command, showing the interface `eth0` is up and running.

The screenshots illustrate the configuration and execution of a network setup across four PCs, showing the output of the `ifconfig` command and the status of the network interfaces.



STP:

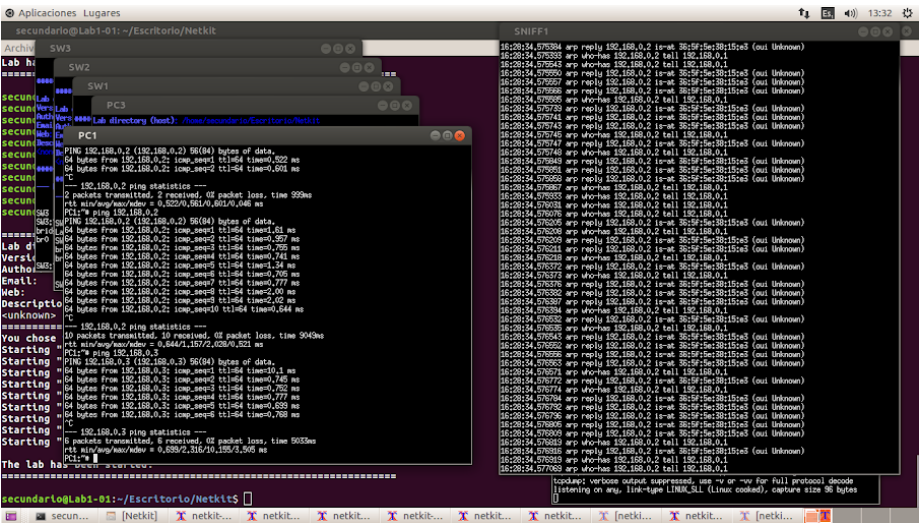
STP significa Spanning Tree Protocol (Protocolo de árbol de expansión), es un protocolo de red de capa 2 del modelo OSI y, se encarga de reconocer y administrar bucles en topologías de redes nacidos a función de la redundancia en la misma.

Es una práctica común que en el diseño de la topología de una red se agreguen enlaces redundantes con el fin de hacer la red tolerable a fallas y de esta forma si falla un enlace la red tiene la capacidad de recuperarse inmediatamente sin que el usuario ni siquiera note la falla.

Este tipo de falla se conoce como tormenta de broadcast: los broadcast en la red son reenviados una y otra vez y permanecen circulando en la misma sin fin, dado que en Ethernet no existe como en IP un campo de TTL. Lógicamente, al no eliminarse la situación se agrava con cada nuevo broadcast.

Ejemplo:

Creamos las PC's y los SW de manera que se produzca la tormenta de broadcast.



Port Bonding:

El Port Bonding es una técnica que permite agregar varios interfaces de red físicos en uno único virtual. A cada interfaz físico se le denominará slave. Con esto podemos realizar un balanceo de carga entre las dos interfaces y conseguir un ancho de banda final igual a la suma de los anchos de banda de cada slave. Además de una ventaja adicional inmediata: redundancia de la conexión, lo que implica que si tenemos varios enlaces físicos a la red, perder alguno de ellos supondría una degradación de servicio pero no la pérdida completa de conexión. Tiene 7 posibles “modos” para interfaces enlazadas. Estos modos determinan la manera en que el tráfico enviado desde la interfaz enlazada se dispersa en las interfaces reales.

Modo 0: Este modo transmite paquetes en un orden secuencial desde el primer slave disponible hasta el último. Si dos interfaces reales son slaves en el enlace y dos paquetes llegan destinados fuera de la interfaz enlazada, el primero se transmitirá en el primer slave y la segunda trama se transmitirá en el segundo slave. El tercer paquete se enviará el primero y así sucesivamente. Esto proporciona equilibrio de carga y tolerancia a fallos.

Modo 1: (copia de seguridad activa). Coloca una de las interfaces en un estado de respaldo y solo la activará si la interfaz activa pierde el enlace. Solo un slave en el enlace está activo en un momento determinado. Un slave diferente se activa solo cuando falla el slave activo. Este

modo proporciona tolerancia a fallos.

Modo 2: Transmite en base a la fórmula XOR. Esto selecciona el mismo slave para cada dirección MAC de destino y proporciona balanceo de carga y tolerancia a fallas.

Modo 3: El modo de transmisión transmite todo en todas las interfaces de slave. Este modo se usa menos (solo para un propósito específico) y proporciona solo tolerancia a fallas.

Modo 4: El modo 802.3ad se conoce como modo de agregación de enlace dinámico. Crea grupos de agregación que comparten la misma velocidad y configuraciones de dúplex. Este modo requiere un conmutador que admita el enlace dinámico IEEE 802.3ad. La selección de slaves para el tráfico saliente se realiza de acuerdo con la política de hash de transmisión. No todas las políticas de transmisión pueden ser compatibles con 802.3ad, particularmente en lo que respecta a los requisitos de pedido erróneo de paquetes de la sección 43.2.4 de la norma 802.3ad. Las diferentes implementaciones entre pares tendrán diferentes tolerancias para el incumplimiento.

Modo 5: El tráfico saliente se distribuye de acuerdo con la carga actual y la cola en cada interfaz de slave. El tráfico entrante es recibido por el slave actual.

Modo 6: Este es el modo de equilibrio de carga adapt-

able. Esto incluye balance-tlb + recibir balanceo de carga (rlb) para el tráfico IPV4. El equilibrio de carga de recepción se logra mediante la negociación ARP. El controlador de enlace intercepta las Respuestas ARP enviadas por el servidor al salir y sobrescribe la dirección src hw con la dirección hw única de uno de los slaves en el enlace, de modo que diferentes clientes utilicen diferentes direcciones hw para el servidor.

Ejemplo:

Creamos dos PC's, dos SW y los conectamos. Los comandos utilizados son:

```
brctl addbr br0
modprobe bonding
Cargamos el driver
echo balance-rr > /sys/class/net/bond0/bonding/mpd
echo +eth0 > /sys/class/net/bond0/bonding/slaves
echo +eth1 > /sys/class/net/bond0/bonding/slaves
brctl addif br0 eth0 eth2
brctl addif br0 eth0 bond0
ifconfig eth0 up
ifconfig eth1 up
ifconfig eth2 up
ifconfig bond0 up
ifconfig br0 up
```

Administración de Redes Locales - Netkit

The screenshot displays a Netkit virtual network environment. It features four terminal windows: SW1, PC1, PC2, and SW2. Each window shows network traffic logs and configuration commands. The SW1 and SW2 windows show the initialization of Netkit phase 2, including commands like 'login root', 'droping NETIF_F_UFO', and 'modprobe bonding'. The PC1 and PC2 windows show the starting of 'PC1' and 'PC2' respectively, with logs indicating network traffic between the hosts. The SW2 window also shows a 'Starting "PC1"...' message. The bottom of the screen shows a taskbar with various application icons, including 'eclipse_workspace', 'Planillas', and 'workspace'.

```
SW1
----- Netkit phase 2 initialization terminated -----
SW1 login root (automatic login)
SW1# brctl addbr br0
br0: Dropping NETIF_F_UFO since no NETIF_F_MCSUM feature.
SW1# modprobe bonding
SW1# echo balancer > /sys/class/net/bond0/bonding/mode
SW1# echo balance > /sys/class/net/bond0/bonding/ndo
SW1# echo eth0 > /sys/class/net/bond0/bonding/slaves
SW1# brctl addif br0 eth0
SW1# ifconfig br0 up
SW1# ifconfig eth0 up
SW1# ifconfig eth0 up
SW1# ifconfig bond0 up
SW1# ifconfig br0 up
SW1#

PC1
64 bytes from 192.168.0.2: icmp_seq=142 ttl=64 time=0.635 ms
64 bytes from 192.168.0.2: icmp_seq=143 ttl=64 time=0.631 ms
64 bytes from 192.168.0.2: icmp_seq=144 ttl=64 time=0.777 ms
64 bytes from 192.168.0.2: icmp_seq=145 ttl=64 time=1.15 ms
64 bytes from 192.168.0.2: icmp_seq=146 ttl=64 time=0.888 ms
64 bytes from 192.168.0.2: icmp_seq=147 ttl=64 time=0.833 ms
64 bytes from 192.168.0.2: icmp_seq=148 ttl=64 time=1.035 ms
64 bytes from 192.168.0.2: icmp_seq=149 ttl=64 time=1.18 ms
64 bytes from 192.168.0.2: icmp_seq=150 ttl=64 time=1.081 ms
64 bytes from 192.168.0.2: icmp_seq=151 ttl=64 time=1.39 ms
64 bytes from 192.168.0.2: icmp_seq=152 ttl=64 time=1.27 ms
64 bytes from 192.168.0.2: icmp_seq=153 ttl=64 time=1.36 ms
64 bytes from 192.168.0.2: icmp_seq=154 ttl=64 time=1.24 ms
64 bytes from 192.168.0.2: icmp_seq=155 ttl=64 time=1.02 ms
64 bytes from 192.168.0.2: icmp_seq=156 ttl=64 time=1.17 ms
64 bytes from 192.168.0.2: icmp_seq=157 ttl=64 time=0.874 ms
64 bytes from 192.168.0.2: icmp_seq=158 ttl=64 time=1.02 ms
64 bytes from 192.168.0.2: icmp_seq=159 ttl=64 time=1.00 ms
64 bytes from 192.168.0.2: icmp_seq=160 ttl=64 time=1.12 ms
64 bytes from 192.168.0.2: icmp_seq=161 ttl=64 time=1.11 ms
64 bytes from 192.168.0.2: icmp_seq=162 ttl=64 time=1.11 ms
64 bytes from 192.168.0.2: icmp_seq=163 ttl=64 time=1.11 ms
64 bytes from 192.168.0.2: icmp_seq=164 ttl=64 time=1.11 ms

PC2
64 bytes from 192.168.0.1: icmp_seq=1153 ttl=64 time=0.839 ms
64 bytes from 192.168.0.1: icmp_seq=1154 ttl=64 time=0.858 ms
64 bytes from 192.168.0.1: icmp_seq=1155 ttl=64 time=1.01 ms
64 bytes from 192.168.0.1: icmp_seq=1156 ttl=64 time=1.03 ms
64 bytes from 192.168.0.1: icmp_seq=1157 ttl=64 time=1.03 ms
64 bytes from 192.168.0.1: icmp_seq=1158 ttl=64 time=1.04 ms
64 bytes from 192.168.0.1: icmp_seq=1159 ttl=64 time=1.04 ms
64 bytes from 192.168.0.1: icmp_seq=1160 ttl=64 time=1.04 ms
64 bytes from 192.168.0.1: icmp_seq=1161 ttl=64 time=1.04 ms
64 bytes from 192.168.0.1: icmp_seq=1162 ttl=64 time=1.04 ms
64 bytes from 192.168.0.1: icmp_seq=1163 ttl=64 time=1.04 ms
64 bytes from 192.168.0.1: icmp_seq=1164 ttl=64 time=1.04 ms
64 bytes from 192.168.0.1: icmp_seq=1165 ttl=64 time=1.04 ms
64 bytes from 192.168.0.1: icmp_seq=1166 ttl=64 time=1.04 ms
64 bytes from 192.168.0.1: icmp_seq=1167 ttl=64 time=1.04 ms
64 bytes from 192.168.0.1: icmp_seq=1168 ttl=64 time=1.04 ms
64 bytes from 192.168.0.1: icmp_seq=1169 ttl=64 time=1.04 ms
64 bytes from 192.168.0.1: icmp_seq=1170 ttl=64 time=1.04 ms
64 bytes from 192.168.0.1: icmp_seq=1171 ttl=64 time=1.04 ms
64 bytes from 192.168.0.1: icmp_seq=1172 ttl=64 time=1.04 ms
64 bytes from 192.168.0.1: icmp_seq=1173 ttl=64 time=1.04 ms
64 bytes from 192.168.0.1: icmp_seq=1174 ttl=64 time=1.04 ms
64 bytes from 192.168.0.1: icmp_seq=1175 ttl=64 time=1.04 ms

SW2
----- Netkit phase 2 initialization terminated -----
SW2 login root (automatic login)
SW2# brctl addbr br0
br0: Dropping NETIF_F_UFO since no NETIF_F_MCSUM feature.
SW2# modprobe bonding
SW2# echo balancer > /sys/class/net/bond0/bonding/mode
SW2# echo eth0 > /sys/class/net/bond0/bonding/slaves
SW2# brctl addif br0 eth0
SW2# ifconfig br0 up
SW2# ifconfig eth0 up
SW2# ifconfig eth0 up
SW2# ifconfig bond0 up
SW2# ifconfig br0 up
SW2#
```

The image shows a Kali Linux desktop environment with three terminal windows open. The top window, titled "SW1", displays the output of a "netifd phase 2 initialization terminated" command, showing a list of IP addresses and their corresponding netifd configurations. The middle window, titled "PC1", shows a similar output for a different device. The bottom window, titled "SW2", shows the output of a "netifd phase 2 initialization terminated" command, displaying a list of IP addresses and their corresponding netifd configurations. The desktop background is a dark theme with a "Kali Linux" logo. The taskbar at the bottom shows various application icons, including a file manager, a terminal, and a network manager.

VLAN:

Virtual LAN, es un método para crear redes lógicas independientes dentro de una misma red física. Varias VLAN pueden coexistir en un único conmutador físico o en una única red física. Son útiles para reducir el dominio de difusión y ayudan en la administración de la red, separando segmentos lógicos de una red de área local que no deberían intercambiar datos usando la red local (aunque podrían hacerlo a través de un enrutador o un conmutador de capa OSI 3 y 4).

Una VLAN consiste en dos o más redes de computadoras que se comportan como si estuviesen conectados al mismo conmutador, aunque se encuentren físicamente conectados a diferentes segmentos de una red de área local.

Ejemplo:

Creamos las PC's y pingueamos la PC1 tanto con la PC3 como con la PC2 y la PC4. Los paquetes sólo son alcanzados por aquellas PC's con la misma id. Los comandos que utilizamos son:

```
ifconfig eth0 up
vconfig add eth0 N°ID
ifconfig eth0.N°ID 192.168.0 N°PC/24 up
```

Cada uno con su respectivo número que lo identifica.

Administración de Redes Locales - Netkit

```
PC1
From 192.168.0.1 icmp_seq=1 Destination Host Unreachable
From 192.168.0.1 icmp_seq=2 Destination Host Unreachable
From 192.168.0.1 icmp_seq=3 Destination Host Unreachable
From 192.168.0.1 icmp_seq=4 Destination Host Unreachable
From 192.168.0.1 icmp_seq=5 Destination Host Unreachable
From 192.168.0.1 icmp_seq=6 Destination Host Unreachable
From 192.168.0.1 icmp_seq=7 Destination Host Unreachable
From 192.168.0.1 icmp_seq=8 Destination Host Unreachable
From 192.168.0.1 icmp_seq=9 Destination Host Unreachable
From 192.168.0.1 icmp_seq=10 Destination Host Unreachable
^C
--- 192.168.0.2 ping statistics ---
10 packets transmitted: 0 received, 100% packet loss, time 903ms
^C
PC1's ping 192.168.0.3
PING 192.168.0.3 (192.168.0.3) 56(84) bytes of data:
From 192.168.0.1 icmp_seq=1 Destination Host Unreachable
From 192.168.0.1 icmp_seq=2 Destination Host Unreachable
From 192.168.0.1 icmp_seq=3 Destination Host Unreachable
From 192.168.0.1 icmp_seq=4 Destination Host Unreachable
From 192.168.0.1 icmp_seq=5 Destination Host Unreachable
From 192.168.0.1 icmp_seq=6 Destination Host Unreachable
From 192.168.0.1 icmp_seq=7 Destination Host Unreachable
From 192.168.0.1 icmp_seq=8 Destination Host Unreachable
From 192.168.0.1 icmp_seq=9 Destination Host Unreachable
From 192.168.0.1 icmp_seq=10 Destination Host Unreachable
^C
--- 192.168.0.3 ping statistics ---
10 packets transmitted: 0 received, 100% packet loss, time 903ms
^C
PC1's ping 192.168.0.4
PING 192.168.0.4 (192.168.0.4) 56(84) bytes of data:
64 bytes from 192.168.0.4: icmp_seq=1 ttl=64 time=0.77 ms
64 bytes from 192.168.0.4: icmp_seq=2 ttl=64 time=0.350 ms
64 bytes from 192.168.0.4: icmp_seq=3 ttl=64 time=0.333 ms
64 bytes from 192.168.0.4: icmp_seq=4 ttl=64 time=0.354 ms
64 bytes from 192.168.0.4: icmp_seq=5 ttl=64 time=0.372 ms
64 bytes from 192.168.0.4: icmp_seq=6 ttl=64 time=0.764 ms
64 bytes from 192.168.0.4: icmp_seq=7 ttl=64 time=0.844 ms
64 bytes from 192.168.0.4: icmp_seq=8 ttl=64 time=0.574 ms
64 bytes from 192.168.0.4: icmp_seq=9 ttl=64 time=0.595 ms
64 bytes from 192.168.0.4: icmp_seq=10 ttl=64 time=0.802 ms
64 bytes from 192.168.0.4: icmp_seq=11 ttl=64 time=0.769 ms
64 bytes from 192.168.0.4: icmp_seq=12 ttl=64 time=0.506 ms
64 bytes from 192.168.0.4: icmp_seq=13 ttl=64 time=0.500 ms
64 bytes from 192.168.0.4: icmp_seq=14 ttl=64 time=0.572 ms
64 bytes from 192.168.0.4: icmp_seq=15 ttl=64 time=0.544 ms
64 bytes from 192.168.0.4: icmp_seq=16 ttl=64 time=0.883 ms
64 bytes from 192.168.0.4: icmp_seq=17 ttl=64 time=1.541 ms
64 bytes from 192.168.0.4: icmp_seq=18 ttl=64 time=0.318 ms
64 bytes from 192.168.0.4: icmp_seq=19 ttl=64 time=0.380 ms
64 bytes from 192.168.0.4: icmp_seq=20 ttl=64 time=1.00 ms
^C
--- 192.168.0.4 ping statistics ---
20 packets transmitted: 20 received, 0% packet loss, time 1019ms
rtt min/avg/max/mdev = 0.309/1.144/7.570/1.487 ms
PC1:~#
```


Bibliografía:

<http://wiki.netkit.org/man/man7/netkit.7.html>
http://wiki.netkit.org/index.php/Main_Page
[https://es.wikipedia.org/wiki/Conmutador_\(dispositivo_de_red\)](https://es.wikipedia.org/wiki/Conmutador_(dispositivo_de_red))
<https://cloudswxsecure.wordpress.com/2015/03/10/que-es-stp-y-para-que-sirve/>
<https://www.cloudibee.com/network-bonding-modes/>
<https://es.wikipedia.org/wiki/VLAN>