Redes II

Eduardo Grosclaude

2014-08-13

[2015/12/26, 18:07:58- Material en preparación, se ruega no imprimir mientras aparezca esta nota]

Resumen

En este escrito se presenta la descripción y material inicial de la asignatura **Redes II**, para la carrera de Tecnicatura Universitaria en Administración de Sistemas y Software Libre, de la Universidad Nacional del Comahue.

La materia es electiva, cuatrimestral en modalidad presencial y las clases son de carácter teórico-práctico, desarrolladas en forma colaborativa. Está preparada con los objetivos generales de **actualizar y ampliar el instrumental de trabajo en ambientes de redes**.

Índice

Parte I
La asignatura

1. Objetivos

De la carrera

Según el documento fundamental de la Tecnicatura, el Técnico Superior en Administración de Sistemas y Software Libre estará capacitado para:

- Desarrollar actividades de administración de infraestructura. Comprendiendo la administración de sistemas, redes y los distintos componentes que forman la infraestructura de tecnología de una institución, ya sea pública o privada.
- Aportar criterios básicos para la toma de decisiones relativas a la adopción de nuevas tecnologías libres.
- Desempeñarse como soporte técnico, solucionando problemas afines por medio de la comunicación con comunidades de Software Libre, empresas y desarrolladores de software.
- Realizar tareas de trabajo en modo colaborativo, intrínseco al uso de tecnologías libres.
- Comprender y adoptar el estado del arte local, nacional y regional en lo referente a implementación de tecnologías libres. Tanto en los aspectos técnicos como legales.

De la asignatura

- Actualizar y ampliar el instrumental de trabajo en ambientes de redes.

2. Cursado

- Cuatrimestral de 16 semanas, 4 horas semanales, 64 horas totales
- Clases teórico-prácticas presenciales
- Promocionable con trabajos prácticos

3. Contenidos

Contenidos mínimos

- Switching.
- Redes Privadas Virtuales.
- Balance de carga y Alta Disponibilidad en redes.

Programa

1. Switching
   - Red Ethernet, dominio de colisión y dominio de Broadcast
   - Bridges, switches, arquitectura y funcionamiento.
   - VLANs, trunking
   - Wireless
2. Redes Privadas Virtuales
   - Encapsulamiento
   - VPN de nivel 2 y de nivel 3

- Configuración y administración de OpenVPN
- Configuración de ruteo y de firewalling en VPN

3. Alta Disponibilidad y Balance de carga en Redes
   - Arquitecturas redundantes
   - Bonding y modos de configuración
   - Protocolo STP 802.1d
   - Balance de carga, Ruteo por políticas

4. Bibliografía inicial

- C., Zimmerman, Joann Spurgeon, Ethernet switches. Sebastopol, CA: O'Reilly Media, 2013.
- M. Feilner, A. Bämler, N. Graf, and M. Heller, Open VPN building and integrating virtual private networks. Birmingham, U.K.: Packt Pub., 2006.
- T. Bourke, Server load balancing, 1st ed. Beijing; Sebastopol, Calif: O'Reilly, 2001.
- Matthew Gast, 802.11® Wireless Networks, The Definitive Guide. O'Reilly, 2005.
- Rami Rosen, Linux Kernel Networking: Implementation and Theory. Apress, 2013.
- L. Gheorghe, Designing and implementing Linux firewalls and QoS using netfilter, iproute2, NAT, and L7-filter. Birmingham, U.K.: Packt Pub., 2006.
- R. A. Andrade, P. H. Salas, and D. S. Paredes, Tecnología Wi-Fi. Argentina, 2008.

5. Cronograma

Fecha	Semana	Unidad	Trabajo I	Trabajo II
14/8	1	1. Switching		
21/8	2			
28/8	3			
4/9	4			
11/9	5	2. Redes Privadas Virtuales		
18/9	6			
25/9	7			
2/10	8			
9/10	9	Parcial I		
16/10	10	3. Balance de Carga en Redes		
23/10	11			
30/10	12			
6/11	13			
13/11	14	4. Alta Disponibilidad en Redes		
20/11	15			
27/11	16	Parcial II		

Parte II
Switching

1. Ethernet

1. ¿Qué módulos del kernel Linux están controlando las interfaces de red de su equipo? ¿Cuáles comandos permiten conocer las marcas y modelos de las placas instaladas? ¿Qué diferencia hay entre placas y chipsets? ¿Cómo se puede investigar cuál módulo corresponde a cuál marca y modelo de placa de red?
2. ¿Qué es el bus PCI y qué son los números de dispositivos asignados? ¿Cuál es la relación entre el bus PCI y los nombres de dispositivos de red en Linux?
3. ¿Qué información aporta el comando ethtool? ¿Qué clases de placas de red soporta? ¿Qué significa *Link detected* en la salida de este comando?
4. ¿Qué información transporta un frame Ethernet? ¿Qué diferencias existen entre los frames que circulan por una Ethernet cableada y una inalámbrica?
5. ¿Qué diferencias existen entre el diseño de la red Ethernet original, implementada con coaxil, y la que está implementada con cableado en estrella, conectada mediante hubs? ¿Qué diferencias existen entre esta última y una implementada con switches?

Dominios de colisión y de broadcast

1. ¿Qué es un bridge, qué configuración lleva, qué función realiza sobre los frames que circulan por la red, y cómo aprende su información de trabajo?
2. ¿Existen bridges inalámbricos?
3. ¿A qué se llama segmentación?
4. ¿Un hub es equivalente a un bridge? ¿Un router es equivalente a un bridge? ¿Un switch es equivalente a un bridge?
5. ¿Qué es un frame de broadcast? ¿Qué es un frame unicast? ¿Cómo se distingue un frame de broadcast?
6. ¿Qué protocolos utilizan frames de broadcast? ¿Cómo es aprovechado el mecanismo de broadcast por el protocolo ARP? ¿Qué otros usos podría recibir el mecanismo de broadcast?
7. ¿Qué conducta observan los hosts de la red al recibir un frame de broadcast? ¿En qué casos puede resultar esto un problema y por qué? ¿Cuáles son las contramedidas?
8. ¿Qué es un dominio de colisión? ¿Qué es un dominio de broadcast? ¿Qué tipo de dispositivo es el que delimita la frontera entre dos dominios de colisión? ¿Qué tipo de dispositivo es el que delimita la frontera entre dos dominios de broadcast?
9. ¿Qué efecto tiene la introducción de un bridge en un dominio de colisión? El número de equipos en un mismo dominio de colisión ¿se incrementa o se reduce? ¿Qué efecto tiene esto sobre la probabilidad de colisión?
10. ¿Qué diferencia hay entre un paquete de broadcast y un frame de broadcast? Los routers, ¿repiten los frames de broadcasts?
11. En la figura 1, ¿cuáles son los dominios de colisión y de broadcast que pueden distinguirse? Si el host H1 emite un frame de broadcast, ¿qué otros hosts lo reciben? Si el host H4 emite un broadcast, ¿qué otros hosts lo reciben?
12. En la figura 3, ¿cuáles son los dominios de colisión y de broadcast que pueden distinguirse? Si el host H1 emite un frame de broadcast, ¿qué otros hosts lo reciben?
13. El concepto de dominio de colisión, ¿sigue siendo aplicable en la actualidad, cuando prácticamente la totalidad de las redes usan bridging en lugar de repetidores?
14. El uso de bridges, ¿limita el tráfico de broadcast sobre un segmento de red?

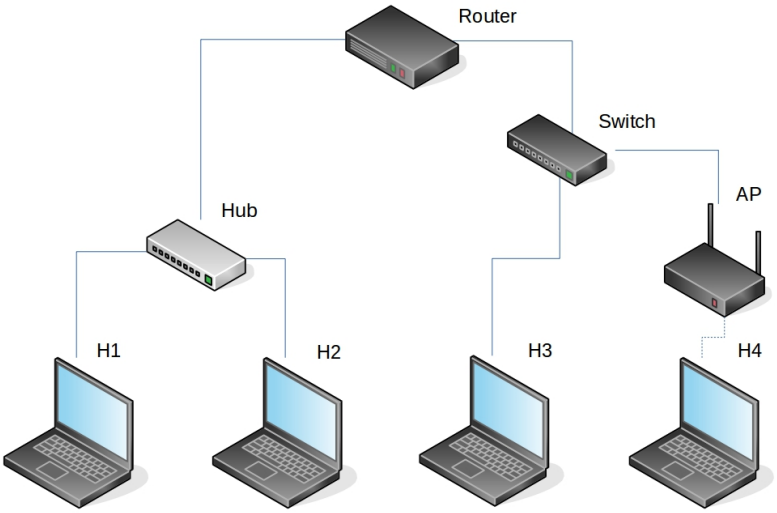

Figura 1: Colisiones y broadcasts

Switches

1. ¿Cuál es la función de un switch?
2. ¿Cuántos dominios de colisión hay en una red implementada sobre un switch? ¿Cuántos dominios de broadcast hay en una red implementada sobre un switch?
3. ¿En qué consiste el proceso de conmutación de un frame? ¿Qué modos de conmutación son los más usados?
4. ¿Cómo aprende su información de trabajo un switch? ¿Dónde almacena dicha información? ¿Cómo accede a esta información?
5. ¿Qué debe hacer un switch ante un frame dirigido a una estación ("unicast") cuya dirección destino figura en sus tablas?
6. ¿Qué debe hacer un switch ante un frame dirigido a una estación ("unicast") que no figura en ninguna de sus tablas? ¿Cómo se llama esta acción?
7. ¿Qué debe hacer un switch ante un frame cuya dirección origen no figura en sus tablas?
8. ¿Qué debe hacer un switch ante un frame de broadcast? ¿Cómo se llama esta acción?
9. En una topología compuesta por un árbol de tres switches como en la figura 3, ¿cómo aprenden los switches las direcciones de los equipos que no están directamente conectados? ¿Cuántos dominios de broadcast aparecen en dicha figura?
10. ¿En qué orden aparecen las direcciones destino y origen en el frame Ethernet? ¿En qué orden aparecen las direcciones destino y origen en los paquetes IP y en los segmentos TCP o UDP? ¿De qué forma aprovechan los switches la diferencia?
11. En la figura 4, si el host H1 establece conexión TCP con H3, ¿quiénes más pueden ver el tráfico entre H1 y H3? Si el host H1 establece conexión TCP con H4, ¿quiénes más pueden ver el tráfico entre H1 y H4?
12. ¿Cuántas redes aparecen, respectivamente, en las topologías de figuras 1, 3 y 4? ¿Cuántos espacios de direccionamiento IP diferentes habrá en cada topología?
13. ¿Existen switches inalámbricos? ¿En qué se parecen y en qué se diferencian un switch cableado y un punto de acceso inalámbrico?

14. ¿Qué diferencias existen entre las conexiones de switches en cascada y apiladas (*stacked*)?
15. ¿Qué velocidad suelen tener los ports de un switch? ¿Qué dicen las normas de cableado respecto de la calidad del cable, su longitud y su forma de instalación, para cada caso?
16. Si los ports de un switch tienen una velocidad determinada, ¿tiene sentido conectar a un port un conjunto de hosts capaces de transmitir, en forma agregada, mayor cantidad de bits por segundo que la velocidad admitida por el port? ¿Qué significa *oversubscription*?
17. En una red compartida implementada con hubs, ¿cuál es el ancho de banda obtenido por cada cliente? ¿Y en una red implementada con switches?
18. En la Fig. 2, supongamos que los hosts H1 y H3 mantienen tráfico entre sí al mismo tiempo que H2 y H4 lo hacen entre sí. ¿Cuál será la velocidad máxima de dicho tráfico si el dispositivo es un hub que funciona a 100Mbps? ¿Y si es un switch que funciona a 100Mbps?
19. En aquellos switches con ports de diferentes velocidades, ¿cuáles podrían ser los usos de los ports de mayor velocidad?
20. ¿Qué son los dispositivos de red administrables? ¿En qué consiste el protocolo SNMP? ¿Qué características operativas de un switch administrable pueden consultarse o modificarse mediante SNMP?
21. ¿Qué características administrables de un switch tienen que ver con la gestión de seguridad en redes?
22. ¿Qué formas de ataque a un switch se conocen? ¿Qué ocurre si el atacante logra inyectar direcciones MAC arbitrarias en un port del switch? ¿Qué ocurre si el atacante logra emitir una cantidad muy grande de frames con diferentes direcciones MAC origen por uno o más ports del switch? ¿Cómo pueden evtarse estos ataques?

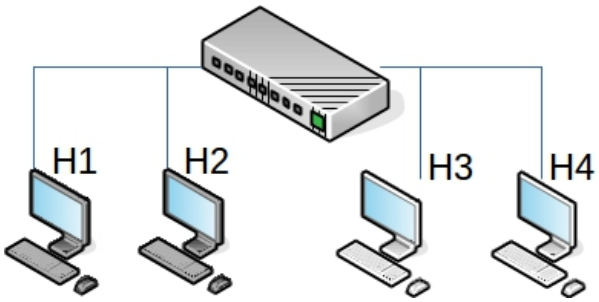

Figura 2: Competencia por el medio

Arquitecturas de redundancia

1. ¿Qué ventajas aporta la redundancia de enlaces en una red LAN? ¿Qué formas de redundancia se pueden implementar? ¿En qué consiste la tolerancia a fallos?
2. ¿Qué ocurre en una red Ethernet con múltiples caminos entre dos hosts? ¿Qué escenarios de fallo se pueden presentar?
3. ¿Cómo se puede detectar una tormenta de broadcasts?
4. ¿Qué ventaja ofrece el protocolo IEEE 802.1d (STP)? ¿Cómo es su modo de operación y qué intenta construir? ¿Qué dispositivos soportan 802.1d?
5. ¿Cómo se elige un dispositivo raíz del árbol STP? ¿Cómo se determinan los caminos al raíz? ¿Qué demora puede tener normalmente la convergencia de STP?
6. ¿Qué datos de configuración o estado, que sean relevantes para 802.1d, y cuya configuración por el administrador sea posible, admiten los bridges y switches? ¿En qué puede influir el administrador de redes sobre la elección del raíz? ¿Cómo se puede aprovechar este hecho?
7. ¿En qué consiste el protocolo RSTP?

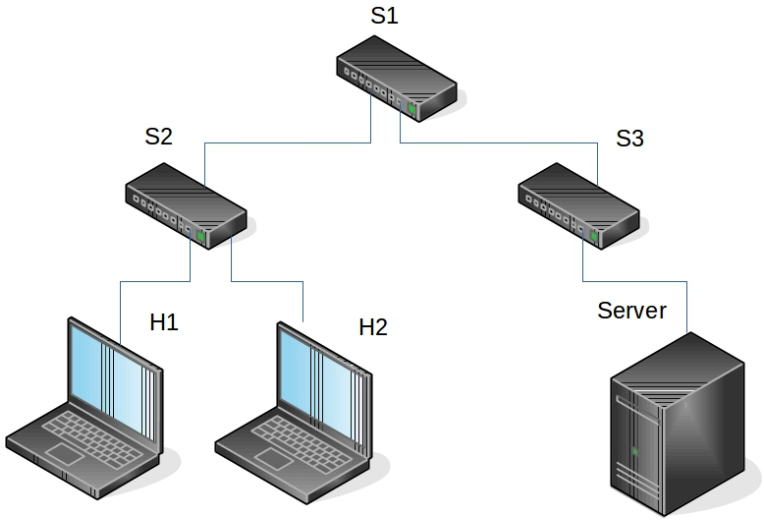

Figura 3: Jerarquía de switches

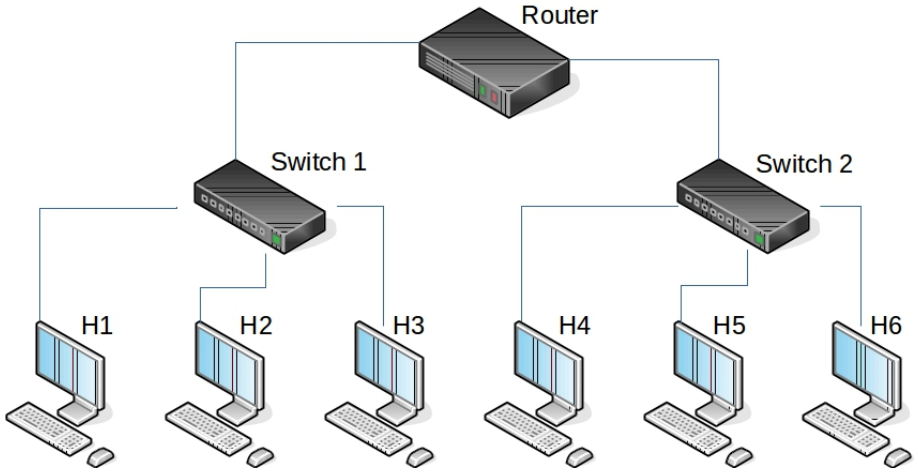

Figura 4: Dominios

2. VLANs

Un diseño clásico de redes de campus consiste en un router que proyecta segmentos de LAN como radios de una estrella (Fig. 5). La función de comunicar los radios, que anteriormente era cumplida por el backbone de la red, en este diseño se logra por la conmutación efectuada por el router, por lo cual suele llamarse de backbone colapsado.

Con este diseño, los dominios de broadcast, y por lo tanto los espacios IP definidos sobre ellos, quedan limitados geográficamente. Si en una zona del campus donde llega un radio de la estrella se necesita ubicar nodos sobre dos dominios de broadcast (porque se desea aislarlos por motivos de seguridad, porque se desea situar equipos sobre dos espacios IP diferentes, o porque se desea limitar la competencia de ambas clases de tráfico por el medio), debe haber dos cableados y deben ocuparse dos bocas del router central.

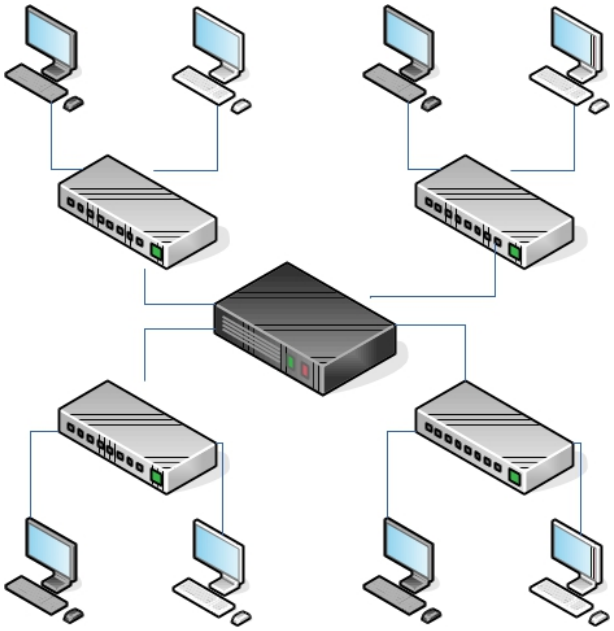

Figura 5: Backbone colapsado

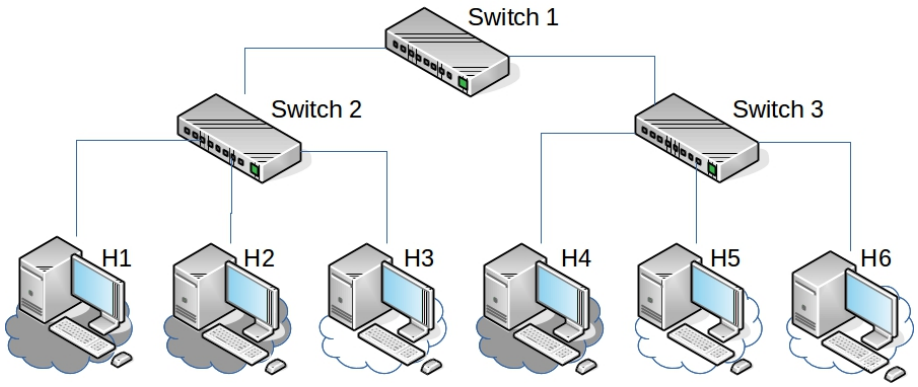

Figura 6: VLANs

Con la funcionalidad avanzada de VLANs provista por algunos switches (y definida en el estándar IEEE 802.1Q), el mismo cableado, y el sistema de switches de llegada, puede usarse para conducir dos o más dominios de broadcast.

- ¿En qué consiste el diseño de red de campus de backbone colapsado? ¿Qué impacto tiene este diseño sobre la posibilidad de distribuir equipos de diferentes clases sobre una misma región de la red?
- ¿Cuál es la finalidad de definir VLANs en un switch?
- ¿Cómo se modifica el formato de frame Ethernet para lograr la capacidad de separar los dominios de broadcast al definir VLANs?
- ¿Qué debe hacer un switch con VLANs definidas al recibir un frame de broadcast sobre una de sus interfaces? ¿Qué debe hacer con un frame unicast?
- En la topología de la figura 6, los tres switches tienen definidas dos VLANs. Los hosts H1, H2 y H4 pertenecen a la VLAN 1, y los hosts H3, H5 y H6 a la VLAN 2.
  - ¿Qué deben hacer los switches con un frame de broadcast recibido desde el host H2?
  - ¿Qué deben hacer los switches con un frame unicast recibido desde el host H5 y dirigido a H6? ¿Lo mismo, pero desde H5 a H3? ¿Qué diferencia en el formato de los frames existe entre un caso y otro, en cada punto del camino?
  - ¿Qué condición deben cumplir los ports que interconectan los switches entre sí para poder distribuir las VLANs por toda la topología?
  - ¿Es posible que una aplicación en el host H5 inicie conexión TCP/IP con una aplicación servidora situada en H2?
- ¿Qué elemento externo es necesario para conectar diferentes VLANs en una misma jerarquía de switches?
- ¿Qué son los switches multicapa o *multilayer*?

3. Estudio de caso I

Una organización desarrolla sus actividades en un campus con dos edificios principales, que alojan las oficinas y talleres. En la organización existen tres áreas principales: Operaciones, Comercialización, e Ingeniería.

Cada área utiliza un servidor de base de datos principal que es propio del área. La organización cuenta además con un servidor de web y de correo electrónico, un servidor de archivos y un servidor de backups, los tres de uso general para las tres áreas. Se desea que todos los puestos de trabajo puedan acceder además a Internet.

Una preocupación especial de la organización es darle protección a los servidores y puestos de trabajo de Ingeniería, que no deben ser accedidos desde las demás áreas.

La planta del campus y sus edificios, con los principales puntos donde se necesita conectividad, es como indica la Fig. 7. En este diagrama se muestran, con diferentes colores, los puestos de trabajo de cada área.

En base a esta información, indique:

- Dónde situaría los servidores mencionados.
- Dónde ubicaría los elementos de conectividad (switches y routers).
- Qué cantidad de puertos debería tener cada uno de estos elementos.
- Con qué medios (cobre, fibra, inalámbricos) vincularía los clientes y los elementos de conectividad.
- Cómo distribuiría direcciones IP para los clientes y servidores.
- Si utilizaría alguna arquitectura de VLANs, cuál, y por qué. En caso afirmativo, cómo relacionaría las VLANs entre sí, qué enlaces entre switches y routers deben ser de trunking y por qué.
- Si utilizaría alguna forma de redundancia, y en caso afirmativo, cuál debe ser el camino normal del tráfico y por qué.

4. Wireless

Tuning

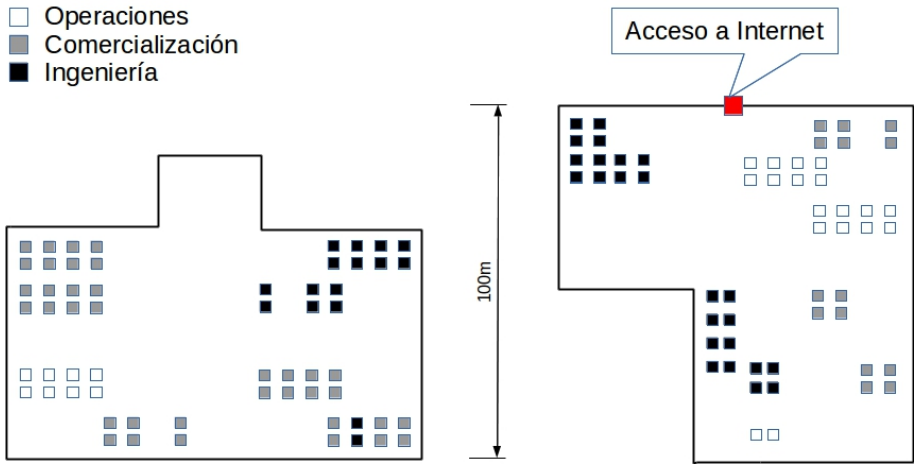

Figura 7: Conectando el campus de una organización

Parámetro	Significado
`Beacon Interval`	Tiempo entre transmisión de beacons
`Fragmentation Threshold`	Los frames más largos que el FT se transmiten usando el procedimiento de fragmentación
`RTS Threshold`	A los frames más largos que el RTS Threshold se les antepone el intercambio RTS/CTS
`Listen Interval`	Las estaciones que entran en ahorro de energía despiertan al transcurrir esta cantidad de interva
`Long Retry Limit`	Cantidad de intentos de retransmisión para frames más largos que el RTS Threshold
`Short Retry Limit`	Cantidad de intentos de retransmisión para frames más cortos que el RTS Threshold

Cuadro 1: Algunos parámetros ajustables y sus efectos

Parte III
Redes Privadas Virtuales

1. Elementos de las VPN

- VPN = *Virtual Private Network*
- Un software específico (cliente-servidor) establece una red *virtual* entre las entidades
- Sólo aquellos miembros autorizados de la red virtual *privada* pueden utilizarla
- *Tunneling* o encapsulamiento de ciertas capas en otras
- Encriptación

Motivación de las VPN

- El software de la organización evoluciona a aplicaciones colaborativas (Groupware, CRM , ERP...)
- Trabajo de todos los colaboradores sobre el espacio digital de la organización
- Diferentes sedes, o colaboradores (domiciliarios o móviles), en diferentes lugares de la región o del mundo
- Soporte universal de Internet en lugar de enlaces arrendados, dedicados
- Requerimientos de seguridad
- Firewalls y NAT requieren *conduits* o perforaciones para acceder a los servicios de las sedes

Modelos de VPN

- Implementadas y administradas por un proveedor de comunicaciones
- Implementadas y administradas por el usuario sobre un transporte común (Internet)

Niveles de implementación

- De nivel de Enlace o de capa 2
  - Único espacio IP y dominio de broadcast
  - La VPN trafica frames
  - Bridging
- De nivel de Red o de capa 3
  - Varias redes con diferentes espacios IP y un servidor VPN que las conecta como backbone
  - La VPN trafica paquetes
  - Ruteo

Algunas tecnologías

- GRE (RFCs 1701, 1702) Mecanismo general de tunneling, no define encriptación
- Protocolos de capa 2: PPTP (Microsoft), L2F, L2TP, L2Sec
- Protocolos de capa 4: SSL/TLS, OpenVPN

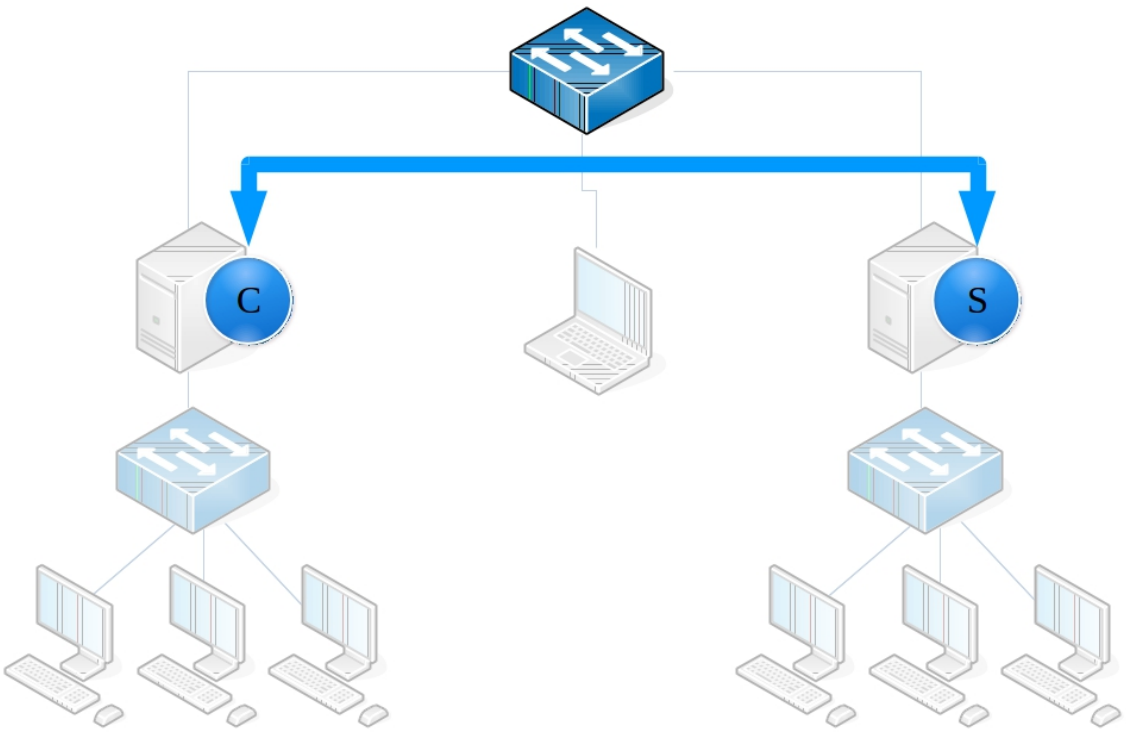

Figura 8: VPN de capa 2, equivalente a un bridge

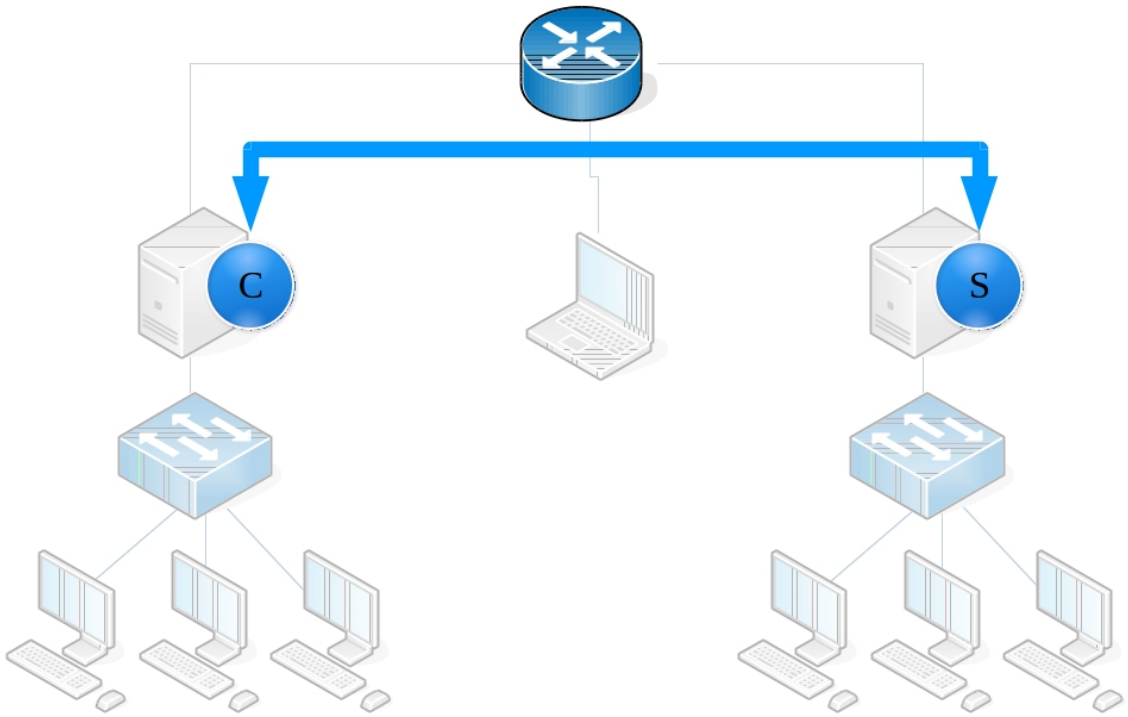

Figura 9: VPN de capa 3, equivalente a un router

2. OpenVPN

- OpenVPN puede funcionar como VPN de capa 2 o de capa 3.
- Proyecto libre compatible con Windows.
- No interoperable con IPSec.
- Dos procesos, cliente y servidor, corren en diferentes hosts en diferentes redes.
- Cliente y servidor establecen una conexión TCP/IP que utilizan como túnel.
- El resto del trabajo lo hace el bridging (en las VPN de capa 2) o el ruteo (de capa 3).
- La arquitectura de OpenVPN (y otras implementaciones de redes virtuales) es un ejemplo claro de uso de interfaces virtuales como los bridges y tun/tap.

Soporte necesario para OpenVPN

Tun/Tap Dispositivos de red virtuales. Trafican unidades de datos entre procesos.

- Un dispositivo tun (de capa 3) trafica paquetes, mientras que un tap (de capa 2) intercambia frames.
- El propósito básico de Tun/Tap es la creación de túneles.
- Un proceso que tiene un dispositivo tun abierto puede escribir un paquete en él. Del mismo modo, un dispositivo tun que recibe un paquete IP lo envía a un proceso de usuario que tiene abierto ese dispositivo.
- Un proceso que tiene un dispositivo tap abierto puede escribir un frame en él. Del mismo modo, un dispositivo tap que recibe un frame lo envía a un proceso de usuario que tiene abierto ese dispositivo.

Bridge Un dispositivo bridge de software funciona exactamente igual que un bridge físico pero entre interfaces del mismo host. Es decir, copia frames de una a otra interfaz de capa 2 (reales o virtuales) efectuando filtrado por dirección MAC e inundando los broadcasts.

- Un bridge Linux *esclaviza* a dos o más interfaces (reales o virtuales).
- Al ser incorporada una interfaz a un bridge, pierde su dirección IP y ésta pasa al bridge.
- Los bridges Linux se controlan desde la línea de comandos con el comando brctl.

Ruteo El kernel utiliza la tabla de ruteo para mover paquetes entre interfaces. Cuando aparece por una interfaz (real o virtual) un paquete de capa 3, se consulta la tabla de ruteo, que dice en cuál interfaz (real o virtual) debe copiarse.

OpenSSL Biblioteca de encriptación de uso general, para proteger la privacidad del tráfico a través de la VPN.

LZO Biblioteca de compresión de datos para mejorar la performance del túnel creado por la VPN.

Funcionamiento de OpenVPN en capa 2

1. En el host cliente:
   - Se crea un dispositivo tapX.
   - Se establece un bridge entre una interfaz local (ethX) y el tapX.
   - La dirección IP de la interfaz local pasa al bridge.
   - El proceso cliente de OpenVPN abre el dispositivo tapX y además establece conexión con el servidor a través de Internet.
   - Cuando el host cliente reciba de su red local un frame por ethX, el bridge filtrará el frame hacia el tapX. El proceso cliente de OpenVPN recibirá el frame y lo derivará por la conexión TCP/IP hacia el servidor.
   - Cuando el proceso cliente reciba un frame a través de la conexión TCP/IP, lo escribirá por la interfaz tapX. El bridge del host cliente recibirá el frame y le aplicará la acción de bridging correspondiente.
2. En el host servidor:
   - Se crea un dispositivo tapX.
   - Se establece un bridge entre una interfaz local (ethX) y el tapX.
   - La dirección IP de la interfaz local pasa al bridge.
   - El proceso servidor de OpenVPN abre el dispositivo tapX y espera pedido de conexión del cliente a través de Internet.
   - Cuando el host servidor reciba de su red local un frame por ethX, el bridge filtrará el frame hacia el tapX. El proceso servidor de OpenVPN lo tomará y lo derivará por la conexión TCP/IP hacia el cliente.
   - Cuando el proceso servidor reciba un frame a través de la conexión TCP/IP, lo escribirá por la interfaz tapX. El bridge del host cliente recibirá el paquete y le aplicará la acción de bridging correspondiente.

Funcionamiento de OpenVPN en capa 3

1. En el host cliente:
   - Se crea un dispositivo tunX.
   - El proceso cliente de OpenVPN abre el dispositivo tunX y además establece conexión con el servidor a través de Internet.
   - El tunX local recibe dirección IP A. Queda establecido un túnel entre procesos cliente y servidor cuyos extremos tienen direcciones IP A y B.
   - Se establece una ruta que dirige los paquetes destinados a la red Y via tunX (próximo salto: B).
   - Cuando el host cliente reciba de su red local, o de procesos locales, un paquete destinado a la red Y, lo escribirá en la interfaz de red virtual tunX. El proceso cliente de OpenVPN lo recibirá y lo derivará por la conexión TCP/IP hacia el servidor.

- Cuando el proceso cliente reciba un paquete a través de la conexión TCP/IP, lo escribirá por la interfaz tunX. El kernel del host cliente recibirá el paquete y le aplicará las reglas de ruteo correspondientes.

2. En el host servidor:
   - Se crea un dispositivo tunX.
   - El proceso servidor de OpenVPN abre el dispositivo tunX y espera pedido de conexión del cliente a través de Internet.
   - El tunX local recibe dirección IP B. Queda establecido un túnel entre procesos cliente y servidor cuyos extremos tienen direcciones IP A y B.
   - Se establece una ruta que dirige los paquetes destinados a la red X via tunX (próximo salto: A).
   - Cuando el kernel del host servidor reciba de su red local, o de procesos locales, un paquete destinado a la red X, lo escribirá en la interfaz de red virtual tunX. El proceso servidor de OpenVPN tomará el paquete y lo derivará por la conexión TCP/IP hacia el cliente.
   - Cuando el proceso servidor reciba un paquete a través de la conexión TCP/IP, lo escribirá por la interfaz tunX. El kernel del host servidor recibirá el paquete y le aplicará las reglas de ruteo correspondientes.

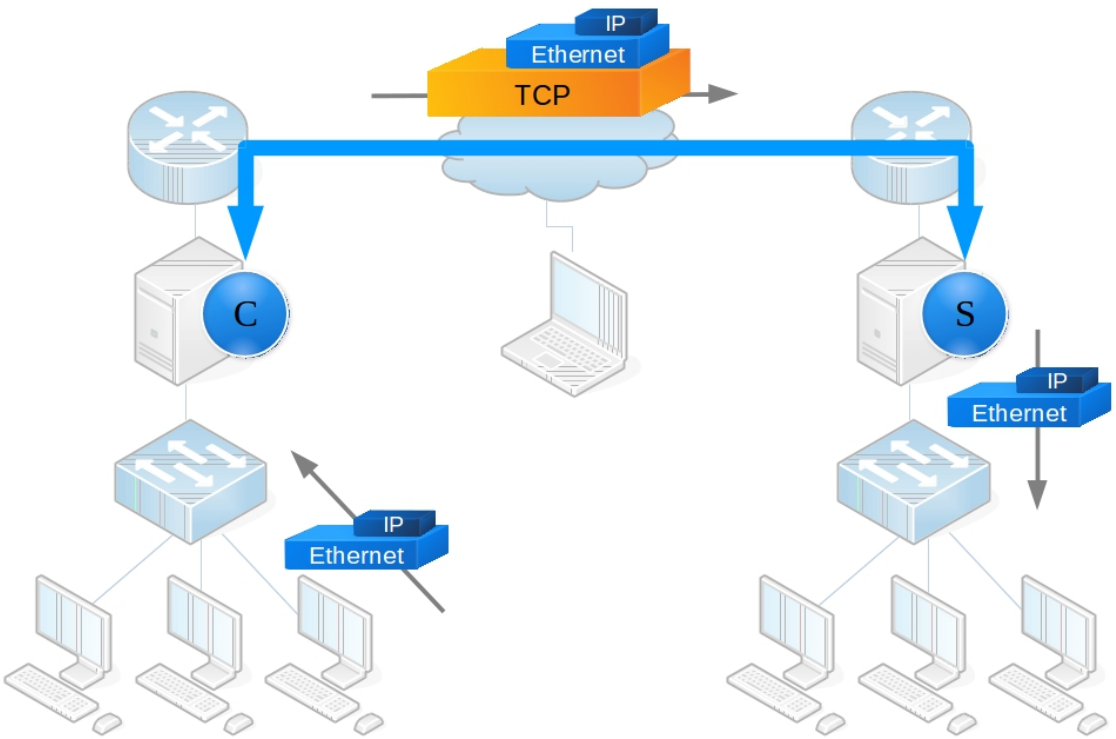

Figura 10: VPN de capa 2, trafica frames

3. Configuración de OpenVPN

Autenticación mediante clave privada

- PKI = Public Key Infrastructure
- La seguridad en redes comprende problemas como la confidencialidad y la integridad de los mensajes, y la autenticación de los usuarios o dispositivos

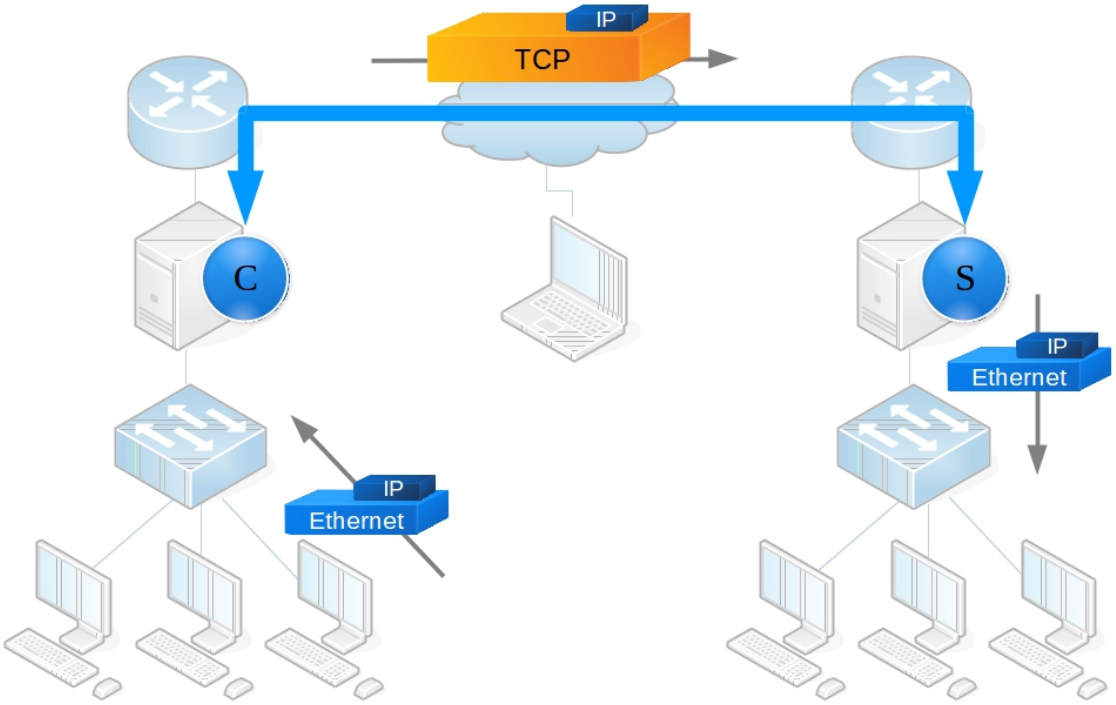

Figura 11: VPN de capa 3, trafica paquetes

- Clave simétrica, con secreto compartido, vs. Claves asimétricas
- Dificultad de la distribución de claves, solución: claves asimétricas
- Cada usuario tiene $K+$ = Clave pública, $K-$ = Clave privada
- Dos escenarios de uso:
  - Cuando A encripta con la $K+$ de B:
    - A envía a B en forma segura, nadie puede leer el mensaje
    - Sólo B puede desencriptarlo con su $K-$
    - Se asegura la confidencialidad
  - Cuando A encripta con su propia $K-$:
    - Cualquiera puede leer el mensaje con la $K+$ de A
    - Pero sólo A puede haberlo escrito, con su $K-$ (firma digital)
    - Se aseguran la integridad y la autenticidad
- El problema al utilizar la $K+$ de A que hemos recibido es *asegurar que es realmente la de A*
- Se resuelve mediante autoridades de certificación (CA), entidades confiables
- La clave pública de A, junto con la identidad de A, firmada digitalmente por una CA, es un *certificado* de A
- Conexión segurizada por PKI
  1. Fase de autenticación mutua mediante intercambio de certificados, validando los certificados según la CA.
  2. Usando las $K+$ recibidas, las partes acuerdan en forma segura una clave simétrica que sirve para esta sesión.
  3. El resto de la comunicación se encripta usando esta clave simétrica (métodos DES, 3DES, AES).

Crear una Autoridad de Certificación

En el servidor de VPN, copiar el directorio `/usr/share/doc/openvpn/examples/easy-rsa` y todos sus contenidos a /etc/openvpn

```
cd /etc/openvpn/easy-rsa/2.0
vi vars
. vars
./clean-all
./build-dh
./build-ca
./build-key-server servidor
./build-key cliente1
./build-key cliente2
# Migrar los certificados
# En el server:
# ca.crt ca.key dh1024.pem r3.crt r3.key
#
# En el cliente:
# ca.crt r1.crt r1.key
```

Documentación online

- Proyecto Lihuén[1]
- OpenVPN HOWTO[2]

Vinculación de LANs mediante Openvpn

- Instalar OpenVPN en los nodos de frontera de las LANs.
- Decidir cuál nodo será el servidor. Debe tener dirección IP pública accesible desde los clientes. No es necesaria la traducción DNS salvo que se trate de una IP dinámica.
- Preparar una autoridad de certificación (por ejemplo, en el servidor), y certificados para el servidor y para los clientes.
- Se pueden utilizar los modelos de archivos de configuración existentes en `/usr/share/doc/openvpn/examples`.
- En el directorio `/etc/openvpn` del servidor preparar el archivo de configuración `server.conf`, y en los clientes, `client.conf`. Al arrancar, OpenVPN lee todos los archivos con extensión .conf que existan en ese directorio, y crea un proceso por cada uno, con la configuración que contengan.
- Modificar los parámetros de configuración:
  1. Cliente o servidor
  2. Dirección y port, nodo local o remoto
  3. Protocolo UDP o TCP
  4. Device Tun o Tap, según la capa de operación de la VPN
  5. Nombres de los archivos de clave y certificado
  6. Archivo de log, útil para debugging de la configuración y operación
  7. Rutas a redes propias que deban ser inyectadas en el peer
- Arrancar o detener la VPN con el comando `/etc/init.d/openvpn [start|stop]`.

[1] http://lihuen.linti.unlp.edu.ar/index.php?title=Configurando_Redes_Privadas_Virtuales_con_OpenVPN
[2] http://openvpn.net/index.php/open-source/documentation/howto.html

OpenVPN de nivel de Red

Al configurar OpenVPN en capa 3, o nivel de Red, la configuración de OpenVPN puede asumir la creación de rutas entre las redes clientes y las de detrás del servidor. Estas rutas aparecen y desaparecen en la tabla de ruteo de los nodos extremos de la VPN según se activa o desactiva el proceso de OpenVPN.

- Para que el cliente conozca las redes detrás del servidor, éste le inyecta las rutas correspondientes con la opción push de la configuración.
- El servidor instala rutas a las redes detrás de los clientes si se especifican con la directiva route de la configuración del servidor. Además, para cada cliente, debe haber un archivo con la directiva iroute dentro del subdirectorio /etc/openvpn/ccd.

OpenVPN de nivel de Enlace

Al configurar OpenVPN en capa 2, o nivel de Enlace, la configuración debe incluir la administración del bridge entre la interfaz de red local y el dispositivo Tap que es utilizado por la VPN. Los frames que llegan a la interfaz de la red local serán copiados por el bridge en el Tap y seguirán su camino a través de la VPN.

Para la administración del bridge hay dos estrategias básicas:

- El bridge puede tener una definición estática, y ser siempre activado cada vez que arranca el equipo, y sólo ser desactivado cuando se detiene el equipo.
- El bridge puede ser activado y desactivado cuando se inicia y detiene OpenVPN.

En el primer caso, la existencia del bridge es completamente independiente de la activación de la VPN. Para esta opción existe una forma de configuración del bridge que es dependiente del sistema operativo (ver Anexo B).

Para la segunda opción, es conveniente utilizar los scripts de creación de tap y montado de bridges que se muestran en el Anexo D y dispararlos con las directivas up y down de la configuración de OpenVPN. Por ejemplo:

```
...
ca ca.crt
cert server.crt
key server.key
dh dh1024.pem
...
up /etc/openvpn/bridge-up
down /etc/openvpn/bridge-down
```

4. Temas de práctica

El laboratorio de la Fig. 12 se encuentra implementado sobre máquinas virtuales.

El equipo virtual $R2$, que es gateway default de los otros routers, simula la Internet, en el sentido de que descarta todo tráfico dirigido a redes privadas. Por este motivo las redes con prefijo 10.0.0.0/8, del laboratorio, no son accesibles una desde la otra, siendo necesaria una solución de Red Privada Virtual. La configuración de este equipo $R2$ no debe ser modificada.

- Verifique que las direcciones, ruteo default y topología corresponden a la figura.
- Verifique servicio de nombres, ping y traceroute a www.google.com.
- Ping a los routers de la topología.
- Ping a los hosts de la misma LAN.
- Ping a los hosts de la LAN opuesta.

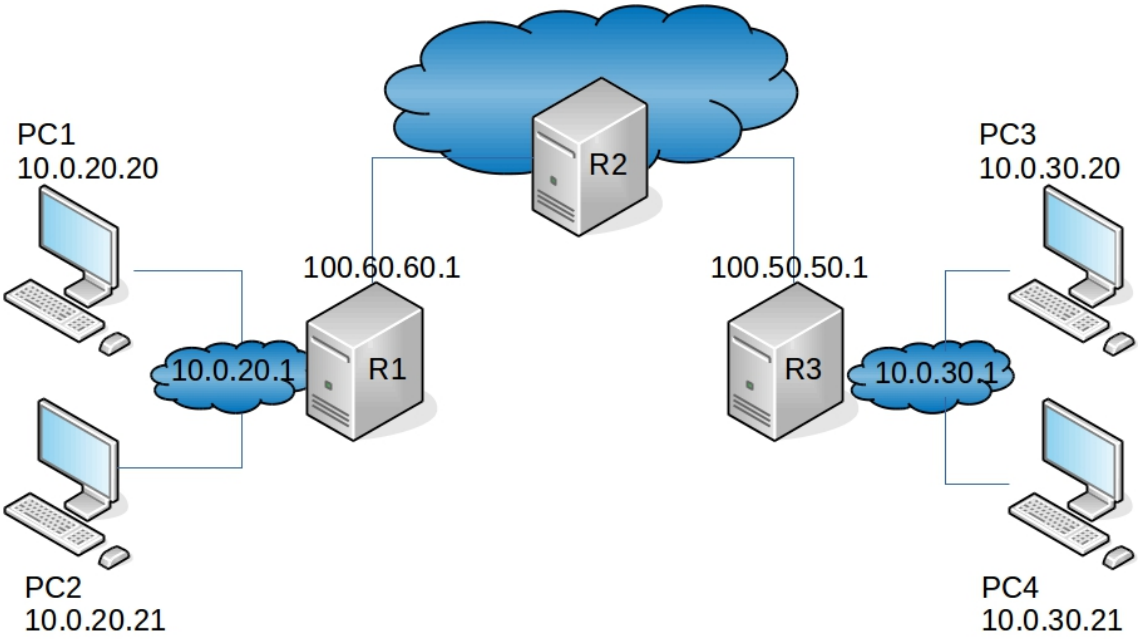

Figura 12: Laboratorio 1 OpenVPN

- Compruebe navegación en ambiente de caracteres con el comando `lynx http://www.google.com`.
- Compruebe que puede correr el web server Apache en cualquiera de las PCs.
- Compruebe navegación en ambiente de caracteres desde una PC al servidor Apache del otro nodo de la misma LAN.

Una vez que esté familiarizado con el escenario:

1. Implementar una VPN de capa 3 entre ambas LANs. Verifique las interfaces existentes en los nodos cabecera y el contenido de sus tablas de ruteo. Comprobar que los clientes de una de las redes pueden utilizar recursos (ssh, servidor http) de clientes de la otra. Instale servicios basados en broadcast (como Samba, sobre el protocolo SMB) y observe desde qué lugares pueden accederse.
2. Modificar el direccionamiento IP del laboratorio para que todos los clientes reciban direcciones en la misma subred e implementar una VPN de capa 2. Comprobar que ambas redes forman un único dominio de broadcast. Verifique el comportamiento de los protocolos basados en broadcast como SMB.
3. ¿Usaría transporte UDP o TCP para una VPN en capa 2?

Parte IV
Balance de carga y Alta Disponibilidad en redes

1. Bonding

Acoplamiento de dos o más interfaces de red, creando una interfaz virtual capaz de funcionar en diferentes modos. Se conoce con diferentes nombres (*channel bonding, teaming, link aggregation*). Los modos de configuración de bonding definen el comportamiento del conjunto de interfaces y cumplen con diferentes objetivos. Algunos modos proporcionan tolerancia a fallos mediante redundancia; otros aumentan el ancho de banda disponible por agregación de enlaces.

Los modos active-backup, balance-tlb, y balance-alb no requieren una configuración especial en los switches a los cuales está conectado el bond. Sin embargo, los modos 802.3ad, balance-rr, balance-xor y broadcast requieren capacidades especiales del switch definidas en documentos IEEE.

Modo 0 (balance-rr) Este modo transmite frames en orden secuencial desde el primer esclavo disponible hasta el último. Si un bond tiene dos interfaces reales, y llegan simultáneamente dos frames a ser enviados desde la interfaz bond, el primero será transmitido por el primer esclavo; el segundo frame, por el segundo esclavo; el tercer frame será transmitido por el primer esclavo, etc. Esto provee a la vez balance de carga y tolerancia a fallos.

Modo 1 (active-backup) Este modo coloca a las interfaces esclavas en estado de backup, y sólo se activará una de ellas si se pierde el link de la interfaz activa. En este modo, sólo hay un esclavo activo en el bond en cada momento. Sólo se activa un esclavo diferente si falla el esclavo activo. Este modo provee tolerancia a fallos.

Modo 2 (balance-xor) Transmite basándose en una fórmula XOR. Se computa la operación XOR entre la dirección MAC de origen y la de destino, módulo la cantidad de esclavos (es decir, se toma el resto de dividir por la cantidad de esclavos). Este procedimiento tiene el efecto de seleccionar siempre el mismo esclavo para cada dirección MAC destino. Provee balance de carga y tolerancia a fallos.

Modo 3 (broadcast) Este modo transmite todos los frames por todas las interfaces esclavas. Es el menos usado, sólo para propósitos específicos, y sólo provee tolerancia a fallos.

Modo 4 (802.3ad) Este modo se conoce como el modo de agregación dinámica de enlaces (Dynamic Link Aggregation). Crea grupos de agregación que comparten la misma velocidad y modos de duplexing. Este modo requiere un switch que soporte la norma IEEE 802.3ad (Dynamic Link).

Modo 5 (balance-tlb) Llamado balance de carga adaptativo en transmisión. El tráfico de salida se distribuye de acuerdo a la carga actual y es encolado en cada interfaz esclava. El tráfico entrante es recibido por el esclavo actual.

Modo 6 (balance-alb) Este modo es el de balance de carga adaptativo. Esto incluye balance-tlb y balance de carga en recepción (rlb) para tráfico IPv4. El balance de carga en recepción se logra por negociación ARP. El driver de bonding intercepta las respuestas ARP enviadas por el servidor y sobreescribe la dirección MAC origen con la dirección MAC única de uno de los esclavos en el bond, de forma que diferentes clientes usen diferentes direcciones MAC para dirigirse al server.

Detección de eventos

Los modos que ofrecen tolerancia a fallos necesitan algún mecanismo para detectar eventos de caída de las interfaces de red (NIC) locales, los enlaces, o las NICs de los extremos opuestos de los vínculos. Ante la detección de un evento de fallo, el bond fuerza la conmutación a otra interfaz, que entonces se convierte en primaria. Esta acción se llama *failover*.

Para la detección de eventos hay dos opciones posibles.

1. **MII (Medium Independent Interface)**. Especificación que cumplen la mayoría de las NICs modernas, que presenta datos de link activo o inactivo en forma independiente de la implementación del medio conectado. Se deben configurar los parámetros `bond-miimon` (intervalo de revisión del estado del link), `bond-downdelay` (tiempo desde que se detecta fallo hasta que se da de baja la interfaz) y `bond-updelay` (tiempo para volver a poner en servicio la interfaz una vez que vuelve el link a estado activo).
2. **ARP**. Se establece por configuración una cantidad de direcciones IP de control en la red local, y las interfaces esclavas del bond emiten consultas ARP periódicas a estas direcciones. Cuando un bond advierte que una de sus interfaces esclavas no ha emitido tráfico en una cantidad de tiempo dado, la considera caída. Se deben configurar los parámetros `bond-arp-interval` (intervalo entre emisión de ARP) y `bond-arp-ip-target` (lista de IPs confiables). Este control permite detectar modos de fallo de la interfaz, pero no del enlace o del peer, por lo cual se agrega un mecanismo más: si además se configura el parámetro `bond-arp-validate`, cuando se deja de recibir respuesta al tráfico ARP por la interfaz activa durante una cantidad de tiempo, configurable, se considera que ha caído el enlace.

Configuración en Debian

Configuración con MII

```
# /etc/network/interfaces
# The loopback network interface
auto lo
iface lo inet loopback
auto eth0
iface eth0 inet manual
       bond-master bond0
       bond-primary eth0 eth1

auto eth1
iface eth1 inet manual
       bond-master bond0
       bond-primary eth0 eth1

auto bond0
iface bond0 inet static
       address 192.168.1.15
       netmask 255.255.255.0
       network 192.168.1.0
       gateway 192.168.1.1
       bond-slaves eth0 eth1
       bond-mode active-backup
       bond-miimon 100
       bond-downdelay 200
       bond-updelay 200
```

Configuración con ARP

```
# /etc/network/interfaces
# The loopback network interface
auto lo
iface lo inet loopback
```

```
# No se especifican las auto ethX

auto bond0
iface bond0 inet static
       address 10.1.1.1
       netmask 255.255.255.0
       network 10.1.1.0
       bond_primary eth0
       slaves eth0 eth1
       bond-mode active-backup
       bond-arp-interval 2000
       bond-arp-ip-target 10.1.1.2 10.1.1.3
```

Temas de práctica

La topología de la Fig. 13 comprende un nodo llamado **switch**, y tres nodos conectados a él. El switch tiene cinco interfaces, cada una conectada a un enlace diferente. Dos de los nodos tienen dos interfaces cada uno, y el tercero una sola. Cada interfaz de los nodos host1 a host3 está conectada a su propio enlace. Los enlaces se denominan, de izquierda a derecha en el diagrama, link1 a link5.

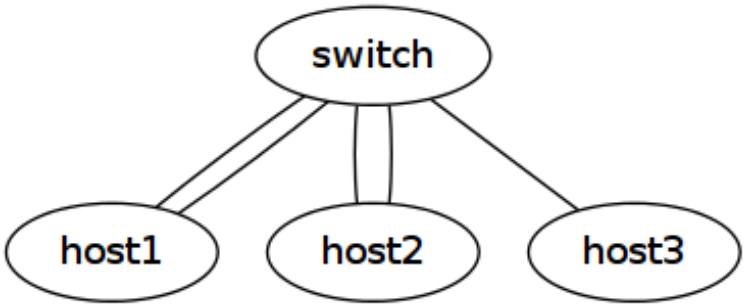

Figura 13: Configuración del laboratorio de bonding

> ¡Ver las Notas sobre el laboratorio en la sección más abajo!

1. Configure en forma estática la red del nodo denominado switch implementando un bridge que esclavice a todas sus interfaces. Dé direcciones en el mismo dominio de broadcast a los demás nodos. Compruebe llegada por ping. Observe el tráfico que pasa por las interfaces del switch con el comando `tcpdump -i ethX`. El nodo virtual **switch**, ¿resulta un buen modelo de un switch de hardware? ¿Necesita tener una dirección IP?
2. En los nodos host1 y host2, instale el módulo que permite esclavizar las interfaces con `dpkg -i /hostlab/ifenslave-...deb`.
3. Configure el nodo host1 indicando en `/etc/modules` que debe ser cargado el módulo `bonding` al arranque. Configure la red del mismo nodo, ligando ambas interfaces mediante un bond en modo *active-backup*. ¿Qué capacidades tiene este modo?
4. Mantenga un ping desde host3 a host1, observando el tráfico por las interfaces del switch. Simule la caída del vínculo con host1. ¿Se interrumpe el ping? ¿Sigue pasando el tráfico por las mismas interfaces?
5. Configure el nodo host2 del mismo modo que en el punto 3, pero con el bond en modo *balance-rr*. ¿Qué capacidades tiene este modo?
6. Ejecute el mismo experimento del punto 4 pero desde host3 a host2.

Notas sobre el laboratorio

- Es necesario instalar el paquete `ifenslave`. En nuestro laboratorio se ha provisto un ejemplar del archivo DEB correspondiente en el directorio `/hostlab`.
- Es necesario indicar el modo, técnica de detección de eventos, y parámetros, en el archivo `/etc/modules` (a pesar de indicarlo en `/etc/network/interfaces`), de la siguiente manera:

```
alias bond0 bonding
options bonding mode=1 miimon=100 downdelay=200 updelay=200
```

  (detección por MII) o bien:

```
alias bond0 bonding
options bonding mode=1 arp_interval=1000 arp_ip_target=10.1.1.1
```

  (detección por ARP).
- Al probar sistemas de Alta Disponibilidad, una parte muy importante es la inyección de fallas. En este laboratorio, cuando se usa la técnica de detección de fallas **MII**, la tecnología de virtualización utilizada no permite replicar correctamente los eventos de caída de la interfaz, el enlace o el *peer* (la interfaz aparece siempre conectada)[3]. La mejor aproximación que hemos logrado a la simulación de la falla es cuando se elige detección de fallas por **ARP**.
- Para simular la caída de un vínculo se sugiere detener el proceso UML que simula el enlace. Logramos esto buscando entre todos los procesos `uml_switch` de la máquina host aquel relacionado con el nombre del enlace. Por ejemplo, para simular la caída del link1:

```
# ps f | grep uml_switch
23137 pts/1   S+     0:00 \_ grep uml_switch
...
19381 pts/1   S      0:00 /home/rii/netkit/bin/uml_switch -hub -unix
    /home/oso/.netkit/hubs/vhub_oso_link1.cnct
# kill -STOP 19381
```

  El proceso se hace continuar (se restablece el vínculo virtual) con `kill -CONT 19381`.
- Cada vez que hagamos un cambio de configuración será preferible, antes de hacer una nueva prueba, bajar y volver a levantar el laboratorio completo con `lhalt` y `lstart`.
- La interfaz virtual bond puede monitorearse mirando el pseudo archivo correspondiente en el directorio /proc:

```
# cat /proc/net/bonding/bond0
Ethernet Channel Bonding Driver: v3.2.5 (March 21, 2008)

Bonding Mode: fault-tolerance (active-backup)
Primary Slave: eth0
Currently Active Slave: eth0
MII Status: up
MII Polling Interval (ms): 0
Up Delay (ms): 0
Down Delay (ms): 0
ARP Polling Interval (ms): 2000
ARP IP target/s (n.n.n.n form): 10.1.1.2, 10.1.1.3

Slave Interface: eth0
MII Status: up
```

[3] Debido a que el kernel UML utilizado en Netkit no implementa MII.

```
Link Failure Count: 0
Permanent HW addr: a6:8a:d7:63:1e:49

Slave Interface: eth1
MII Status: up
Link Failure Count: 0
Permanent HW addr: e6:c9:c8:a3:eb:72
```

Referencias

- https://www.kernel.org/doc/Documentation/networking/bonding.txt
- http://www.linuxfoundation.org/collaborate/workgroups/networking/bonding
- http://www.cyberciti.biz/tips/debian-ubuntu-teaming-aggregating-multiple-network-connections.html

2. Ruteo por políticas

Consideremos el caso general de una organización con dos o más accesos a Internet a través de diferentes proveedores.

- Los enlaces extra pueden haberse contratado con la idea de seguir obteniendo acceso en caso de que uno de los proveedores falle (Fig. 14). Es decir, el objetivo puede ser obtener **tolerancia a fallos**.
- También puede ser que la organización desee aumentar la capacidad de su vínculo con Internet, utilizando todos los enlaces. En este caso, el objetivo se definiría como lograr **balance de carga**.

¿Cómo administrar los enlaces para satisfacer el requerimiento de tolerancia a fallos? Habrá que disponer de algún mecanismo para detectar fallos, cuando ocurran, y algún mecanismo para conmutar a un enlace que siga activo. Una posibilidad inmediata es funcionar en modo *activo-standby* (con sólo uno de los enlaces activo y sin aprovechar los demás). Sin embargo, esta solución no resulta eficiente, ya que, la mayor parte del tiempo, se estará pagando por un recurso que no se utiliza. Lo ideal, entonces, será considerar alguna forma de distribución de tráfico por todos los enlaces disponibles. Es decir que, aunque se trate de objetivos diferentes, el de tolerancia a fallos lleva naturalmente al de balance de carga.

Este objetivo combinado tiene ciertas restricciones. Diferentes proveedores tendrán asignadas direcciones de redes diferentes. Si quisiéramos emplear varios enlaces en forma agregada para una misma conexión TCP, los segmentos que salen de la organización hacia un mismo destino llegarían a ese destino con direcciones IP origen diferentes (aquellas que resulten del ruteo realizado por los diferentes proveedores), y no podrían ser reconocidos como parte de la misma conexión.

En consecuencia, por la naturaleza de los protocolos de transporte confiable de Internet, en general no será posible utilizar el total del ancho de banda agregado de todos los enlaces en una sola conexión[4].

- Es decir, si se tienen dos accesos de 1 Mbps cada uno, no es posible utilizar ambos enlaces agregados para efectuar una única transferencia de datos a 2 Mbps en una sola conexión.
- En cambio, sí es posible distribuir la totalidad de la carga entre todos los vínculos, destinando una parte de las conexiones por cada enlace.

Ruteo

La estrategia de cuál enlace se utilizará en cada momento implica una decisión de ruteo. Como sabemos, el router toma decisiones paquete a paquete, observando la dirección destino y basándose en la información contenida en una tabla de ruteo. Esta tabla de ruteo contiene varias rutas, indicando:

[4] Aunque IETF desarrolla una versión *Multipath* de TCP, por ahora en estado experimental (RFC 6824).

Destination	Gateway	Genmask	Iface
10.0.2.0	0.0.0.0	255.255.255.0	eth0
10.0.3.0	0.0.0.0	255.255.255.0	eth1
170.210.83.20	10.0.2.6	255.255.255.255	eth0
10.0.4.0	10.0.3.8	255.255.255.0	eth1
0.0.0.0	10.0.2.1	0.0.0.0	eth0

Cuadro 2: Esquematización de una tabla de ruteo típica

- Cuáles son las redes directamente conectadas, y a qué interfaz está conectada cada una. Esta información aparece en la tabla automáticamente al configurar las direcciones IP de las interfaces.
- Opcionalmente, otras redes no directamente conectadas, pero conocidas (o hosts conocidos), y la dirección IP del router al cual debe enviarse el tráfico con destino a esas redes (o hosts). La tabla de ruteo puede no contener ninguna de estas rutas específicas opcionales.
- A qué router debe enviarse el tráfico que no corresponda a redes directamente conectadas ni conocidas (ruta de *default gateway* o ruta por defecto). Esta ruta técnicamente no es obligatoria, pero en la práctica es casi siempre indispensable.

Por ejemplo, una tabla de ruteo típica, esquematizada, podría contener rutas como las del Cuadro 2. Esta tabla de ruteo contiene cinco reglas que dicen:

1. La red 10.0.2.0/24 está directamente conectada a la interfaz eth0.
2. La red 10.0.3.0/24 está directamente conectada a la interfaz eth1.
3. El host 170.210.83.20 debe accederse a través del gateway 10.0.2.6.
4. La red 10.0.4.0/24 debe accederse a través del gateway 10.0.3.8.
5. Todo otro tráfico debe ser enviado al gateway default, 10.0.2.1, y desde allí será reenviado a su dirección destino.

Acceso dividido

El ruteo por defecto, es decir, el uso de un *default gateway* como en el ejemplo anterior, permite indicar solamente **un** camino de salida, de manera que, para distribuir el tráfico entre dos o más enlaces, se necesitará alguna técnica que permita decidir por cuál enlace se dirigirá el tráfico en cada caso.

En general, cuando se necesita establecer alguna decisión administrativa sobre cómo se utilizan las rutas disponibles, se dice que estamos frente a una situación de ruteo por políticas. Una política es simplemente un conjunto de decisiones.

Lo que necesitamos, es precisamente, herramientas para imponer ruteo por políticas.

El escenario de múltiples accesos a Internet suele llamarse *Split Access* (acceso dividido). Los dos problemas del Acceso Dividido son:

1. Hacer que el tráfico ingresado al sistema desde Internet por una interfaz vuelva por la misma interfaz.
2. Lograr distribución del tráfico, o balance de carga.

Las principales herramientas de software para conseguir estos objetivos son el paquete `iproute2` y el comando `iptables`. Con el primero definiremos una estructura de ruteo especial para el caso de uso de Split Access y con el segundo manipularemos el tráfico para aplicar diferentes políticas de ruteo.

Paquete iproute2

Paquete de configuración de red integral, que reemplaza a utilitarios como `ifconfig`, `route`, `netstat`. Permite configurar interfaces, información de ARP, ruteo por políticas, túneles, etc. Consiste de dos comandos: `ip`, que controla estado de interfaces y configuración de IPv4 e IPv6, y `tc`, que administra aspectos de calidad de servicio (*Quality of Service*, QoS). Ejemplos en Cuadro 3.

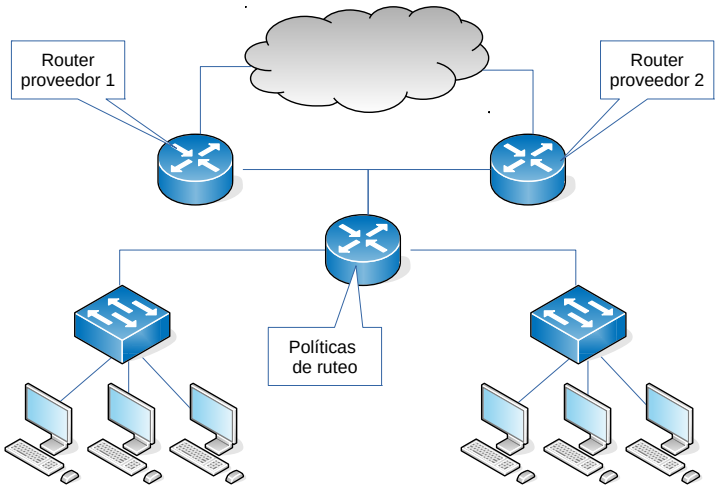

Figura 14: Escenario de acceso dividido

Comando iptables

Es la interfaz de usuario al subsistema Netfilter[5] del kernel Linux, que permite modificar la forma como los paquetes IP atraviesan el kernel en su tránsito entre una interfaz de entrada, procesos locales, e interfaz de salida.

Según el caso, a cada paquete pueden aplicársele las reglas configuradas por el usuario y agrupadas en diferentes **cadenas** (`PREROUTING`, `INPUT`, `OUTPUT`, `POSTROUTING`, `FORWARD`). Las reglas se almacenan en **tablas** que se aplican secuencialmente dentro de cada cadena. Las cadenas se disponen en diferentes lugares del recorrido del paquete por el kernel, y el usuario puede crear las suyas propias.

Las cadenas, entonces, se corresponden con el tipo de paquete según el tránsito que haga a través del host. Las tablas, con el tipo de acción que se desea aplicar. Cada tabla tiene una aplicación específica:

Tabla `filter` Aplica reglas de filtrado para implementar políticas de firewalling. Es la tabla por defecto.

Tabla `nat` Ejecuta edición de direcciones. Por ejemplo, cuando se aplica NAT (*Network Address Translation*), convierte la dirección de salida de un paquete a una dirección propia.

Tabla `mangle` Permite efectuar cualquier modificación de cualquier zona del paquete, además de afectar datos que acompañan al paquete durante su viaje por el kernel.

Tabla `raw` Implementa excepciones al seguimiento de conexiones.

Tabla `security` Implementa control de acceso obligatorio y permite separar las acciones que puede tomar el usuario de las que impone el sistema.

Netfilter diferencia entre paquetes **destinados al host** (la dirección IP destino del paquete es alguna de las propias), **originados en el host** (la dirección origen es una dirección propia), o **en tránsito** (ni la dirección origen ni la dirección destino son propias). Este último caso es el que se presenta habitualmente en un router.

Ingreso A todos los paquetes ingresantes, antes de que el proceso de ruteo determine si son o no dirigidos al host, se les aplican las reglas contenidas en las tablas `mangle` y `nat` de la cadena `PREROUTING`.

Egreso Todos los paquetes que salen del host por cualquier motivo atraviesan las tablas `mangle` y `nat` de la cadena `POSTROUTING`.

[5] https://es.wikipedia.org/wiki/Netfilter/iptables

`ip link list`	Consultar interfaces
`ip address show`	Consultar direcciones de interfaces
`ip neigh show`	Tabla ARP
`ip route show`	Rutas
`ip route list table main`	Rutas, tabla principal
`ip route list table T`	Rutas, tabla T
`ip route flush table main`	Borrar rutas de tabla principal
`ip route flush cache`	Borrar cache de rutas
`ip rule add from 10.0.0.0/24 table T`	Agregar regla de asignación de tablas
`ip rule list`	Consultar reglas de asignación

Cuadro 3: Ejemplos de uso del comando ip

Paquetes destinados al host Luego de determinarse que el paquete está destinado al host, éste atraviesa la cadena de `INPUT`, con las reglas que haya en la tabla `mangle` y la tabla `filter` de esa cadena, en ese orden (Fig. 15).

Paquetes originados en el host Son producidos por procesos locales que emiten pedidos de servicio o responden a servicios solicitados. Atraviesan la cadena de `OUTPUT` (`mangle`, `nat` y `filter`, en ese orden) y luego se dirigen a la cadena de `POSTROUTING` (Fig. 16).

Paquetes que atraviesan el host Se trata de paquetes que el router debe reenviar. Atraviesan las tablas `mangle` y `filter` de la cadena `FORWARD` y luego se dirigen a la cadena `POSTROUTING` (Fig. 17).

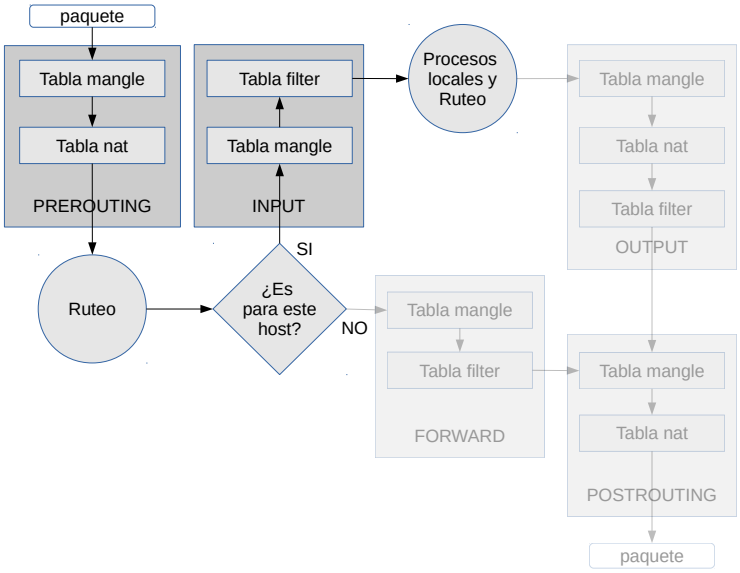

Figura 15: Recorrido de un paquete destinado al host

Ruteo por políticas

Como hemos dicho, un host determina por cuál de sus interfaces debe emitir un paquete mediante el proceso de ruteo, usando la información de rutas contenida en una tabla de ruteo o reenvío. Habitualmente, la tabla utilizada para el ruteo regular es única (la tabla `main`), pero con la técnica de ruteo por políticas, se pueden utilizar diferentes tablas de ruteo.

Usando el comando `ip` es posible asociar una tabla de ruteo distinta con cada acceso en el escenario de Split Access. La decisión de qué tráfico utilizará cada tabla de ruteo se puede configurar en varias

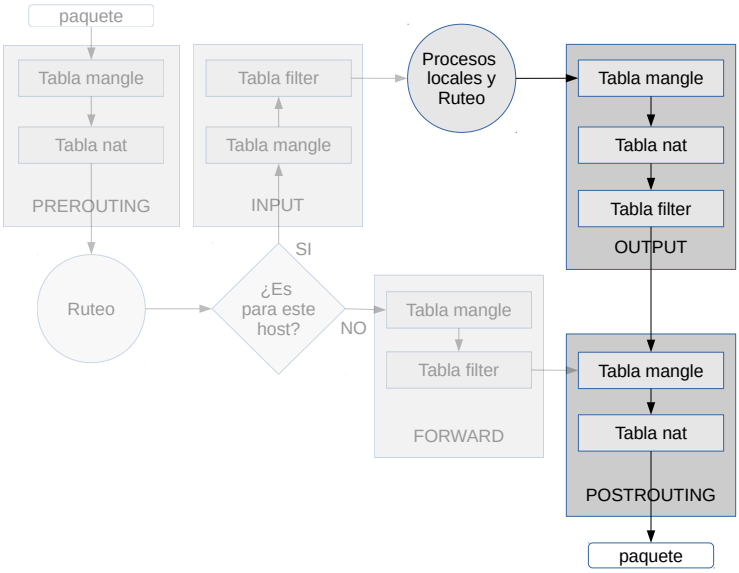

Figura 16: Recorrido de un paquete originado en el host

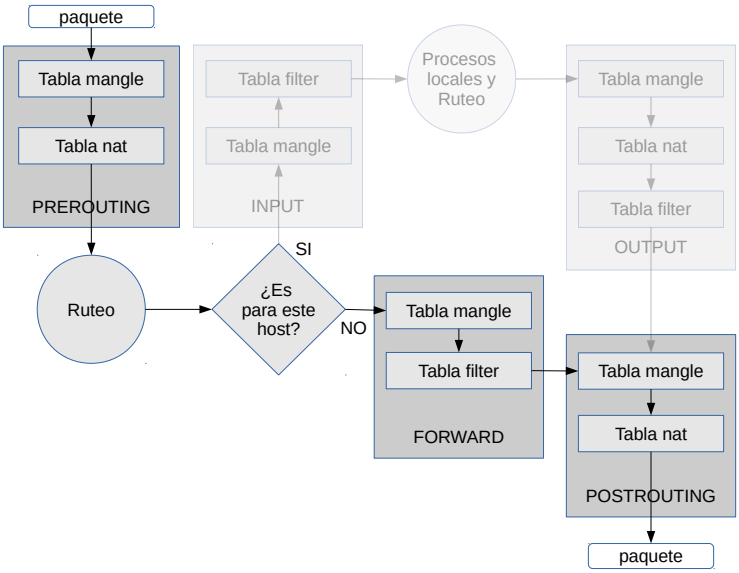

Figura 17: Recorrido de un paquete reenviado por el host

formas. Para definir qué paquetes usarán cada tabla de ruteo del sistema, se aplican reglas de selección de tablas. Estas reglas no aparecen en las tablas de ruteo sino que figuran en una tabla de reglas especial mantenida por el sistema iproute2.

Ante cada paquete nuevo que llega, el proceso de ruteo buscará primero reglas para seleccionar una tabla de ruteo, y sólo si no las encuentra, aplicará las rutas en la tabla `main`. Cada tabla de ruteo puede tener una ruta default propia, lo que define por cuál de los accesos irá el tráfico saliente. En particular, una ruta default en una tabla específica tendrá prioridad sobre la ruta default habitual.

Una vez seleccionada una tabla de ruteo, el kernel utilizará las rutas en dicha tabla para determinar qué interfaz de salida le corresponde a un paquete. Para cada tabla de ruteo se aplica siempre el mecanismo clásico de examinar la dirección destino del paquete y localizar en esa tabla el prefijo coincidente más largo.

Creación de tablas de ruteo

El paquete `iproute2` identifica las tablas con números, pero es más cómodo para el administrador establecer unos nombres simbólicos en el archivo `/etc/iproute2/rt_tables`. Simplemente se eligen números y nombres arbitrarios, que no colisionen con los ya reservados. Los nombres servirán posteriormente para referirse a las diferentes tablas de ruteo usando el comando `ip`, y pueden ser los nombres de los ISP o proveedores de acceso. Una vez creados los nombres simbólicos, por cada acceso a Internet disponible es necesario identificar los siguientes datos.

- Nombre de la tabla de ruteo (P).
- Dirección del router del proveedor (R).
- Dirección de red donde se ubica el router (N).
- Interfaz del router de la organización que comunica con el router del proveedor (IF).
- Dirección de dicha interfaz (IP).

Con estos datos, por cada acceso disponible:

1. Se insertan las reglas de ruteo en su tabla correspondiente.

```
ip route add N dev IF src IP table P
ip route add default via R table P
```

2. Se inserta en la tabla `main` la regla de ruteo al gateway del acceso.

```
ip route add N dev IF src IP
```

Definición de políticas

A continuación se establecen las reglas por las cuales cada host, cuyo tráfico se va a reenviar, utilizará una u otra tabla de ruteo. Consideraremos tres diferentes formas de diseñar y establecer las políticas.

1. Política según el origen

La idea es simplemente separar los hosts de las redes locales en grupos, estáticamente, por cualquier criterio conveniente; y derivar el tráfico de cada grupo por rutas diferentes y fijas.

Por cada host o red con dirección IP que se quiere conducir por el acceso P:

```
ip rule add from IP table P
```

> Las reglas se almacenan en una lista en memoria y se aplican en el orden en que están almacenadas. La primera regla cuya cláusula `from` coincide con el origen del tráfico determina la tabla que se aplicará.
> Sin embargo, cada regla introducida se inserta al principio de la lista, de forma que las últimas reglas introducidas tienen prioridad sobre las anteriores.

En este ejemplo, se desea que la red 10.0.0.0/24 en general use la tabla T1; **pero** la mitad superior de esos hosts (los que tienen último octeto mayor o igual que 128) deben usar la T2; **excepto**, de éstos, el host 129, que usará la tabla T1.

- Al insertar las reglas:

```
# ip rule add from 10.0.0.0/24 table T1
# ip rule add from 10.0.0.128/25 table T2
# ip rule add from 10.0.0.129/32 table T1
```

- En memoria:

```
# ip rule show
0:      from all lookup local
32763: from 10.0.0.129 lookup T1
32764: from 10.0.0.128/25 lookup T2
32765: from 10.0.0.0/24 lookup T1
32766: from all lookup main
32767: from all lookup default
```

Las reglas se insertarán de lo general a lo particular, para que, al aplicarlas, las que se encuentren primero en la lista en memoria sean las más específicas.

2. Política según la clase de tráfico

Utilizando `iptables` para manipular la tabla `mangle` de Netfilter, podemos crear **marcas** numéricas, que acompañan a los paquetes en su tránsito por el kernel. Estas marcas son reconocidas por el proceso de ruteo, y sirven, entre otras cosas, para determinar la tabla de ruteo a utilizar para cada paquete. Las marcas existen únicamente durante el viaje de los paquetes dentro del sistema, y al ser emitido el paquete por su interfaz de salida desaparecen.

Marcando los paquetes podemos definir cualquier política basada en las características que puede distinguir iptables en ellos (direcciones IP origen o destino, protocolo, etc.). Cada conjunto de valores de estas características definirá una clase de tráfico.

Por ejemplo, para el tráfico TCP al port X, con IP origen S y destino D (o cualesquiera otras condiciones que sea posible seleccionar mediante iptables), marcamos los paquetes en el ingreso con la marca numérica Z.

```
iptables -t mangle -A PREROUTING -p tcp --dport X -s S -d D -j MARK --set-mark Z
```

Luego se establecen las reglas que derivan el tráfico con cada marca Z a la tabla de ruteo P que se desee:

```
ip rule add fwmark Z table P
```

3. Política de balance de carga

El paquete iproute2 permite establecer una ruta default con capacidad de *multipath*. Con la configuración adecuada, esta ruta por defecto tomará alternativamente cada uno de los enlaces disponibles, asociando pesos a los enlaces.

Para cada paquete que se emita, el proceso de ruteo elegirá una ruta entre las disponibles en modo *round-robin*, afectando la elección con determinados pesos. Es decir, si un enlace A tiene peso doble (o triple, etc.) que otro enlace B, entonces A será elegido el doble (o triple, etc.) de veces que B.

Sin embargo, una vez que se elija automáticamente una ruta a un destino, ésta quedará en la cache de rutas. Esto es necesario para que la conexión no cambie la dirección IP de origen, pero al mismo tiempo influye en la distribución de tráfico, porque las conexiones siguientes al mismo destino utilizarán la misma ruta hasta que ésta caiga de la cache de rutas o hasta que sea vaciada administrativamente (con `ip route flush cache`). La consecuencia es que no se logrará equilibrar exactamente el uso de los enlaces en la proporción especificada, sino en forma aproximada.

En este ejemplo, $R1$ y $R2$ son las direcciones de los gateways de los enlaces, $IF1$ e $IF2$ las interfaces que conectan al router con ellos, y $W1$ y $W2$ son números que designan los pesos relativos de la carga que se desea hacer circular por cada enlace. Si los enlaces contratados tienen aproximadamente la misma velocidad de transmisión, pesos iguales deben utilizar los enlaces aproximadamente en la misma proporción.

```
ip route add default scope global nexthop via R1 dev IF1 weight W1 nexthop via
    R2 dev IF2 weight W2
```

Temas de práctica

Laboratorio virtual de Split Access

La organización tiene dos redes locales o VLANs, LAN1 y LAN2, conectadas por un router. Se ha contratado acceso a Internet de dos proveedores diferentes. Ambos han provisto sus routers r1 y r2 con una dirección interna privada. El problema consiste en poder ofrecer a las redes locales de la organización acceso por uno u otro de los servicios, en la forma más flexible posible.

En el laboratorio se han fijado las direcciones que aparecen en el diagrama de Fig. 18. Todas las demás deben ser configuradas con direcciones a elección.

1. Establecer ruteo estático según la red de origen.
   - Comprobar que r1 y r2 tienen acceso a Internet.
   - Dar direcciones a las redes locales y establecer ruta por defecto para los hosts.
   - En `router` reemplazar el ruteo simple por *masquerading*.
   - Comprobar que los hosts pc1 a pc4 tienen ruta hacia las interfaces internas de r1 y r2.
   - Especificar en `router` un gateway default en la tabla main.
   - Comprobar que los hosts pc1 a pc4 tienen acceso a sitios de Internet.
   - Por cada proveedor, establecer en `router` una tabla de ruteo en `/etc/iproute2/rt_tables`. Configurar cada tabla de ruteo con regla default y especificación de IP de salida.
   - Agregar reglas de ruteo por origen estáticas por cada red local.
   - Desde los hosts, hacer ping a un host público de Internet. Ver con tcpdump en las interfaces internas de r1 y de r2 que `router` aplica la política por origen correspondiente.
2. Establecer políticas por clase de tráfico.
   - Modificar la configuración de las reglas de selección de tablas, aplicando con `iptables` una política que derive el tráfico de HTTP por el acceso 1 y todo otro tráfico por el acceso 2.
   - Establecer que el tráfico ICMP debe usar un enlace, y el tráfico TCP el otro.
3. Incorporar excepciones.
   - Establecer la excepción de que un determinado host de la organización utilizará un acceso diferente que el resto de la red a la que pertenece.

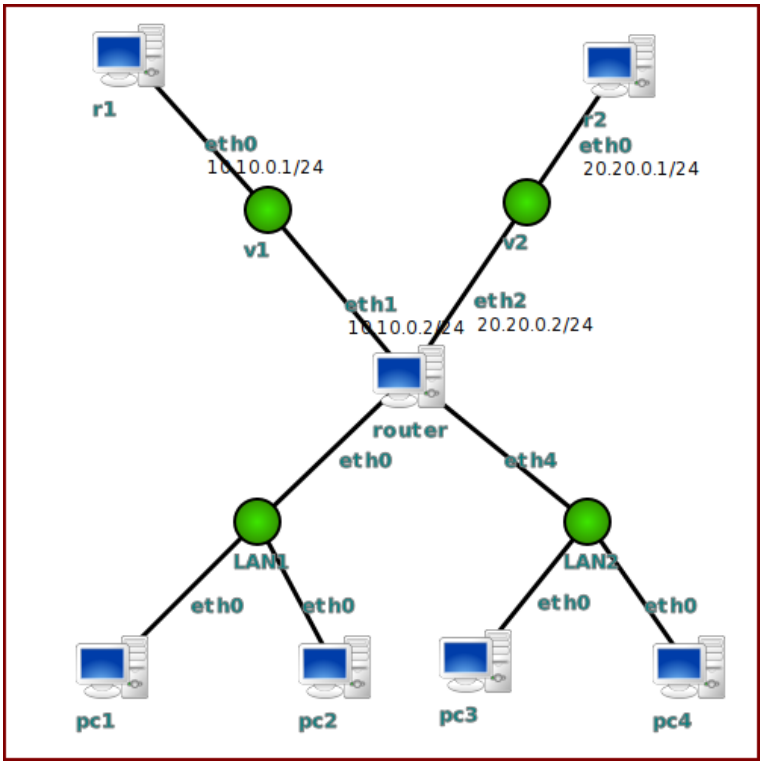

Figura 18: Esquema del laboratorio virtual de Split Access

- Establecer la excepción de que un determinado servidor HTTP, externo a la organización, se accederá siempre por el acceso 2. ¿Cómo se diferencia este caso del anterior?

4. Eliminar la regla de ruteo default, y agregar la directiva ip con pesos para balancear carga.
   - Experimentar con transferencias de archivos, ping, etc.
   - Experimentar con pesos diferentes. Ensayar transferencias desde diferentes mirrors de descargas (para que las conexiones sean hacia direcciones diferentes) y observar los bytes traficados por cada interfaz del router según muestra `ifconfig`.
   - ¿Es posible combinar políticas por clase de tráfico con política de balance de carga? Es decir, ¿se puede establecer una política general de balance de carga, pero además asociar estáticamente una clase particular de tráfico con un enlace fijo?
5. En `router`, implementar scripts o funciones que conviertan el acceso a Internet en tolerante a fallos de ISP. Paso 1: pensar el problema. Paso 2 (opcional): ver Sugerencias en las Notas del Laboratorio.

Notas del Laboratorio

- Para dar acceso a Internet a las máquinas virtuales del laboratorio que funcionan como routers, es importante dar en la consola **del host** los comandos que enmascaran el tráfico de salida proveniente de esas máquinas virtuales.

```
iptables -t nat -A POSTROUTING -s 172.16.100.0/24 -o eth0 -j MASQUERADE
iptables -A FORWARD -i eth0 -m state --state RELATED,ESTABLISHED -j ACCEPT
```

- Para efectuar una transferencia de un archivo sin descargarlo en el disco se puede utilizar:

```
wget -O - http://servidor/archivo > /dev/null
```

- Sugerencias de análisis para el problema de Tolerancia a Fallos
  - `testping`: Monitorea periódicamente el alcance mediante ping a una dirección IP testigo. Devuelve alcanzable o no alcanzable, luego de una secuencia de n pings con m exitosos o fallidos. Cantidades n y m configurables. Argumentos: dirección IP testigo, n, m.
  - `monitorear`: Ciclo infinito con intervalo configurable que ejecuta a `testping`. Recibe como argumentos los nombres de dos scripts o funciones, el intervalo, la dirección IP testigo, n, m. Reconoce los eventos resultantes de `testping` y actúa en consecuencia:
    - Evento de dirección testigo inalcanzable $\rightarrow$ ejecuta el primer script.
    - Evento de dirección testigo que vuelve a ser alcanzable $\rightarrow$ ejecuta el segundo script.
  - `toleranciaAFallos`: Lanzado en dos instancias, una para ISP A, y otra para ISP B. Ejecuta a `monitorear` dándole como argumentos los nombres de dos funciones o scripts que contienen las reglas de ruteo para implementar el acceso por uno u otro proveedor.

Parte V
Anexos

A. VLANs en Linux

Creación de VLAN

```
# ip link add link eth0 name eth0.VLAN2 type vlan id 2
```

La interfaz virtual eth0.VLAN2 se comportará como una interfaz Ethernet habitual. Todo el tráfico que sea dirigido a esta interfaz será enviado por la interfaz maestra pero llevará el tag con VLAN ID número 2. Este tráfico sólo puede ser aceptado por dispositivos con conocimiento del formato de trunking.

Consultar el VLAN ID de una interfaz

```
# ip -d link show eth0.VLAN2
```

Dar una dirección IP

```
# ip addr add 192.168.100.1/24 brd 192.168.100.255 dev eth0.VLAN2
# ip link set dev eth0.VLAN2 up
```

Desactivar administrativamente una interfaz

```
# ip link set dev eth0.VLAN2 down
```

Eliminar una interfaz

```
# ip link delete eth0.VLAN2
```

B. Bridging en Linux

Interfaz para creación e inspección de bridges

```
# brctl
Usage: brctl [commands]
commands:
       addbr          <bridge>              add bridge
       delbr          <bridge>              delete bridge
       addif          <bridge> <device>     add interface to bridge
       delif          <bridge> <device>     delete interface from bridge
       hairpin        <bridge> <port> {on|off}     turn hairpin on/off
       setageing      <bridge> <time>       set ageing time
       setbridgeprio  <bridge> <prio>       set bridge priority
```

```
       setfd           <bridge> <time>       set bridge forward delay
       sethello        <bridge> <time>       set hello time
       setmaxage       <bridge> <time>       set max message age
       setpathcost     <bridge> <port> <cost> set path cost
       setportprio     <bridge> <port> <prio> set port priority
       show            [ <bridge> ]          show a list of bridges
       showmacs        <bridge>              show a list of mac addrs
       showstp         <bridge>              show bridge stp info
       stp             <bridge> {on|off}     turn stp on/off
```

Configuración estática de bridges

Configuración en RedHat, CentOS, derivados

Archivo /etc/sysconfig/network-scripts/ifcfg-br0

```
DEVICE=br0
BOOTPROTO=none
ONBOOT=yes
TYPE=Bridge
IPADDR=192.168.0.1
NETMASK=255.255.255.0
NETWORK=192.168.0.0
BROADCAST=192.168.0.255
```

Archivo /etc/sysconfig/network-scripts/ifcfg-eth0

```
TYPE=Ethernet
DEVICE=eth0
ONBOOT=yes
USERCTL=no
BRIDGE=br0
```

Configuración en Debian, Ubuntu, derivados

Archivo /etc/network/interfaces

```
auto br0
iface br0 inet static
       address 10.10.0.15
       netmask 255.255.255.0
       gateway 10.10.0.1
       bridge_ports eth0 eth1
       up /usr/sbin/brctl stp br0 on
```

C. Ejemplos de configuración OpenVPN

Servidor

```
################################################
# Sample OpenVPN 2.0 config file for           #
# multi-client server.                         #
```

```
#                                             #
# This file is for the server side            #
# of a many-clients <-> one-server            #
# OpenVPN configuration.                      #
#                                             #
# OpenVPN also supports                       #
# single-machine <-> single-machine           #
# configurations (See the Examples page       #
# on the web site for more info).             #
#                                             #
# This config should work on Windows          #
# or Linux/BSD systems. Remember on           #
# Windows to quote pathnames and use          #
# double backslashes, e.g.:                   #
# "C:\\Program Files\\OpenVPN\\config\\foo.key" #
#                                             #
# Comments are preceded with '#' or ';'       #
###############################################

# Which local IP address should OpenVPN
# listen on? (optional)
;local a.b.c.d

# Which TCP/UDP port should OpenVPN listen on?
# If you want to run multiple OpenVPN instances
# on the same machine, use a different port
# number for each one. You will need to
# open up this port on your firewall.
port 1194

# TCP or UDP server?
;proto tcp
proto udp

# "dev tun" will create a routed IP tunnel,
# "dev tap" will create an ethernet tunnel.
# Use "dev tap0" if you are ethernet bridging
# and have precreated a tap0 virtual interface
# and bridged it with your ethernet interface.
# If you want to control access policies
# over the VPN, you must create firewall
# rules for the the TUN/TAP interface.
# On non-Windows systems, you can give
# an explicit unit number, such as tun0.
# On Windows, use "dev-node" for this.
# On most systems, the VPN will not function
# unless you partially or fully disable
# the firewall for the TUN/TAP interface.
;dev tap
dev tun

# Windows needs the TAP-Win32 adapter name
# from the Network Connections panel if you
# have more than one. On XP SP2 or higher,
# you may need to selectively disable the
```

```
# Windows firewall for the TAP adapter.
# Non-Windows systems usually don't need this.
;dev-node MyTap

# SSL/TLS root certificate (ca), certificate
# (cert), and private key (key). Each client
# and the server must have their own cert and
# key file. The server and all clients will
# use the same ca file.
#
# See the "easy-rsa" directory for a series
# of scripts for generating RSA certificates
# and private keys. Remember to use
# a unique Common Name for the server
# and each of the client certificates.
#
# Any X509 key management system can be used.
# OpenVPN can also use a PKCS #12 formatted key file
# (see "pkcs12" directive in man page).
ca ca.crt
cert server.crt
key server.key # This file should be kept secret

# Diffie hellman parameters.
# Generate your own with:
#   openssl dhparam -out dh1024.pem 1024
# Substitute 2048 for 1024 if you are using
# 2048 bit keys.
dh dh1024.pem

# Configure server mode and supply a VPN subnet
# for OpenVPN to draw client addresses from.
# The server will take 10.8.0.1 for itself,
# the rest will be made available to clients.
# Each client will be able to reach the server
# on 10.8.0.1. Comment this line out if you are
# ethernet bridging. See the man page for more info.
server 10.8.0.0 255.255.255.0

# Maintain a record of client <-> virtual IP address
# associations in this file. If OpenVPN goes down or
# is restarted, reconnecting clients can be assigned
# the same virtual IP address from the pool that was
# previously assigned.
ifconfig-pool-persist ipp.txt

# Configure server mode for ethernet bridging.
# You must first use your OS's bridging capability
# to bridge the TAP interface with the ethernet
# NIC interface. Then you must manually set the
# IP/netmask on the bridge interface, here we
# assume 10.8.0.4/255.255.255.0. Finally we
# must set aside an IP range in this subnet
# (start=10.8.0.50 end=10.8.0.100) to allocate
# to connecting clients. Leave this line commented
```

```
# out unless you are ethernet bridging.
;server-bridge 10.8.0.4 255.255.255.0 10.8.0.50 10.8.0.100

# Configure server mode for ethernet bridging
# using a DHCP-proxy, where clients talk
# to the OpenVPN server-side DHCP server
# to receive their IP address allocation
# and DNS server addresses. You must first use
# your OS's bridging capability to bridge the TAP
# interface with the ethernet NIC interface.
# Note: this mode only works on clients (such as
# Windows), where the client-side TAP adapter is
# bound to a DHCP client.
;server-bridge

# Push routes to the client to allow it
# to reach other private subnets behind
# the server. Remember that these
# private subnets will also need
# to know to route the OpenVPN client
# address pool (10.8.0.0/255.255.255.0)
# back to the OpenVPN server.
;push "route 192.168.10.0 255.255.255.0"
;push "route 192.168.20.0 255.255.255.0"

# To assign specific IP addresses to specific
# clients or if a connecting client has a private
# subnet behind it that should also have VPN access,
# use the subdirectory "ccd" for client-specific
# configuration files (see man page for more info).

# EXAMPLE: Suppose the client
# having the certificate common name "Thelonious"
# also has a small subnet behind his connecting
# machine, such as 192.168.40.128/255.255.255.248.
# First, uncomment out these lines:
;client-config-dir ccd
;route 192.168.40.128 255.255.255.248
# Then create a file ccd/Thelonious with this line:
#   iroute 192.168.40.128 255.255.255.248
# This will allow Thelonious' private subnet to
# access the VPN. This example will only work
# if you are routing, not bridging, i.e. you are
# using "dev tun" and "server" directives.

# EXAMPLE: Suppose you want to give
# Thelonious a fixed VPN IP address of 10.9.0.1.
# First uncomment out these lines:
;client-config-dir ccd
;route 10.9.0.0 255.255.255.252
# Then add this line to ccd/Thelonious:
#   ifconfig-push 10.9.0.1 10.9.0.2

# Suppose that you want to enable different
# firewall access policies for different groups
```

```
# of clients. There are two methods:
# (1) Run multiple OpenVPN daemons, one for each
#     group, and firewall the TUN/TAP interface
#     for each group/daemon appropriately.
# (2) (Advanced) Create a script to dynamically
#     modify the firewall in response to access
#     from different clients. See man
#     page for more info on learn-address script.
;learn-address ./script

# If enabled, this directive will configure
# all clients to redirect their default
# network gateway through the VPN, causing
# all IP traffic such as web browsing and
# and DNS lookups to go through the VPN
# (The OpenVPN server machine may need to NAT
# or bridge the TUN/TAP interface to the internet
# in order for this to work properly).
;push "redirect-gateway def1 bypass-dhcp"

# Certain Windows-specific network settings
# can be pushed to clients, such as DNS
# or WINS server addresses. CAVEAT:
# http://openvpn.net/faq.html#dhcpcaveats
# The addresses below refer to the public
# DNS servers provided by opendns.com.
;push "dhcp-option DNS 208.67.222.222"
;push "dhcp-option DNS 208.67.220.220"

# Uncomment this directive to allow different
# clients to be able to "see" each other.
# By default, clients will only see the server.
# To force clients to only see the server, you
# will also need to appropriately firewall the
# server's TUN/TAP interface.
;client-to-client

# Uncomment this directive if multiple clients
# might connect with the same certificate/key
# files or common names. This is recommended
# only for testing purposes. For production use,
# each client should have its own certificate/key
# pair.
#
# IF YOU HAVE NOT GENERATED INDIVIDUAL
# CERTIFICATE/KEY PAIRS FOR EACH CLIENT,
# EACH HAVING ITS OWN UNIQUE "COMMON NAME",
# UNCOMMENT THIS LINE OUT.
;duplicate-cn

# The keepalive directive causes ping-like
# messages to be sent back and forth over
# the link so that each side knows when
# the other side has gone down.
# Ping every 10 seconds, assume that remote
```

```
# peer is down if no ping received during
# a 120 second time period.
keepalive 10 120

# For extra security beyond that provided
# by SSL/TLS, create an "HMAC firewall"
# to help block DoS attacks and UDP port flooding.
#
# Generate with:
#   openvpn --genkey --secret ta.key
#
# The server and each client must have
# a copy of this key.
# The second parameter should be '0'
# on the server and '1' on the clients.
;tls-auth ta.key 0 # This file is secret

# Select a cryptographic cipher.
# This config item must be copied to
# the client config file as well.
;cipher BF-CBC      # Blowfish (default)
;cipher AES-128-CBC # AES
;cipher DES-EDE3-CBC # Triple-DES

# Enable compression on the VPN link.
# If you enable it here, you must also
# enable it in the client config file.
comp-lzo

# The maximum number of concurrently connected
# clients we want to allow.
;max-clients 100

# It's a good idea to reduce the OpenVPN
# daemon's privileges after initialization.
#
# You can uncomment this out on
# non-Windows systems.
;user nobody
;group nogroup

# The persist options will try to avoid
# accessing certain resources on restart
# that may no longer be accessible because
# of the privilege downgrade.
persist-key
persist-tun

# Output a short status file showing
# current connections, truncated
# and rewritten every minute.
status openvpn-status.log

# By default, log messages will go to the syslog (or
# on Windows, if running as a service, they will go to
```

```
# the "\Program Files\OpenVPN\log" directory).
# Use log or log-append to override this default.
# "log" will truncate the log file on OpenVPN startup,
# while "log-append" will append to it. Use one
# or the other (but not both).
;log        openvpn.log
;log-append openvpn.log

# Set the appropriate level of log
# file verbosity.
#
# 0 is silent, except for fatal errors
# 4 is reasonable for general usage
# 5 and 6 can help to debug connection problems
# 9 is extremely verbose
verb 3

# Silence repeating messages. At most 20
# sequential messages of the same message
# category will be output to the log.
;mute 20
```

Cliente

```
################################################
# Sample client-side OpenVPN 2.0 config file #
# for connecting to multi-client server. #
#                                        #
# This configuration can be used by multiple #
# clients, however each client should have #
# its own cert and key files.            #
#                                        #
# On Windows, you might want to rename this #
# file so it has a .ovpn extension       #
################################################

# Specify that we are a client and that we
# will be pulling certain config file directives
# from the server.
client

# Use the same setting as you are using on
# the server.
# On most systems, the VPN will not function
# unless you partially or fully disable
# the firewall for the TUN/TAP interface.
;dev tap
dev tun

# Windows needs the TAP-Win32 adapter name
# from the Network Connections panel
# if you have more than one. On XP SP2,
# you may need to disable the firewall
```

```
# for the TAP adapter.
;dev-node MyTap

# Are we connecting to a TCP or
# UDP server? Use the same setting as
# on the server.
;proto tcp
proto udp

# The hostname/IP and port of the server.
# You can have multiple remote entries
# to load balance between the servers.
remote my-server-1 1194
;remote my-server-2 1194

# Choose a random host from the remote
# list for load-balancing. Otherwise
# try hosts in the order specified.
;remote-random

# Keep trying indefinitely to resolve the
# host name of the OpenVPN server. Very useful
# on machines which are not permanently connected
# to the internet such as laptops.
resolv-retry infinite

# Most clients don't need to bind to
# a specific local port number.
nobind

# Downgrade privileges after initialization (non-Windows only)
;user nobody
;group nogroup

# Try to preserve some state across restarts.
persist-key
persist-tun

# If you are connecting through an
# HTTP proxy to reach the actual OpenVPN
# server, put the proxy server/IP and
# port number here. See the man page
# if your proxy server requires
# authentication.
;http-proxy-retry # retry on connection failures
;http-proxy [proxy server] [proxy port #]

# Wireless networks often produce a lot
# of duplicate packets. Set this flag
# to silence duplicate packet warnings.
;mute-replay-warnings

# SSL/TLS parms.
# See the server config file for more
# description. It's best to use
```

```
# a separate .crt/.key file pair
# for each client. A single ca
# file can be used for all clients.
ca ca.crt
cert client.crt
key client.key

# Verify server certificate by checking
# that the certicate has the nsCertType
# field set to "server". This is an
# important precaution to protect against
# a potential attack discussed here:
#  http://openvpn.net/howto.html#mitm
#
# To use this feature, you will need to generate
# your server certificates with the nsCertType
# field set to "server". The build-key-server
# script in the easy-rsa folder will do this.
ns-cert-type server

# If a tls-auth key is used on the server
# then every client must also have the key.
;tls-auth ta.key 1

# Select a cryptographic cipher.
# If the cipher option is used on the server
# then you must also specify it here.
;cipher x

# Enable compression on the VPN link.
# Don't enable this unless it is also
# enabled in the server config file.
comp-lzo

# Set log file verbosity.
verb 3

# Silence repeating messages
;mute 20
```

D. Scripts para OpenVPN en modo bridge

Archivo sample-scripts/bridge-start

```
#!/bin/bash

###################################
# Set up Ethernet bridge on Linux
# Requires: bridge-utils
###################################

# Define Bridge Interface
br="br0"
```

```
# Define list of TAP interfaces to be bridged,
# for example tap="tap0 tap1 tap2".
tap="tap0"

# Define physical ethernet interface to be bridged
# with TAP interface(s) above.
eth="eth0"
eth_ip="192.168.8.4"
eth_netmask="255.255.255.0"
eth_broadcast="192.168.8.255"

for t in $tap; do
    openvpn --mktun --dev $t
done

brctl addbr $br
brctl addif $br $eth

for t in $tap; do
    brctl addif $br $t
done

for t in $tap; do
    ifconfig $t 0.0.0.0 promisc up
done

ifconfig $eth 0.0.0.0 promisc up

ifconfig $br $eth_ip netmask $eth_netmask broadcast $eth_broadcast
```

Archivo sample-scripts/bridge-stop

```
#!/bin/bash

########################################
# Tear Down Ethernet bridge on Linux
########################################

# Define Bridge Interface
br="br0"

# Define list of TAP interfaces to be bridged together
tap="tap0"

ifconfig $br down
brctl delbr $br

for t in $tap; do
    openvpn --rmtun --dev $t
done
```