

Configuración de nodos wireless (3)

Participando en una red inalámbrica ciudadana(W.O.M.A.N).

Bienvenidos de nuevo a la serie de artículos de cómo pertenecer de una forma activa a una de las tantísimas comunidades wireless que hay en todos lados ; gracias al esfuerzo de muchos para hacer que éste, nuestro sueño ,se haga realidad.

La mejor forma de lograr éste objetivo es el hacer ruido dentro de nuestro barrio , pueblo o lo que sea. Así que continuamos por donde lo dejamos el mes anterior .

0.Resumiendo lo irresumible .

Cómo es costumbre vamos a hacer un resumen de lo visto en el número anterior:

1) Analizamos el software necesario en todo nodo wireless que se precie , el software más importante es el siguiente:

- 1.1)Wireless-tools
- 1.2)Wavemon
- 1.3)Plugins para el gkrellm (si lo usas)
- 1.4)Herramientas que integra el propio paquete del hostap

2) Aprendimos los distintos modos de las tarjetas wireless, es decir , vimos las formas en las que pueden funcionar dependiendo de nuestras necesidades , los modos que vimos fueron .

- 2.1)Managed
- 2.2)Ad-hoc
- 2.3)Master(Modo Access Point)

3) Por último lo que vimos fue el manejo de las wireless-tools , y empezamos a crear servicios para nuestro nodo , como fue el servicio de DHCP(Asignación dinámica de IP) para que se automatice de una forma coherente y normal .

Pues como dijimos ,en éste número vamos a ver como damos servicios importantes cómo es el servicio de internet a los clientes del nodo (vamos a ver distintas formas) , veremos como funciona la estructura del backbone de nuestra comunidad y la interconexión entre nodos(VPN) .



Foto:Mi viejo portatil conectado por wireless en un nodo en la Sierra de Madrid (Gallinerowireless)

1.Dando a nuestros clientes servicios importantes.

Bueno pues una vez más vamos a empezar por el principio por que sino no se avanza demasiado .

Una de las cosas que nos preguntamos (cosa obvia) fué la de dotar un servicio importante y consistente como es el de poder repartir (compartir) nuestra banda de internet por medio de nuestros nodos; Ya que vimos que un gran porcentaje de los anchos de banda de cada nodo estaba muy desaprovechado , por lo que analizamos lo que podiamos hacer .

Una vez más, vimos que nuestro kernelcito de Linux traía la posibilidad de hacer bridging o hacer NAT (Network address Translation) por lo que nos pusimos a ver las posibilidades que se nos presentaban con cada uno de los modos.

Éstos dos modos de topología lógica de red se diferencian en muchos aspectos, vamos a ir viendo que ventajas e inconvenientes tienen éstos modos de red.

A)Montando nuestro nodo por medio de bridging .

Cómo hemos dicho éste modo se nos presenta en el kernel de Linux , lo que vamos a conseguir con éste modo es que nuestro nodo se integre dentro de nuestra red de cable cómo un equipo más . Ésto lo que conlleva son problemas de seguridad , ya que si cualquier personajillo que ande cerca de nuestro nodo se puede conectar a nuestra red cómo si estuviera en su propia casa .

La forma de configurar nuestro nodo cómo bridge sería de la siguiente forma:

Nos imaginamos que nuestra red interna tiene un rango IP (clase C) tipo 192.168.0.x, el router tiene la 192.168.0.1 que es el que nos va a proporcionar la salida a internet, lo que el gateway de nuestros clientes sería la ip del router y no la IP del nodo(Acces Point) .

Lo que hay que activar en nuestro kernel es la opción Ethernet bridging ésta opción aparece en los nucleos de la version 2.4.x y lo nucleos 2.2.x cómo parche.

Vamos a ver un ejemplo de cómo sería la forma de configurarlo

Tenemos que descargar las bridge-utils que son las herramientas que nos va a permitir manejar los interfaces br*(dispositivos de bridge) que han sido activados en el kernel.

Ejemplo de construcción de una red por medio de bridging

```
brctl addbr br0 #Lo primero es crear un dispositivo de bridge en este caso br0
brctl addif br0 eth0 #Le decimos que interface es el que está conectado a la red cableada
brctl addif br0 wlan0 #Le decimos que interface es el de la red wireless.
ifconfig eth0 0.0.0.0 #ponemos los interfaces con la IP genérica
ifconfig wlan0 0.0.0.0
ifconfig br0 192.168.0.34 up #Le damos una ip al interfaz de bridge y levantamos el interface
```

¿Que es exactamente lo que hemos conseguido ?

Pues lo que hemos conseguido es ; Hacer que nuestro AP(nodo) logre mover los paquetes que le pasan por la tarjeta ethernet a la red wireless por lo que la máquina va a poder ver cualquiera de los dispositivos de nuestra red y cualquiera de las máquinas asociadas a nuestro AP pueden comunicarse con las máquinas de la red .

La otra solución que vimos fue hacer un gateway wireless con nuestro nodo por medio de NAT(network address translation) , es una forma más segura ya que la red tendrá un rango y los clientes que accedan por la red wireless tendrán otro rango , para orientarnos en ésto voy a exponer como tenemos la estructura de red en madridwireless (www.madridwireless.net) .

B) Creando una red por medio de NAT a través de la topología de madridwireless.

En madridwireless y en otras comunidades cómo www.zaragozawireless.org empezamos a pensar en montar un estandar para nuestra red , el estandar iba a estar formado de dos rangos de ip un rango dentro de la clase A 10.64.x.y y otro rango de clase C del tipo 172.16.x.y cuya misión iba a ser la de proporcionar una cierta estructura de los nodos y de los clientes de nuestras comunidades. Todo ésto está de acuerdo con el estandar de la comunidad de freenetworks (www.freenetworks.org) a la cual nuestras comunidades estamos completamente unidas debido a su experiencia en redes de ambito libre .

Los rangos corresponden a:

El rango de clase A (10.64.x.y) lo pusimos para los clientes, es decir para toda máquina que requiera de uno de los servicios proporcionados por nuestro “nodo-güireles” , por lo que nuestro interfaz de red wireless tiene que tener una ip de ese rango en resumen toda máquina que tenga la intención de conectarse se le asignaría una ip tipo 10.64.10.2 (éste es mi rango para “clientes”).

El rango de clase C (172.16.x.y) lo pusimos para el denominado backbone, es decir, el esqueleto que va a interconectar todos los nodos que van a formar nuestra red de la comunidad, en conclusión, lo que vamos a hacer es unir los dispositivos dedicados para repartir servicios a todos los clientes y nodos .

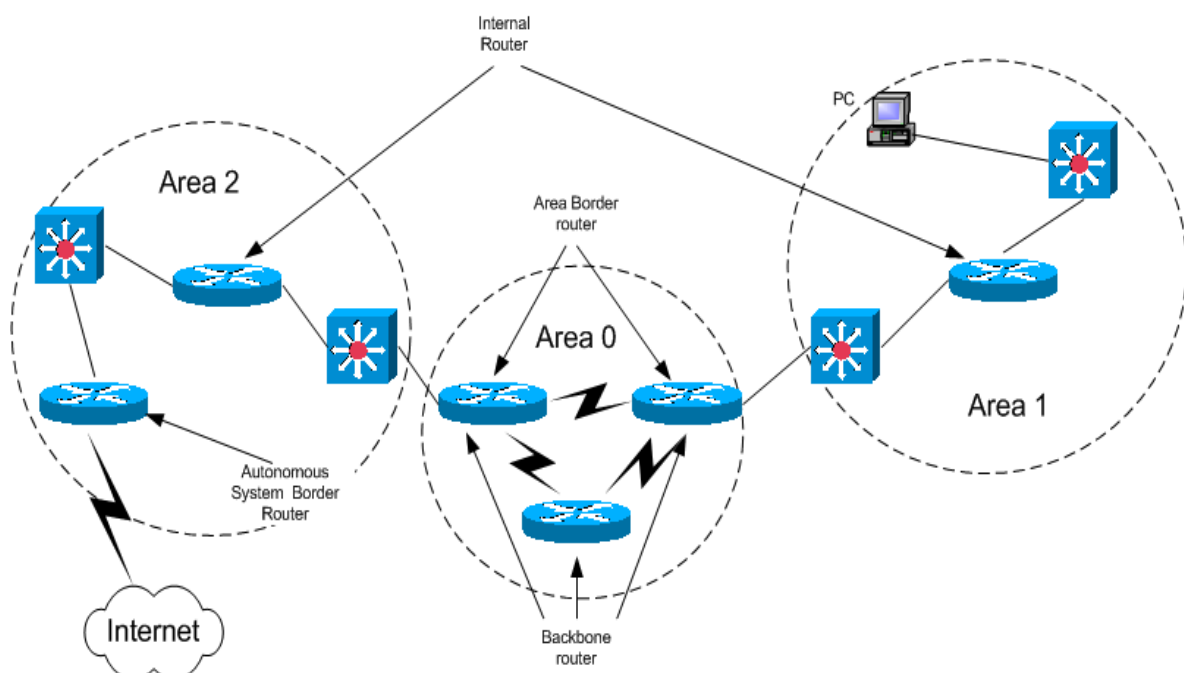


Imagen de la conexión de la red de backbone

Nuestra idea original dentro de madridwireless fue la de “repartirnos” Madrid dependiendo de distritos más o menos de una forma un poco más libre , el tema es conseguir que poco a poco la gente de cada zona (debido a la superficie tan elevada a cubrir) se fuera uniendo por barrios o incluso por comunidades de vecinos para lograr una cantidad considerable de nodos por zonas más delimitadas.

El tema del backbone es como se ve en la imagen, la mejor forma de distribuir la información y los servicios de nuestra red , ya que un grupo de nodos puede tener distribuidos distintos servicios dependiendo de las necesidades logrando así una optimización muy correcta de los recursos en cada zona. Uno de los nodos , por ejemplo, puede tener internet y otro nodo cercano puede tener un servidor de irc y así lograr que no se saturen nuestras maquinitas .

Viendo que necesitamos esa diferenciación entre rangos de IP (Clientes vs Backbone) , nos metimos con lo que en éste punto nos llevaba que es NAT.

Lo que voy a exponer es el caso real de nuestros nodos sin meternos en la conexión entre ellos (OSPF , BGP) que va a ser algo de lo que hablaremos más adelante .Vamos a ver como configuramos nuestro nodo como gateway (router wireless).

La forma es la siguiente una vez configurado el kernel(lo vimos en el número 1 de la serie).

En los nucleos 2.4.x tenemos la posibilidad de configurar nuestra seguridad y enrutar por medio de unas aplicaciones reunidas en un paquete que se denomina Netfilter.

Netfilter (iptables y NAT si queremos diferenciarlos) vino como una solución más moderna a la obtenida por ipchains e ipfwadm en los nucleos 2.2.x ,lo que conseguimos con netfilter es poder hacer de nuestra máquina o bien un firewall muy potente o un poderoso enrutador para nuestra red, ahora mismo nos centramos en NAT.

Lo que vamos a conseguir con NAT es, modificar las cabeceras IP para enrutar los paquetes que van por el interfaz Ethernet al interfaz Wireless (eth0) en resumidas cuentas el ejemplo seria :

```
#!/bin/sh
ipt=/sbin/iptables
echo 1 > /proc/sys/net/ipv4/ip_forward #Se activa el Ip forward(reenvio de
paquetes)

$ipt -F      # limpiamos todas reglas del firewall

$ipt --table nat --flush #limpiamos todas las reglas de Nat

$ipt --table nat --append POSTROUTING --out-interface eth0 -j MASQUERADE

#Se hace la regla de nat esto lo que nos dice es cual es el flujo a capturar
para realizar el reenvio

$ipt --append FORWARD --in-interface wlan0 -j ACCEPT #Hacemos la regla de
Forward para hacer el reenvio por el interfaz wireless en mi caso wlan0
```

Con éstas reglas lo que se ha logrado hacer es que por ejemplo si en el interfaz Ethernet (eth0) tenemos la Ip de clase C podemos tener en el interface wireless una Ip de clase A cómo nuestro estandar nos dicta que debe ser.

B.2) Realicemos un ejemplo más complejo con iptables para securizar nuestro nodo .

Ya que estamos usando iptables y Nat os puedo poner un caso real de un

firewall que hicimos unos amiguetes de madridwireless y yo , la situación es lo suficientemente compleja como para que le hagamos caso y asi poder tener una guia en un caso que nos pueda surgir .

```
#!/bin/bash
ipt=/sbin/iptables
#(sleep 20; /etc/init.d/firewall-off.sh) &
# Opciones del kernel
echo 1 > /proc/sys/net/ipv4/ip_forward
# se limpian las tablas
$ipt -F INPUT
$ipt -F OUTPUT
$ipt -F FORWARD
$ipt -t nat -F

# por defecto, se deniega la entrada a todo
$ipt --policy INPUT DROP
$ipt --policy OUTPUT ACCEPT
$ipt --policy FORWARD DROP

# Puertos por encima del 1024
$ipt -A FORWARD -p tcp ! --syn --dport 1024: -j ACCEPT
$ipt -A FORWARD -p udp --dport 1024: -j ACCEPT
$ipt -A FORWARD -p tcp --dport 1024: -j ACCEPT

# SSH al FIREWALL permitido solo para las macs de los roots (El firewall hacia
de gateway por lo que seria muy util para los plastas)

$ipt -A INPUT -p tcp -m mac --mac-source 00:04:75:8B:DF:74 --dport 22 -j ACCEPT
$ipt -A INPUT -p tcp -m mac --mac-source 00:07:85:92:8B:8C --dport 22 -j ACCEPT

#Reglas

# 1.- Permitir a todo el mundo pasar sslo a los puertos 80 53 443 21 y 22 (FORW)
# 2.- Permitir salir a los mismos puertos y tirarlos al router (DROP o FORWARD)
# 3.- Denegar accesos directos al router y a otros elementos claves de la red
# 4.- Proxy transparente a 10.64.10.3 (NAT)
$ipt -A FORWARD -s 10.64.10.0/24 -p tcp --dport 20 -d ! 172.16.100.1 -j ACCEPT
$ipt -A FORWARD -s 10.64.10.0/24 -p tcp --dport 20 -d ! 172.16.100.2 -j ACCEPT
$ipt -A FORWARD -s 10.64.10.0/24 -p tcp --dport 21 -d ! 172.16.100.1 -j ACCEPT
$ipt -A FORWARD -s 10.64.10.0/24 -p tcp --dport 21 -d ! 172.16.100.2 -j ACCEPT
$ipt -A FORWARD -s 10.64.10.0/24 -p tcp --dport 22 -d ! 172.16.100.1 -j ACCEPT
$ipt -A FORWARD -s 10.64.10.0/24 -p tcp --dport 22 -d ! 172.16.100.2 -j ACCEPT
$ipt -A FORWARD -s 10.64.10.0/24 -p tcp --dport 53 -d ! 172.16.100.1 -j ACCEPT
$ipt -A FORWARD -s 10.64.10.0/24 -p udp --dport 53 -d ! 172.16.100.2 -j ACCEPT
$ipt -A FORWARD -s 10.64.10.0/24 -p tcp --dport 53 -d ! 172.16.100.1 -j ACCEPT

#Para la radio
$ipt -A INPUT -s 10.64.2.0/27 -p tcp --dport 8000 --syn -j ACCEPT
$ipt -A INPUT -s 10.64.2.0/27 -p udp --dport 8000 -j ACCEPT

# Cerramos puertos para edonkey
$ipt -A FORWARD -s 0.0.0.0/24 -p udp --dport 4665 -j DROP
$ipt -A FORWARD -s 0.0.0.0/24 -p tcp --dport 4661 -j DROP
$ipt -A FORWARD -s 0.0.0.0/24 -p tcp --dport 4662 -j DROP

# Permitir a la red el forward y el NAT
$ipt -A FORWARD -s 10.64.10.0/24 -j ACCEPT
$ipt -t nat -A POSTROUTING -o eth2 -j MASQUERADE
```

```
# Ataques -> denegacion
```

```
$ipt -A FORWARD -p tcp --syn -m limit --limit 1/s --limit-burst 1 -j ACCEPT
```

```
$ipt -A FORWARD -p tcp --tcp-flags SYN,ACK,FIN,RST RST -m limit --limit 1/s --limit-burst 1 -j ACCEPT
```

```
$ipt -A FORWARD -p icmp --icmp-type echo-request -m limit --limit 1/s --limit-burst 1 -j ACCEPT
```

Analizamos el ejemplo de una forma rápida : Como se ve con unas simples reglas se puede lograr desde la securización contra ataques de denegación(D.O.S) incluso podemos filtrar algunos puertos cómo pueden ser los puertos del Edonkey o de cualquier cosa que queramos tener cerrado cara al público o activar puertos para servicios determinados cómo fue en éste caso para una radio. En resumen es muy simple siguiendo un ejemplo cómo éste hacer un firewall o sistema de seguridad medianamente complejo para que nuestro nodo funcione de una forma más optima.

Si logramos tener un sistema fiable podemos jugar con parámetros más complejos cómo por ejemplo podríamos hacer un sistema de QoS(Quality of Services) o de packet Matching .

En el siguiente número seguiremos con conceptos también muy complejos como van a ser el manejo de los protocolos OSPF y BGP y el por qué los utilizamos para la interconexión de nodos .

Una vez más me gustaría dar las gracias a todos mis ex-compañeros de meta4 javi , chemita, señora doña lourdes ,a la futura mama Bettina y cómo no a Boludo por aguantar las charlas que se han comido sobre mi sueño de crear una red libre e independiente y autogestionada como es Madridwireless .

Licencia del Documento

Copyright (c) 2004 José Javier Espejo Corral(qemm@madridwireless.net) Se otorga permiso para copiar, distribuir y/o modificar este documento según los términos de la Licencia GNU Para Documentación Libre (GNU Free Documentation License), versión 1.2 o cualquier versión posterior publicada por la Free Software Foundation. Esta licencia está disponible en <http://gnu.org/copyleft/fdl.html>.