Bisoños Usuarios de GNU/Linux de Mallorca y Alrededores | Bergantells Usuaris de GNU/Linux de Mallorca i Afegitons

Configuración de PPP sobre RDSI con *bandwidth on-demand* (17025 lectures)

Per **Ricardo Galli Granada**, ***gallir*** (http://mnm.uib.es/gallir/)

Creado el 01/12/2001 22:28 modificado el 01/12/2001 22:28

En este artículo resumo los paquetes y pasos necesarios para configurar y hacer funcionar el PPP (sincrónico) sobre una tarjeta RDSI (ISDN) interna en Debian. También explico como configurar la `iptables` para hacer de NAT (*masquerading*) a toda la red LAN interna y proteger los puertos del servidor. Además explico como hacer funcionar el MPPP (*multilink PP*P) para que establezca la llamada sobre el segundo canal cuando el tráfico supere un cierto umbral (*bandwidth on-demand*).

Paquetes necesarios:

- isdnutils, o por lo menos algunos de sus siguientes sub-paquetes
  - isdnutils-base
  - ipppd
- ibod
- iptables

Breve introducción al funcionamiento

Como sabréis el PPP es un protocolo punto a punto que sirve para conectar redes de ordenadores usando una línea y haciendo que los dos ordenadores o *routers* en los extremos se encarguen del enrutado. Además el protolo tiene varias optimizaciones y opciones de compresión para hacer un mejor uso de la línea.

Hay dos tipos de protocolos PPP, el asincrónico que es que se usa normalmente en los modems o dispositivos similares conectados a través de una línea serie. El que a nosotros nos interesa en este artículo es el **PPP sincrónico o SyncPPP**.

Las **isdn2linux**, incluidas en el kernel 2.4, son las que se encargan no sólo de gestionar las tarjetas RDSI, sino que también se encargan de las negociación de los diversos protocolos necesarios para establecer la llamada y de permitir la definición de interfaces virtuales, que nos permiten trabajar con la conexión *dial-up* como si se tratase de una interfaz de red normal.

Interfaces virtuales

Las interfaces virtuales que se usan con el SyncPPP son las **ippp0, ippp1, ippp2**... etc. Cada una de ellas representa un canal B del RDSI que se usa para llamar a nuestro proveedor.

¿Que ventajas tiene? Básicamente que podemos definir nuestras tablas de rutas IP usando dichas interfaces virtuales en caso de configuraciones más complejas, por ejemplo llamar a distintos números y opciones dependiendo de la IP a la que queremos acceder. En el caso más común, la conexión a Internet, no hace falta especificar casi nada porque de eso encargará de hacerlo los script **/etc/init.d/isdnutils** de las isdnutils que veremos más adelante.

Daemons y scripts

Para que la llamada al proveedor sea automática, se necesita al **ipppd**. Este daemon es arrancado por el script **/etc/init.d/isdnutils** luego de configurar todas las interfaces virtuales y las características del RDSI.

Si además queremos hacer *bandwidth on-demand*, se pondrá en marcha el **ibod** que es arrancado automáticamente por

el script ubicado en **/etc/ppp/ip-up.d/** luego de establecerse la llamada. Este *script* (normalmente llamado 00-ibod) se genera al instalar el paquete ibod.

Configuración de módulos del kernel

Por supuesto, si no estáis seguro que vuestro kernel de Linux tenga incluido el soporte para iptables y RDSI (*ISDN*) deberéis recompilarlo. El tema de configuración del kernel, si nunca lo has hecho, escapa al objetivo del este artículo, pero es muy raro que alguien que tenga Debian nunca se haya interesado por compilar un kernel ;-)

De todas formas, la configuración no es compleja. A continuación muestro las mías en formato de fichero `.config`, pero os podéis hacer una idea de lo que significa cada una de ellas. Mis opciones relevantes tiene las siguientes opciones habilitadas (en mi caso tengo una tarjeta Conceptronic PCI, que es similar a una Dynalink, usa el chipset Winbond):

```
#
# ISDN subsystem
#
CONFIG_ISDN=y
CONFIG_ISDN_PPP=y
CONFIG_ISDN_PPP_VJ=y
CONFIG_ISDN_MPP=y
CONFIG_ISDN_PPP_BSDCOMP=m
#
# Passive ISDN cards
#
CONFIG_ISDN_DRV_HISAX=m
#
#   D-channel protocol features
#
CONFIG_HISAX_EURO=y
#
#   HiSax supported cards
#
CONFIG_HISAX_W6692=y
#
#   IP: Netfilter Configuration
#
CONFIG_IP_NF_CONNTRACK=m
CONFIG_IP_NF_FTP=m
CONFIG_IP_NF_IPTABLES=m
CONFIG_IP_NF_MATCH_LIMIT=m
CONFIG_IP_NF_MATCH_MARK=m
CONFIG_IP_NF_MATCH_STATE=m
CONFIG_IP_NF_FILTER=m
CONFIG_IP_NF_TARGET_REJECT=m
CONFIG_IP_NF_NAT=m
CONFIG_IP_NF_NAT_NEEDED=y
CONFIG_IP_NF_TARGET_MASQUERADE=m
CONFIG_IP_NF_NAT_FTP=m
```

Especificación del alias del módulo Hisax

Para mantener la claridad, especificaremos un alias del modutil para que se cargue el módulo necesario y sus argumentos. Para ello editamos el fichero **/etc/modutils/ppp** y agregamos la siguientes líneas:

```
options hisax   type=36
alias isdn        hisax
```

Por supuesto, debéis poner en el type=N el tipo de chipset que tiene vuestra placa RDSI. Esta documentación se encuentra en `linux/Documentacion/isdn/README.HiSax`. Si tenéis dudas sobre el tipo de placa que tenéis, haciendo un **`lspci`** obtendréis el tipo. En mi caso es:

```
02:0c.0 Network controller: Winbond Electronics Corp: Unknown device 6692
```

Una vez hecho esto, **ejecutamos `update-modules`** para que se actualice el fichero /etc/modules.conf. Si queremos probar que funcione correctamente, podemos hacer un **`modprobe isdn`** y el módulo debería cargarse y dar el tipo de tarjeta y RDSI que tenemos. Algo así como:

```
Found: Winbond W6692, I/O base: 0xd000, irq: 9
HiSax: W6692 config irq:9 I/O:d000
W6692: Winbond W6692 version (0): W6692 V00
W6692 ISTA=0x0
W6692 IMASK=0xFF
W6692 D_EXIR=0x0
W6692 D_EXIM=0xFF
W6692 D_RSTA=0xA0
Winbond 6692: IRQ 9 count 0
Winbond 6692: IRQ 9 count 6
HiSax: DSS1 Rev. 2.30.6.2
HiSax: 2 channels added
HiSax: MAX_WAITING_CALLS added
```

Si esta información aparece, ya tenemos solucionado la parte del kernel y los módulos, ahora pasaremos al paso siguiente.

Generación de los ficheros esenciales con isdnconfig

Para poder configurar las interfaces virtuales, necesitaremos de dos scripts que definen las características de:

1. dispositivo de red (**/etc/isdn/device.ipppN**, donde **N** es >= 0) y
2. la configuración de la interfaz virtual (**/etc/isdn/ipppd.ipppN**, donde **N** es >= 0) para el ipppd.

Afortunadamente, el comando **`isdconfig`** se encargará de generar ambos scripts. Ejecutamos isdnconfig que nos mostrará el siguiente menú:

```
  Isdnutils configuration
=======================

1       network devices
2       synchronous ppp daemon
3       modem emulation
4       voice box configuration (disabled, ...)

Q       Quit
```

Primero seleccionaremos la **opción 1** y pondremos el nombre **ippp0**. Una vez que haya grabado, seleccionaremos la **opción 2** y le pondremos el mismo nombre. Como resultado habremos obtenido:

- /etc/isdn/device.ippp0
- /etc/isdn/ipppd.ippp0

Configuración del device.ippp0

Este fichero se encargará de hacer:

- realiza las comprobaciones,
- cargará los modulos necesarios,
- configura las interfaces virtuales,
- configura el masquerading y los fitros,
- cambia las rutas IP.

Los lugares que hay que modificar están marcados con **`XXX_`** para que sea más fácil la ubicación.

Sin embargo, le agregaremos una línea adicional casi al principio (después de set) que se encargará de verificar que el módulo de hisax esté cargado (usando el **alias isdn** que hemos definido antes). Nos quedará:

```
set -e # exit on _any_ error
modprobe isdn
```

No os olvidéis de comentar la siguiente línea al principio, sino luego no funcionará.

```
echo "Warning! $0 not configured yet! Aborting..."; exit 1
```

Los parámetros adicionales que hay que indicar son, usando mi ejemplo son:

```
  # NUMERO FALSO
# TENGO IP FIJA, dejar 10.0.0.1 caso contrario
LOCALIP=555.555.555.5
REMOTEIP=10.0.0.2

LOCALMSN=971764080 # __tu__ numero de RDSI, no obligatorio
REMOTEMSN=97111122 # numero del ISP
LEADINGZERO=''  # no tenemos zona en europa

DIALMODE=auto # llamada automática
   isdnctrl huptimeout ${device} 180 # segundos de inactividad
                                     # para colgar

# FIREWALL RULES    XXX_
# Ver la iptables ;-)
```

Configuración de iptables en el device.ippp0

Ahora viene lo más interesante, a continuación del último comentario anterior pondremos poner los comandos del iptables para definir el masquerading y filtros que afectarán a esta interfaz virtual cuando nos conectemos.

En mi caso, mis IP de la red LAN privada son 192.168.0.0/24 (194.224.0.0 a 194.224.0.255) y quiero hacer masquerading para toda esa red, además sólo quiero habilitar el acceso externo a mi Linux a los puertos HTTP, IDENT (o AUTH) y SSH, además de responder a los paquetes ICMP (ping). Las reglas quedan de la siguiente forma:

```
    # FIREWALL RULES    XXX_
    modprobe iptable_nat
    iptables -F
    iptables -t nat -F
    iptables -t nat -A POSTROUTING -s 192.168.0.0/24 -d 0.0.0.0/0  -j MASQUERADE
    iptables -A INPUT -i ippp0 -p ICMP -j ACCEPT
    iptables -A INPUT -i ippp0 -p TCP --dport 80 -m state --state NEW -j ACCEPT
    iptables -A INPUT -i ippp0 -p TCP --dport ssh -m state --state NEW -j ACCEPT
    iptables -A INPUT -i ippp0 -p TCP --dport auth -m state --state NEW -j ACCEPT
    iptables -A INPUT -i ippp0 -m state --state NEW,INVALID -j DROP
    iptables -A FORWARD -i ippp0 -m state --state NEW,INVALID -j DROP
    echo 1 > /proc/sys/net/ipv4/ip_forward
```

Una vez hecho esto, sólo nos queda terminar de definir los parámetros del PPP en `ipppd.ippp0` y las claves de autentificación al servidor en `/etc/ppp/pap-secrets` (en caso de usar PAP).

Configuración del PPP

En este **ipppd.ippp0** se especifican los parámetros del ppp que usará la interfaz ippp0 para conectarse. Los parámetros que **sí** hay que cambiar son:

```
idle 180        # tiempo límite sin tráfico (segundos)
+mp             # habilitar multi line ppp
name gallir     # mi nombre de usuario
noauth          # no solcitar autorización al servidor
defaultroute    # definir como ruta por defecto (a Internet)
ipcp-accept-local   # Acepta nuestra IP
```

```
ipcp-accept-remote  # Acepta la IP del remoto
```

De nuevo, no os olvidéis de comentar la siguiente línea al principio, sino luego no funcionará.

```
echo "Warning! $0 not configured yet! Aborting..."; exit 1
```

Autentificación

Las claves de autentificación están almacenadas en `/etc/ppp/pap-secrets` o `/etc/ppp/chap-secrets` dependiendo si la autentificación que usa el ISP es PAP o CHAP (en mi caso PAP). Por ello agregué una línea (y comenté todas las demás) en el fichero **/etc/ppp/pap-secrets**. La primera palabra es el nombre de usuario que usé en ipppd.ippp0 (gallir), la segunda es para indicar el nombre del servidor de acceso, como en este caso no cabe, se usa el comodín (*), la tercera palabra es la clave de acceso.

```
gallir * mi_clave_de_acceso
```

Ala!!! Si no me olvidé nada (me llevó más tiempo escribir hasta aquí que configurar mi ISDN por primera vez ;-), y habéis seguido todos los pasos, **ya deberías tener funcionando la conexión con un sólo canal**. Haced un **`/etc/init.d/isdnutils start`** y a probar...

A continuación veremos como configurar el ibod y un dispositivo RDSI esclavo (**slave**) para conectar el segundo canal.

Bandwidth on-demand

Lo primero que tenemos que hacer para poder usar un segundo canal es definir una segunda interfaz ippp como esclava de la ippp0 que hemos usado para configurar la conexión. Pero no os preocupéis, ya os dije que los script llamados por /etc/init.d/isdnutils se ocupan de todo esto. En corto: ¿como se hace para configurar una interfaz esclava? Muy fácil, con hacer un enlace simbólico del tipo `device.ippp0+1 -> device.ippp0`

`ln -s /etc/isdn/device.ippp0 /etc/isdn/device.ippp0+1`

ya le estamos diciendo a las isdnutils que defina al **ippp1 como esclava de ippp0**. Muy fácil. Sólo queda **configurar los parámetros del ibod**.

Configuracion del ibod

Para especificar los parámetros hay que editar el fichero **`/etc/isdn/ibod.cf`**. Los parámetros que uso son:

```
INTERVAL    500
LIMIT       80
FILTER      30
STAYUP_TIME 120
```

INTERVAL indica, en milisegundos, cada cuanto se tomarán las muestras del tráfico en el ippp0. En el ejemplo, cada medio segundo.

LIMIT es el porcentaje de ancho de banda consumido para que se active el segundo canal.

FILTER especifica durante cuantas muestras se debe superar el LIMIT para que e active el segundo canal. En el ejemplo 15 segundos (30 muestras cada medio segundo).

STAYUP_TIME especifica el tiempo mínimo que permanecerá conectado el segundo canal antes de desconectarse.

En resumen, si durante 15 segundos mi tráfico es superior al 80% de 64 kbps, se hace la segunda llamada que permanecerá conectada todo el tiempo que ese tráfico sea superado o un mínimo de 2 minutos.

Ala!, ya está otra vez, haced un **`/etc/init.d/isdnutils restart`** y ya debería funcionar todo.

Os recomiendo que os miréis los paquetes **isdnbutton**, **isdnlog**, e **isdnutils-xtools**. Sobre todo este último para ver el uso de la red, también sirve para controlar las conexiones con los botones del ratón... En otros artículos os explicaré como configurar el ntp y el fetchmail para ahorrar llamadas y evitar las *tormentas de llamadas* o estar conectados todo el día.

Espero que haya sido de ayuda a los Debianitas.

E-mail del autor: gallir _ARROBA_ uib.es
Podrás encontrar este artículo e información adicional en: http://bulma.net/body.phtml?nIdNoticia=1036