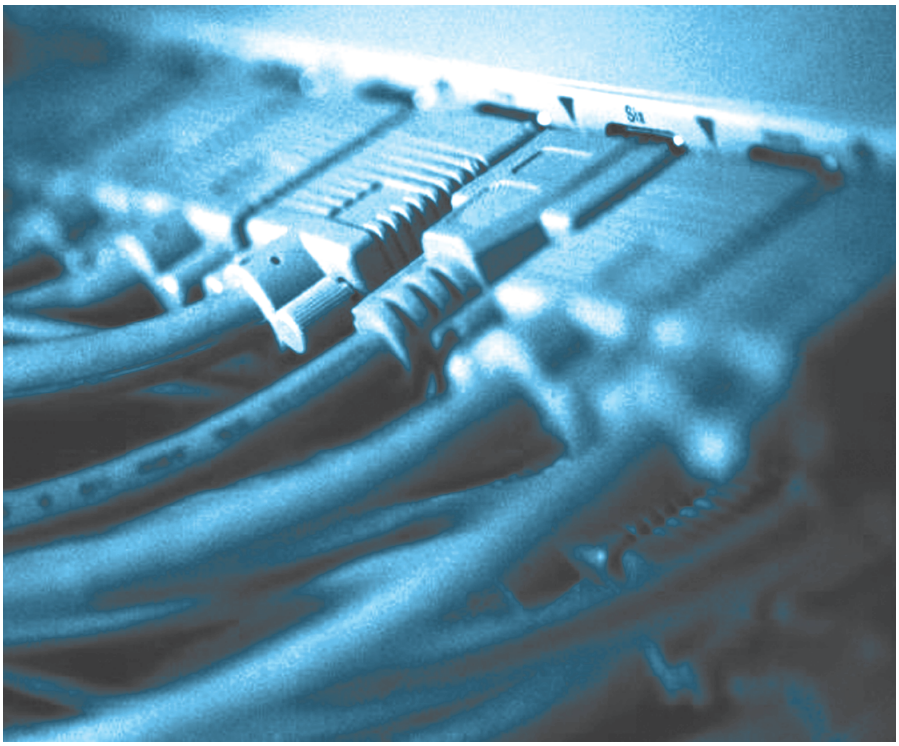
Linux in LANs

Safely Sharing Data

It does not matter if you are simply connecting a host to a network or that you use a laptop at both home and at the office to log on to different networks every day; the mystery of connecting up to a network can remain the same.

BY MARC ANDRÉ SELIG



Nearly all of us will, at some time in their career, have cast jealous glances at experts connecting their Linux laptops up to the networks at the office, at home, or even at conferences and immediately getting on to the Internet, as if nothing had happened in the meantime – this is especially true if the two or three machines in our SOHO network have been giving us a hard time. Fortunately, there is no black magic involved – it is just a question of understanding the basic procedures.

Thanks to the sophisticated setup tools that are supplied with most modern Linux distributions, we often require very little user intervention to connect to a local network. Even if things do not work straight away, a few clicks in the appropriate menus will normally see the laptop happily connected to a new network. A fully automated network configuration supplied by DHCP [1], for example, will allow your machine to retrieve any necessary information automatically.

Even if automation fails there is no need to panic. In this article we will be taking a look at the configuration steps those GUI tools perform inside to ensure that the connection to your **LAN** and from there to the Internet works. Anything those tools can do, can of course be done manually.

Hardware

At the lowest level, a machine requires a network device with appropriate cabling to connect to a network (refer to [2] for more information on wireless LANs). The network device can be embedded in the machine, reside on an adapter card or – as is the case for most modern lap-

GLOSSARY

LAN:
The expression "Local Area Network" describes a network of neighboring computers typically connected by simple network media (although they may connect to a wireless LAN, such as WaveLAN [2]).
The most important difference between a LAN and other types of networks is the fact that a LAN does not use leased lines, such as serial, ISDN or DSL lines. The WAN or "Wide Area Network" is the opposite of a LAN.

PCMCIA:
The "Personal Computer Memory Card International Association" published a PC card standard of the same name (amongst other things). The standard permits uniform hardware extensions for laptops and similar devices. The adapters are typically referred to as PCMCIA adapters, a name that immediately identifies the device type, in contrast to a "PC Card" which could be anything. CardBus is a successor of the PCMCIA standard and was designed for faster data transfer rates.

USB:
The "Universal Serial Bus" is a convenient way of attaching peripheral hardware to a computer. Even primitive USB devices like mice or keyboards need a lot of embedded intelligence, although this normally provides for fully automatic configuration. The USB standard specifies that the driver installation should not require user intervention! That does not always work on Linux, but that's no big deal, as it doesn't work on other systems either...

Listing 1: A configured network adapter

```
mas@swan:~$ /sbin/ifconfig
eth0      Link encap:Ethernet  HWaddr 00:00:C0:77:D8:F5
          inet addr:172.16.45.12  Bcast:172.16.45.255
Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:2565 errors:0 dropped:0 overruns:0 frame:0
          TX packets:10723 errors:21 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:596219 (582.2 Kb)  TX bytes:3753306 (3.5 Mb)
          Interrupt:9 Base address:0x2000
[...]
```

tops – be integrated on the motherboard. Of course you might also have a **PCMCIA** adapter or an external **USB** device.

A driver is required to allow Linux to communicate with the adapter. In the "Plug & Pray" age the driver should hopefully be loaded automatically. You can use one of the most important network configuration commands to ensure that this step has actually worked: *ifconfig*. The command without any parameters displays the current configuration information for the network interface (Listing 1). If you are not currently working as *root*, *ifconfig* will probably not be in your path. In this case, you should supply the path explicitly: */sbin/ifconfig*.

Every Linux system displays at least one interface, the local "loopback"-device *lo*. But LAN jockeys will only be interested in interfaces that start with the magic letters *eth* for "*Eth*ernet". The output in Listing 1 shows the most important data for each network connection:

- The hardware address (HWaddr) identifies the network adapter in the so-called Data Link layer, that is at a purely hardware based level. The six colon-separated bytes are also referred to as the MAC address which is a globally unique address, unless a malevolent hacker has been messing around with it, that is. Well managed networks tend to integrate the MAC address in the configuration – that is why you should avoid replacing your office desktop with your laptop without any preparatory steps.
- The IP address (inet addr) identifies your computer on the network. This is the address that the programs on your computer use. But although the name might suggest otherwise, the IP address is not necessarily accessible via the Internet. The address shown in our example, 172.16.45.12, is part of a standardized pool of private IP addresses used for private networks.
- The broadcast address (Bcast) and the netmask ("network mask", Mask) provide a more accurate description of the network the computer attaches to. In simple terms, part of the enormous IP address space is specified as the network neighborhood. It used to be possible to derive these parameters from the IP address, but this is no longer possible today. Fortunately, these parameters are not that important for a stand-alone system such as a laptop. If you do happen to know the correct values, ensure that you set them correctly – anything else often leads to a disaster.

These values are followed by a whole bunch of statistics about the interface. The received (*RX*) and transmitted (*TX*) packet counters are particularly interesting.

A word of warning: Our example does not indicate that *10723 errors* have occurred, but merely *errors:21*. The Linux *ifconfig* display is rounded off by information on the interrupt used and other hardware data for the network interface card. If your network card was recognized and configured without any trouble, but still does not work, you should look at these figures first!

There are two possible reasons for *ifconfig* failing to display an *eth0*. The less likely, but less troublesome variant would be that your distribution has not accessed the network adapter yet, and is waiting for you to wake it up.

How do you wake up a sleeping NIC? A kiss is not going to help, so let's try *ifconfig* instead:

```
mas@swan:~$ su
Password: [secret_root_password]
root@swan:~mas# /sbin/ifconfig ➥
eth0 up 172.16.45.12 netmask ➥
255.255.255.0
```

If your kernel recognizes the network card, you are done. If the other possibility now rears its ugly head, you will see error messages such as the following:

```
eth0: unknown interface: ➥
No such device
SIOCSIFADDR: No such device
```

This indicates that the kernel module that contains the driver for the network card is missing, or was not loaded correctly. The system may require more details on the hardware. In this case, you should try searching with Google or reading appropriate newsgroups such as *comp.os.linux.hardware* or *comp.os.linux.networking*.

On the Net!

Armed with just the basic equipment discussed so far, your Linux machine should be able to exchange packets with its neighbors. If you know the IP address of a computer in the same LAN, you can test your system using the steps shown in Listing 2.

The *ping* command transmits test packets to the specified target. If the packets reach their target, the target machine will send "pong" packets back. This makes *ping* a perfect network diagnostic tool. You will need to type [Ctrl-C] to quit the continual pinging, when you finally get bored.

Listing 2: Is anybody out there?

```
mas@swan:~$ ping 172.16.45.1
PING 172.16.45.1 (172.16.45.1) from 172.16.45.12 : 56(84) bytes of data.
64 bytes from 172.16.45.1: icmp_seq=0 ttl=62 time=210 usec
[...]
```

Linux is still incapable of talking to the outside world, that is to other networks – unless of course your distribution offers a tool that automagically takes care of this behind your back…

Linux can deliver packages within its local network directly (within the address block covered by the network configuration to be more precise). Of course there are a lot of sophisticated things going on in the background, but you typically do not need to pay attention to them.

Linux needs help to deliver packets to external networks. Instead of delivering the packets itself, Linux will send them to another computer that will take care of the job. This other computer is referred to as a gateway or default router. This may be a WAN access point at your office, or an ISDN adapter or modem at home.

The so-called routing table describes what packets are sent where. You can use the *route* command (or */sbin/route*, if you are not working as *root*) to list the table. If you really want to perform a manual configuration, add the *-n* flag as shown in Listing 3 to prevent delays caused by translating IP addresses to symbolic names.

Each line contains a routing table entry. Take a look at the first and last columns for a quick overview. The first column with the *Destination* shows the data packets the entry refers to (experts will quickly notice that the third column is also significant. The rest of us will just have to rely on intuition to do the right thing). Thus, the first line in Listing 3 shows what happens to packets sent to IP addresses that start with *172.16.45*. The last column shows what *Iface* – "*inter*face" will be used to transmit the packets, in this case *eth0* as discussed previously.

The second line follows the same pattern structure and defines that packets for IP addresses starting with *127* will be sent to the *lo* interface. As previously mentioned, this is the "loopback" device, which a machine can use to talk to itself.

The third line is a kind of joker. Any target addresses not previously covered follow the rule defined in this line. And this is where the gateway, or default router appears. Any packets whose target address does not start with *172.16.45* or *127* are passed on by Linux to a machine with IP *172.16.45.1*. This is also the machine that will take care of everything else.

Practical, isn't it? Your computer only knows two networks, and one of those is its own backyard, but it can still talk to the whole world.

The bad news is that users working with laptops can expect to run into trouble with default routes sooner or later. After all, you can not just log on to the office network, your SOHO network, use a mobile phone, or a modem in your hotel room, and simply hope that your laptop does not make any mistakes!

Many distributions really start to panic when faced with promiscuous traffic of this form. The most common symptom of promiscuity is a broken default route. The network connection still works okay, but the default route is missing or points to a black hole. Listing 4 shows an example.

This machine's LAN connection (*eth0*) is correctly configured and works fine. As it happens, it is not connected to the LAN at present, but has just dialed up the network via a mobile phone (*ppp0*). Unfortunately, the default route for the LAN is still set, and Linux will now attempt to send any packets destined for the Internet via the LAN adapter and not via the mobile phone. Of course, that is not going to work.

Help is at hand in the form of the *route* command, which can be used to manipulate the routing table (Listing 5). *root* first invokes *route del* (as in "*del*ete") to remove the default route and adds the new route by typing *route add*. This applies for *-net* (that is the network and not only to a single address), *default* (that is for any unspecified IP addresses) and points to the *gw* ("*G*ate*w*ay") *195.71.150.164*. It is a good idea to copy this IP address from the *route* table viewed previously.

Now, what was your name?

So far we have only referred to computers and networks in terms of IP addresses, but IP addresses are no use whatsoever under real life conditions.

After all, who wants to type *208.62.23.150* if they can reach the target far more easily by typing *www.zpid.com*?

Name resolution is provided by a service called DNS, "*D*omain *N*ame *S*ervice [3]. To be more precise, it is the so-called resolver that is responsible for translating symbolic names into IP addresses, and vice versa. In contrast to many other UNIX services the resolver is not a daemon but part of a standard UNIX library.

However, the resolver does not know all of the myriad computer names on the internet itself; instead it asks a DNS server on the Web. So, if you want to work with symbolic host names (and you should!), you will need to tell

Listing 3: The Route

```
mas@swan:~$ /sbin/route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
172.16.45.0     0.0.0.0         255.255.255.0   U     0      0        0 eth0
127.0.0.0       0.0.0.0         255.0.0.0       U     0      0        0 lo
0.0.0.0         172.16.45.1     0.0.0.0         UG    0      0        0 eth0
```

Listing 4: Somewhat Confused

```
mas@swan:~$ /sbin/route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
195.71.150.164  0.0.0.0         255.255.255.255 UH    0      0        0 ppp0
172.16.45.0     0.0.0.0         255.255.255.0   U     0      0        0 eth0
127.0.0.0       0.0.0.0         255.0.0.0       U     0      0        0 lo
0.0.0.0         172.16.45.1     0.0.0.0         UG    0      0        0 eth0
```

Listing 5: A new default route

```
root@swan:~mas# /sbin/route del default
root@swan:~mas# /sbin/route add -net default gw 195.71.150.164
```

Linux where to find a DNS server. The bad news is that you need a different DNS server for each access type – office, mobile, home etc.

The current configuration file for the resolver is called */etc/resolv.conf*, and the only relevant entry in the file is the *nameserver*.

You can either adopt a cautious approach and leave all the other entries in this file untouched, or "boldly throw" them out. The resulting *resolv.conf* should then look as follows:

```
nameserver 208.62.23.150
```

Short and sweet. Who do you ask for details on the name server? In a real LAN situation you would simply ask your friendly network administrator. At home you are your own admin and you need to simply enter your server's IP.

Things are a little more complicated for those with dialup connections, but fortunately you can normally rely on your distribution to take care of this automagically. Usually the distribution will query the dialup server for the authorized DNS server and update your */etc/resolv.conf* using the new data. Mad idea? Well, it does come from Microsoft.

Name Resolution for Experts

Experts and purchasers of modern distributions should know that resolver configuration involves more than the steps just discussed.

Firstly, some distributions use so-called caching name servers. That means every computer runs a DNS server, even your laptop. The name server corresponds with the local software and relays queries from the local software to the internet. In a scenario of this type, you are well advised to steer clear of */etc/resolv.conf* – as the file will only refer to the local host in this case.

However, you can manipulate the local DNS server configuration instead, although this is often unnecessary as the DNS cache often does not depend on the Internet provider's DNS server.

Secondly, packet filters (often referred to as firewalls) have been introduced to the major distributions. Of course, they need to be informed of any name server address changes and updated accordingly. Fortunately, this is another step that is typically performed automatically.

Web and Mail

When moving from one LAN to another, one or two things that are relevant to application software may change. This is not typically a problem, but you may need to look into two areas: Web proxies and email.

A proxy will save a copy of a website transferred to it. If another computer request the same page, it will first access its local memory before it searches the internet – this saves both time, in that the local memory access is faster than downloading over your connection and money, where the less you have to access means others can access quicker and so you reduce their time and in turn their costs. Dynamic WWW pages have seen the gradual demise of proxies as you need the current dynamically generated information and not that of the old cached proxy.

If you do use a proxy, you will probably need to update your configuration if you change your Internet access point.

When sending email, it is uncommon to transfer messages directly to the receiver's mail server nowadays. Many professional mail providers explicitly prohibit this variant for security reasons. That means that both your mail program (e.g. *kmail* or Mozilla) and your local mail server (e.g. *sendmail*, *qmail* or *postfix*) need a relay that will receive and forward email messages. The relay works along the same lines as a mail box. Your computer puts your outgoing mail in the box and someone comes along and takes care of delivering or forwarding your messages to the recipient. The relay or mail box is known as a *smart host*.

Your environment and your personal preferences will define who plays the role of the smart host. In principle, both your email provider and your Internet provider can act as a smart host. I personally prefer the former. If you opt to transmit mail via your Internet provider, moving from one network to another will involve changing your smart host. So do not forget to update your mail configuration!

Prospects

This short tour allowed you a quick glance at the most important nuts and bolts that you will need to loosen and retighten when moving your computer from one (or no) network to another. The manpages for *ifconfig* and *route* contain a lot more detail and are specifically recommended.

Both of these tools provide critical functionality in networked environments, but you can experiment on stand-alone systems. Rebooting normally resolves any problems that have occurred, and while you are experimenting, why not take a detour to the hidden depths of */proc/net* and */proc/sys/net*?

If you are interested in more functionality and magic, you should check out DHCP. Nearly all the issues and challenges discussed in this article can be resolved by a fully automatic network configuration. Of course, it won't always work, but it is surprisingly good. And being in the know as regards the background operations can not be a bad thing … ■

INFO

[1] Bruce Richardson, Linux Magazine, Issue 23, p44

[2] Cover story Linux Magazine, Issue 25

[3] Marc André Selig, Linux Magazine, Issue 26, p86

THE AUTHOR

Marc André Selig spends half of his time working as a scientific assistant at the University of Trier and as a medical doctor in the Schramberg hospital. If he happens to find time for it, his currenty preoccupation is programing web based databases on various Unix platforms.