Linux Networking-concepts HOWTO

Rusty Russell
$Revision: 453 $ $Date: 2001-07-29 06:45:11 +0200 (dom, 29 jul 2001) $

This document describes what a network (such as the Internet) is, and the very basics of how it works.

Contents

1 Introduction

Welcome, gentle reader.

I have written a number of networking HOWTOs in the past, and it occurred to me that there's a hell of pile of jargon in each one. I had three choices: my other two were ignoring the problem and explaining the terms everywhere. Neither was attractive.

The **point** of Free software is that you should have the freedom to explore and play with the software systems you use. I believe that enabling people to experience this freedom is a noble goal; not only do people feel empowered by the pursuit (such as rebuilding a car engine) but the nature of the modern Internet and Free software allows you to share the experience with millions.

But you have to start somewhere, so here we are.

2 What is a 'computer network'?

A computer network is just a set of stuff for nodes to talk to each other (by 'nodes' I mean computers, printers, Coke machines and whatever else you want). It doesn't really matter **how** they are connected: they could use fiber-optic cables or carrier pigeons. Obviously, some choices are better than others (especially if you have a cat).

Usually if you just connect two computers together, it's not called a network; you really need three or more to become a network. This is a bit like the word 'group': two people is just a couple of guys, but three can be an 'group'. Also, networks are often hooked together, to make bigger networks; each little network (usually called a 'sub-network') can be part of a larger network.

The actual connection between two computers is often called a 'network link'. If there's a bit of cable running out of the back of your machine to the other machines, that's your network link.

There are four things which we usually care about when we talk about a computer network:

Size

If you simply connect your four computers at home together, you have what is called a LAN (Local Area Network). If everything is within walking distance, it's usually called a LAN, however many machines are connected to it, and whatever you've built the network out of.

The other end of the spectrum is a WAN (Wide Area Network). If you have one computer in Lahore, Pakistan, one in Birmingham, UK and one in Santiago, Chile, and you manage to connect them, it's a WAN.

Topology: The Shape

Draw a map of the network: lines are the

2 (network links), and each node is a dot. Maybe each line leads into a central node like a big star, meaning that everyone talks through one point (a 'star topology'):

```
     o   o   o
      \_ | _/
        \|/
   o-----o-----o
       _/|\_
      /  |  \
     o   o   o
```

Maybe everyone talks in a line, like so:

```
     o------o------o-------o--------o
     |                              |
     |                              |
```

```
|                              o
|                              |
o                              |
                               o
```

Or maybe you have three subnetworks connected through one node:

```
            o
o           |  o--o--o
|           |  |
o--o--o--o--o  o
       \       |
        o------o
       /       |
o--o--o--o--o  o
|           |  |
o           |  o--o
            o
```

You'll see many topologies like these in real life, and many far more complicated.

Physical: What It's Made Of

The second thing to care about is what you've built the network out of. The cheapest is 'sneakernet', where badly-dressed people carry floppy disks from one machine to the others. Sneakernet is almost always a 2 (LAN). Floppies cost less than $1, and a solid pair of sneakers can be got for around $20.

The most common system used to connect home networks to far bigger networks is called a 'modem' (for MODulator/DEModulator), which turns a normal phone connection into a network link. It turns the stuff the computer sends into sounds, and listens to sounds coming from the other end to turn them back into stuff for the computer. As you can imagine, this isn't very efficient, and phone lines weren't designed for this use, but it's popular because phone lines are so common and cheap: modems sell for less than $50, and phone lines usually cost a couple of hundred dollars a year.

The most common way to connect machines into a LAN is to use Ethernet. Ethernet comes in these main flavors (listed from oldest to newest): Thinwire/Coax/10base2, UTP (Unshielded Twisted Pair)/10baseT and UTP/100baseT. Gigabit ethernet (the name 1000baseT is starting to get silly) is starting to be deployed, too. 10base2 wire is usually black coaxial cable, with twist-on T-pieces to connect them to things: everyone gets connected in a big line, with special 'terminator' pieces on the two ends. UTP is usually blue wire, with clear 'click-in' phone-style connectors which plug into sockets to connect: each wire connects one node to a central 'hub'. The cable is a couple of dollars a meter, and the 10baseT/10base2 cards (many cards have plugs for both) are hard to get brand new. 100baseT cards, which can also speak 10baseT as well, are ten times faster, and about $30.

On the other end of the spectrum is Fiber; a continuous tiny glass filament wrapped in protective coating which can be used to run between continents. Generally, fiber costs thousands.

We usually call each connection to a node a 'network interface', or 'interface' for short. Linux gives these names like 'eth0' for the first ethernet interface, and 'fddi0' for the first fiber interface. The '/sbin/ifconfig' command lists them.

Protocol: What It's Speaking

The final thing to care about is the language the two are speaking. When two 2 (modems) are talking to each other down a phone line, they need to agree what the different sounds mean, otherwise it simply won't work. This convention is called a 'protocol'. As people discovered new ways of encoding what the computer says into smaller sounds, new protocols were invented; there are at least a dozen different modem protocols, and most modems will try a number of them until they find one the other end understands.

Another example is the 2 (100baseT) network mentioned above: it uses the same physical 2 (network links) (2 (UTP)) as 2 (10baseT) above, but talks ten times as fast.

These two protocols are what are called 'link-level' protocols; how stuff is handed over the individual network links, or 'one hop'. The word 'protocol' also refers to other conventions which are followed, as we will see next.

3 What is the 'Internet'?

The Internet is a 2 (WAN) which spans the entire globe: it is the largest computer network in existence. The phrase 'internetworking' refers to connecting separate networks to build a larger one, hence 'The Internet' is the connection of a whole pile of subnetworks.

So now we look at the list above and ask ourselves: what is the Internet's size, physical details and protocols?

The size is already established above: it's global.

The physical details are varied however: each little sub-network is connected differently, with a different layout and physical nature. Attempts to map it in a useful way have generally met with abject failure.

The protocols spoken by each link are also often different: all of the 2 (link-level protocols) listed above are used, and many more.

3.1 How Does The Internet Work?

The question then arises: how come every node on the Internet can talk to the others, if they all use different link-level protocols to talk to each other?

The answer is fairly simple: we need another protocol which controls how stuff flows through the network. The link-level protocol describes how to get from one node to another if they're connected directly: the 'network protocol' tells us how to get from one point in the network to any other, going through other links if necessary.

For the Internet, the network protocol is the Internet Protocol (version 4), or 'IP'. It's not the only protocol out there (Apple's AppleTalk, Novell's IPX, Digital's DECNet and Microsoft's NetBEUI being others) but it's the most widely adopted. There's a newer version of IP called IPv6, but it's still not common.

So to send a message from one side of the globe to another, your computer writes a bit of Internet Protocol, sends it to your modem, which uses some modem link-level protocol to send it to the modem it's dialed up to, which is probably plugged into a terminal server (basically a big box of modems), which sends it to a node inside the ISP's network, which sends it out usually to a bigger node, which sends it to the next node... and so on. A node which connects two or more networks is called a 'router': it will have one 2 (interface) for each network.

We call this array of protocols a 'protocol stack', usually drawn like so:

```
[ Application: Handles Porn ]          [ Application Layer: Serves Porn ]
```

```
              |                                          ^
              v                                          |
[ TCP: Handles Retransmission ]          [ TCP: Handles Retransmission ]
              |                                          ^
              v                                          |
    [ IP: Handles Routing ]                    [ IP: Handles Routing ]
              |                                          ^
              v                                          |
[ Link: Handles A Single Hop ]           [ Link: Handles A Single Hop ]
              |                                          |
              +------------------------------------------+
```

So in the diagram, we see Netscape (the Application on top left) retrieving a web page from a web server (the Application on top right). To do this it will use 'Transmission Control Protocol' or 'TCP': over 90% of the Internet traffic today is TCP, as it is used for Web and EMail.

So Netscape makes the request for a TCP connection to the remote web server: this is handed to the TCP layer, which hands it to the IP layer, which figures out which direction it has to go in, hands it onto the appropriate link layer, which transmits it to the other end of the link.

At the other end, the link layer hands it up to the IP layer, which sees it is destined for this host (if not, it might hand it down to a different link layer to go out to the next node), hands it up to the TCP layer, which hands it to the server.

So we have the following breakdown:

1. The application (Netscape, or the web server at the other end) decides who it wants to talk to, and what it wants to send).

2. The TCP layer sends special packets to start the conversation with the other end, and then packs the data into a TCP 'packet': a packet is just a term for a chunk of data which passes through a network. The TCP layer hands this packet to the IP layer: it then keeps sending it to the IP layer until the TCP layer at the other end replies to say that it has received it. This is called 'retransmission', and has a whole heap of complex rules which control when to retransmit, how long to wait, etc. It also gives each packet a set of numbers, which mean that the other end can sort them into the right order.

3. The IP layer looks at the destination of the packet, and figures out the next node to send the packet to. This simple act is called 'routing', and ranges from really simple (if you only have one modem, and no other network interfaces, all packets should go out that interface) to extremely complex (if you have 15 major networks connected directly to you).

4 This IP Thing

So the role of the IP layer is to figure out how to 'route' packets to their final destination. To make this possible, every interface on the network needs an 'IP address'. An IP address consists of four numbers separated by periods, like '167.216.245.249'. Each number is between zero and 255.

Interfaces in the same network tend to have neighboring IP addresses. For example, '167.216.245.250' sits right next to the machine with the IP address '167.216.245.249'. Remember also that a router is a node with interfaces on more than one network, so the router will have one IP address for each interface.

So the Linux Kernel's IP layer keeps a table of different 'routes', describing how to get to various groups of IP addresses. The simplest of these is called a 'default route': if the IP layer doesn't know better, this is where it will send a packet onwards to. You can see a list of routes using '/sbin/route'.

Routes can either point to a link, or a particular node which is connected to another network. For example, when you dial up to the ISP, your default route will point to the modem link, because that's where the entire world is.

```
Rusty's              ISP's  ~~~~~~~~~~~~
 Modem               Modem {            }
    o------------------o { The Internet }
                           {            }
                            ~~~~~~~~~~~~
```

But if you have a permanent machine on your network which connects to the outside world, it's a bit more complicated. In the diagram below, my machine can talk directly to Tridge and Paul's machines, and to the firewall, but it needs to know that packets heading the rest of the world need to go to the firewall, which will pass them on. This means that you have two routes: one which says 'if it's on my network, just send it straight there' and then a default route which says 'otherwise, send it to the firewall'.

```
                       o  Tridge's
                       |    Work Machine      ~~~~~~~~~~~~
Rusty's                |                     {            }
 Work Machine o--------+-----------------o--{ The Internet }
                       |            Firewall {            }
                       |                      ~~~~~~~~~~~~
                       o  Paul's
                            Work Machine
```

4.1 Groups of IP Addresses: Network Masks

There is one last detail: there is a standard notation for groups of IP addresses, sometimes called a 'network address'. Just like a phone number can be broken up into an area prefix and the rest, we can divide an IP address into a network prefix and the rest.

It used to be that people would talk about 'the 1.2.3 network', meaning all 256 addresses from 1.2.3.0 to 1.2.3.255. Or if that wasn't a big enough network, they might talk about the '1.2 network' which meant all addresses from 1.2.0.0 to 1.2.255.255.

We usually don't write '1.2.0.0 - 1.2.255.255'. Instead, we shorten it to '1.2.0.0/16'. This weird '/16' notation (it's called a 'netmask') requires a little explanation.

Each number between the dots in an IP address is actually 8 binary digits (00000000 to 11111111): we write them in decimal form to make it more readable for humans. The '/16' means that the first 16 binary digits is the network address, in other words, the '1.2.' part is the the network (remember: each digit represents 8 binary digits). This means any IP address beginning with '1.2.' is part of the network: '1.2.3.4' and '1.2.3.50' are, and '1.3.1.1' is not.

To make life easier, we usually use networks ending in '/8', '/16' and '/24'. For example, '10.0.0.0/8' is a big network containing any address from 10.0.0.0 to 10.255.255.255 (over 16 million addresses!). 10.0.0.0/16 is smaller, containing only IP addresses from 10.0.0.0 to 10.0.255.255. 10.0.0.0/24 is smaller still, containing addresses 10.0.0.0 to 10.0.0.255.

To make things confusing, there is another way of writing netmasks. We can write them like IP addresses:

```
10.0.0.0/255.0.0.0
```

Finally, it's worth noting that the very highest IP address in any network is reserved as the 'broadcast address', which can be used to send a message to everyone on the network at once.

Here is a table of network masks:

Short Form	Full Form	Maximum #Machines	Comment
/8	/255.0.0.0	16,777,215	Used to be called an 'A-class'
/16	/255.255.0.0	65,535	Used to be called an 'B-class'
/17	/255.255.128.0	32,767	
/18	/255.255.192.0	16,383	
/19	/255.255.224.0	8,191	
/20	/255.255.240.0	4,095	
/21	/255.255.248.0	2,047	
/22	/255.255.252.0	1,023	
/23	/255.255.254.0	511	
/24	/255.255.255.0	255	Used to be called a 'C-class'
/25	/255.255.255.128	127	
/26	/255.255.255.192	63	
/27	/255.255.255.224	31	
/28	/255.255.255.240	15	
/29	/255.255.255.248	7	
/30	/255.255.255.252	3	

5 Machine Names and IP Addresses

So every interface on every node has an IP address. It was realized quite quickly that humans are pretty bad at remembering numbers, so it was decided (just like phone numbers) to have a directory of names. But since we're using computers anyway, it's nicer to have the computer look up the names for us automatically.

Hence we have the Domain Name System (DNS). There are nodes with well known IP addresses which programs can ask to look up names, and return IP addresses. Almost all programs you will use are capable of doing this, which is why you can put 'www.linuxcare.com' into Netscape, instead of '167.216.245.249'.

Of course, you need the IP address of at least one of these 'name servers': usually these are kept in the '/etc/resolv.conf' file.

Since DNS queries and responses are fairly small (1 packet each), the TCP protocol is not usually used: it provides automatic retransmission, ordering and general reliability, but at a cost of sending extra packets through the network. Instead we use the very simple 'User Datagram Protocol', which doesn't offer any of the fancy TCP features we don't need.

6 Different Services: Email, Web, FTP, Name Serving

In the earlier example, we showed Netscape sending a TCP request to a web server running on another node. But imagine that the node with the web server is also running an Email server, an FTP server and a name server: how does it know which server the TCP connection is for?

This is where TCP and UDP have a concept of 'ports'. Every packet has space for a 'destination port', which says what service the packet is for. For example, TCP port 25 is the mail server, and TCP port 80 is the web server (although sometimes you find web servers on different ports). A list of ports can be found in '/etc/services'.

Also, if two Netscape windows are both accessing different parts of the same web site, how does the Linux box running Netscape sort out the TCP packets coming back from the web server?

This is where the 'source port' comes in: every new TCP connection gets a different source port, so everyone can tell them apart, even if they are going to the same destination IP address and the same destination port. Usually the first source port given will be 1024, and will increase over time.

7 Dialup Interfaces: PPP

When you dial your modem to an ISP, and it connects to their modem, the kernel doesn't just shove IP packets through it. There is a protocol called 'Point-to-Point Protocol', or 'PPP', which is used to negotiate with the other end before any packets are allowed through. This is used by the ISP to identify who is dialed up: on your Linux box, a program called the 'PPP daemon' handles your end of the negotiation.

Because there are so many dialup users in the world, they usually don't have their own IP address: most ISPs will assign you one of theirs temporarily when you dial up (the PPP daemon will negotiate this). This is often called a 'dynamic IP address', as separate from a 'static IP address' which is the normal case where you have your own address permanently. Usually they are assigned by modem: the next time you dial up, you will probably get a different modem in the modem pool, and hence a different IP address.

8 What Packets Look Like

For the exceptionally curious (and the curiously exceptional), here is a description of what a packet actually looks like. There are several tools which watch what packets are passing in and out of your Linux box: the most common one is 'tcpdump' (which understands more than TCP these days), but a nicer one is 'ethereal'. Such programs are known as 'packet sniffers'.

The start of each packet says where it's going, where it came from, the type of the packet, and other administrative details. This part is called the 'packet header'. The rest of the packet, containing the actual data being transmitted, is usually called the 'packet body'.

So any IP packet begins with an 'IP header': at least 20 bytes long. It looks like (this diagram stolen shamelessly from RFC 791):

```
.-------+-------+---------------+-------------------------------.
|Version|  IHL  |Type of Service|          Total Length         |
|-------+-------+---------------+-------------------------------|
|         Identification        |Flags|      Fragment Offset    |
```

```
|---------------+---------------+-------------------------------|
|  Time to Live |    Protocol   |         Header Checksum       |
|---------------+---------------+-------------------------------|
|                         Source Address                        |
|---------------------------------------------------------------|
|                      Destination Address                      |
`---------------------------------------------------------------'
```

The important fields are the Protocol, which indicates whether this is a TCP packet (number 6), a UDP packet (number 17) or something else, the Source IP Address, and the Destination IP Address.

Now, if the protocol fields says this is a TCP packet, then a TCP header will immediately follow this IP header: the TCP header is also at least 20 bytes long:

```
.-------------------------------+-------------------------------.
|          Source Port          |       Destination Port        |
|-------------------------------+-------------------------------|
|                        Sequence Number                        |
|---------------------------------------------------------------|
|                    Acknowledgment Number                      |
|-------------------+-+-+-+-+-+-+-------------------------------|
|  Data |           |U|A|P|R|S|F|                               |
| Offset| Reserved  |R|C|S|S|Y|I|            Window             |
|       |           |G|K|H|T|N|N|                               |
|-------+-----------+-+-+-+-+-+-+-------------------------------|
|           Checksum            |         Urgent Pointer        |
`---------------------------------------------------------------'
```

The most important fields here are the source port, and destination port, which says which service the packet is going to (or coming from, in the case of reply packets). The sequence and acknowledgement numbers are used to keep packets in order, and tell the other end what packets have been received. The ACK, SYN, RST and FIN flags (written downwards) are single bits which are used to negotiate the opening (SYN) and closing (RST or FIN) of connections.

Following this header comes the actual message which the application sent (the packet body). A normal packet is up to 1500 bytes: this means that the most space the data can take up is 1460 bytes (20 bytes for the IP header, and 20 for the TCP header): over 97%.

9 Summary

So the modern Internet uses IP packets to communicate, and most of these IP packets use TCP inside. Special nodes called ‘routers’ connect all the little networks together into larger networks, and pass these packets through to their destination. Most normal machines are only attached to one network (ie. have only one interface), and so are not routers.

Every interface has a unique IP address, which look like ‘1.2.3.4’: interfaces in the same network will have related IP addresses, with the same start, the same way that phone connections in the same area have the same prefix. These network addresses look like IP addresses, with a ‘/’ to say how much of them is the prefix, eg ‘1.2.0.0/16’ means the first two digits is the network address: each digit represents 8 bits.

Machines are given names by the Domain Name Service: programs ask name servers to give them the IP address, given a name like 'www.linuxcare.com'. This IP address is then used as the destination IP address to talk to that node.

Rusty is really bad at writing documentation, especially for beginners.

Enjoy!

Rusty.

10 Thanks

Thanks to Alison, for sitting through the original terrible draft, and telling me how shit it was, in the nicest possible way.

11 Index