GNU/Linux Networking Tutorial for Newbies

M K Saravanan
Centre for Internet Research
mksarav@comp.nus.edu.sg

February 21, 2002

Contents

1 License **2**

2 Introduction **2**

3 Further Information **2**

4 Networking fundamentals **2**
4.1 Protocols . . . 2
4.2 Origin of Internet . . . 3
4.3 Birth of TCP/IP . . . 4
4.4 Packet Switched Network . . . 4
4.5 IP Addressing . . . 5
4.5.1 More about IP Addresses . . . 6
4.6 Domain Name System (DNS) . . . 7
4.7 TCP/IP Layers . . . 9
4.8 Network Interface Card (NIC) . . . 9
4.9 Routing . . . 11
4.10 Relation between IP Address and MAC address . . . 11
4.11 Data Flow from End-to-End . . . 11
4.12 Note on RFCs/STDs/FYIs/BCPs . . . 12

5 Setting up a small network **12**
5.1 DNS Settings . . . 15
5.2 Installing Client/Server programs . . . 16
5.3 Internet Services . . . 18
5.4 Telnet . . . 19
5.5 ssh . . . 20
5.6 File Transfer Protocol: ftp . . . 20
5.7 Web service . . . 22
5.8 Samba . . . 23

6 Network Troubleshooting Utilities **29**
6.1 ping 29
6.2 tcpdump 29
6.3 ipgrab 30
6.4 netstat 32
6.5 traceroute 32
6.6 tracepath 33

7 Before THE END **34**

1 License

2 Introduction

This tutorial assumes that you already know the basics of GNU/Linux. It won't teach you the basic things like editing a file, cp, rm, etc... Throughout the tutorial we will assume Redhat 7.2 distribution though the commands are similar in other distribution. This tutorial will teach you only the very minimum basics of networking. I will try to update this guide whenever i get free time. You can download the latest copy of this tutorial at `http://mksarav.tripod.com`.

3 Further Information

A large number of tutorials are available for GNU/Linux on all possible topics. Kindly do a GNU/Linux google search at `http://www.google.com/linux`. To start with you can go through the following HOWTOs:

- LINUX: Rute User's Tutorial and Exposition by `Paul Sheer`
  `http://rute.sourceforge.net`
- `DOS-Win-to-Linux-HOWTO`
- `Unix-and-Internet-Fundamentals-HOWTO`
- `RFC#1180 - A TCP/IP Tutorial`
- `Networking-Overview-HOWTO`

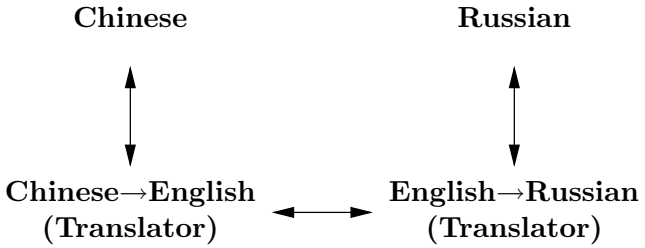
You can download all the HOWTOs and mini-HOWTOs from `http://www.linuxdoc.org`. Most of the time the answer for the question you are asking will be available by doing a google search at `http://groups.google.com`. Please don't expect spoon-feeding. Be prepared to go through man pages first whenever you are in doubt.

4 Networking fundamentals

4.1 Protocols

If you want to share the resources available in various computers (hardware as well as software), then through some means we can connect all of them and make use of the

available resources with each other. We call the interconnection of all these computers as a Network. To connect two computer with each other, first they should understand what the other computer is saying. Say, if you talk in chinese and the opposite guy in Russian, then both of you won't understand what each other is saying. To solve this problem say both of you are following a rule: each should employ a translator whose common language is english. Then you can establish communication with each other like:



Since you followed a rule here, both of you are now able to communicate. Here the rule is: Employ a translator whose common language is english. In networking terminology, a set of rules is often called a PROTOCOL. Thus if both of you follow a common protocol, you can establish communication with each other. This is what happening in a computer network. Thus to establish communication with two computers, both of them must follow a common protocol.

4.2 Origin of Internet

GNU/Linux supports a wide variety of networking protocols. However, TCP/IP is the de-facto standard protocol for networking. Before actually discussing about TCP/IP, let us go through some brief history about how this wonderful protocol came into existence.

On 4th October 1957, the world first satellite SPUTNIK was launched by USSR. This created added tension to the cold war between USSR & US at that time. USDoD (United States Department of Defence) decided to increase the research activities in space programme, and also to integrate the resources and sharing of valuable data spreaded across various research labs. To increase sharing of resources, USDoD started a project called ARPANET (Advanced Research Project Agency NETwork). The then US president "Dwight D Eisenhower" saw the need for the ARPA. In 1962, Dr.J.C.R. Licklider was chosen to head ARPA's research in improving the military's use of computer technology.

The main goals of ARPANET were:

- To increase the sharing of resources, ARPA began coordinating the developments of a vendor-independent network to tie the major research sites together. The need for a vendor-independent network was the first priority, since each facility used different computers with proprietary networking technology. In 1968, work began on a private packet-switched network, which eventually became known as ARPANET (Mother of Internet).

- During those days, Centralised Circuit-Switched network, involving dedicated end-to-end connections between two specific sites, were used. In contrast, the ARPANET

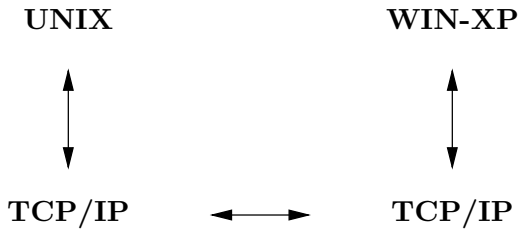
tried to build a De-Centralised network, allowing organisations to interconnect into a mesh-like topology. This way the data can be sent from one computer to another using a variety of different routes. Such a network will survive war-time attack ensuring high reliability due to its decentralised nature.

4.3 Birth of TCP/IP

In order to create a vendor-independent network, i.e. to establish communications between different hardware platforms, we need to follow a certain set of procedures. ARPANET team came up with a new protocol called TCP/IP (Transmission Control Protocol/Internet Protocol).

TCP/IP Protocol Suite is a collection of various protocols like TCP, IP, ARP, RARP, ICMP, etc... Since TCP & IP forms the core of these protocols, the entire protocol suite itself named as TCP/IP. So, whenever you see the word TCP/IP, remember it is a set of protocols.

Once TCP/IP is installed in each computer, we can easily create a vendor-independent network.



Slowly TCP/IP became the de-facto standard for Internet Communications. Nowadays almost all the Intranet also started using TCP/IP. But remember TCP/IP is not the only protocol available for networking. Several other protocols like IPX/SPX, AppleTalk Protocol, etc... are also in use.

If you install TCP/IP in all the computers then you can establish communication with each other by properly configuring each of them. GNU/Linux comes with inbuilt TCP/IP support in the kernel. So you don't have to do anything special to install TCP/IP.

4.4 Packet Switched Network

To establish communication between two computers, traditionally circuit-swtiched mechanism were used which involves dedicated end-to-end connection between two sites. To overcome the problems of Circuit-Switching (war-time attack: no inbuilt fault-tolerance in centralised network), Packet Switching Mechanism was formulated by ARPANET which ensures fault-tolerance due to its decentralised nature.

The data to be sent, will be divided into small packets, each consisting of a HEADER and a FOOTER information. For. e.g. a 10KB data can be broken into ten 1KB packets.

The HEADER will contain source computer (IP) Address, Destination Computer (IP) Address, CRC (Cyclic Redundancy Checksum), TTL (Time To Live), etc...

4.5 IP Addressing

To uniquely identify a computer in a network, a new addressing scheme was introduced called "IP Addressing". According to this, each computer will be identified by a unique IP address made up of 32 bits. For simplicity, it is usually written in "Dotted Decimal Notation" where each octet in the IP address will be written in decimal form. For e.g.

```
11001010 00110110 00000110 00010100  is written as 202.54.6.20
```

IP Addresses are classified as follows:

```
                        IP Address Classification (IPv4)


Class-A:  0xxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx
          0####### %%%%%%%% %%%%%%%% %%%%%%%%


Class-B:  10xxxxxx xxxxxxxx xxxxxxxx xxxxxxxx
          10###### ######## %%%%%%%% %%%%%%%%


Class-C:  110xxxxx xxxxxxxx xxxxxxxx xxxxxxxx
          110##### ######## ######## %%%%%%%%


Class-D:  1110xxxx xxxxxxxx xxxxxxxx xxxxxxxx


Class-E:  11110xxx xxxxxxxx xxxxxxxx xxxxxxxx

x - can be either 0 or 1
# - constitute Net Id.
% - constitute Host Id.
```

The different classes of IP addresses are discriminated by their MSB bits. Out of these, Class-E is reserved for future use and Class-D is meant for "Multicast Group Ids". If we send a packet to all the host in a network, then we will call it as broadcast. Instead if we send only to a group of host within a network, then it is called multicast.

Each of the Class-A, B, & C addresses are further logically divided into two parts namely, "Net id." and "Host id.". For e.g. consider,

```
202.54.6.20 - 11001010 00110110 00000110 00010100
```

since the address starts with MSB 110, it belongs to Class-C. In Class-C the first three octet with MSBs 110 will form the network id. and the last octet is the host id. Therefore the net id is: 202.54.6 and 20 is the host id.

4.5.1 More about IP Addresses

IP Addresses are issued by InterNIC (Inter Network Information Centre), a US based Non-Profitable Organisation. Nowadays, countrywise agencies are also coordinating with InterNIC to get you the IP addresses.

The current IP Addressing that we are following is called IPv4 (IP Version 4) which uses 32 bit IP addresses. At first instance, you may think that we are having a large IP addr. space.

2^{32} = 4294967296 IP addresses. But in reality, due to the nature of IP Address classification we can only use the following IP addresses:

```
                                IPv4 Address Space

Class-A:  0.0.0.0     to   127.255.255.255
Class-B:  128.0.0.0   to   191.255.255.255
Class-C:  192.0.0.0   to   223.255.255.255
Class-D:  224.0.0.0   to   239.255.255.255
Class-E:  240.0.0.0   to   247.255.255.255
```

Thus, we are actually having $< 2^{32}$ IP Addr. Further Class-E addr. are reserved for future use and Class-D is meant for "Multicast Group Ids.". Therefore for any practical usage we are left with Class-A, B and C IP Address space. After the introduction of World Wide Web (WWW) in 1990's by Tim Berners Lee, the growth rate of number of hosts on Internet started becoming exponential. Unfortunately the IPv4 address classification are not planned appropriately to tackle the enormous growth rate of various capacity networks on Internet.

To understand the Net id., consider a University Network. If it want to put a server in each dept. and connect all the departments with Internet, it may easily need 150 to 200 IP addresses approximately. Instead of getting individual IP addr. eachtime we introduce a new server, it is good to get a Class-C (the nearest match) Net Id. like 202.54.7.0. For a given Class-C net id. you can connect a maximum of 254 hosts. For e.g. let us take 202.54.7.0 - Class-C Network Id. (imaginary) representing the Univ. N/W. To this network, now we can connect a max.of 254 hosts with IP addr. from:

```
202.54.7.1  to 202.54.7.254
```

All one's (binary) in the host portion of a Net Id. is a special IP address called "Net Directed BROADCAST IP Address". Thus in our case, 202.54.7.255 is the Broadcast IP Addr. for the Univ. N/W. Any packet with this IP addr. as destination will reach all the hosts in the Univ. N/W (Hence the name Broadcast).

Given an IP address, to extract the network id. from it, "netmask" are used. For e.g. to extract the net id. portion from a Class C address, say 202.54.6.20, we need to do a binary AND operation with the first 24 most significant bits. Remember in Class-C, the first 24 MSBs will form the network id. Let us work out an example.

```
                                Private Network Ids.

Class-A:  (1 network id)      10.x.x.x
Class-B:  (16 network ids)    172.16.x.x  to 172.31.x.x
Class-C:  (256 network ids)   192.168.0.x to 192.168.255.x


                          Example:  Calculating the netmask


Class-C IP address:  202.54.6.20    -->  11001010 00110110 00000110 00010100
Netmask for Class-C: 255.255.255.0  -->  11111111 11111111 11111111 00000000
                                         -----------------------------------
AND operation result: 202.54.6.0    -->  11001010 00110110 00000110 00000000
                                         -----------------------------------
```

To obtain the netmask for a given IP address, count the number of bits which forms the network id.. Then netmask can be obtained by writing that many number of binary ones starting from the first MSB and leaving the remaining bits as binary zeroes.

Due to the exponential growth of Internet, IP addresses started exhausting drastically. Within few years from now, we don't have any more addresses available in IPv4. To solve this problem, the Internet Engineering Task Force (IETF) came up with a new version of Internet Protocol called IPv6. IPv6 addresses are 128 bits wide and won't be exhausted in the foreseeable future. The transition to IPv6 will take considerable amount of time and will happen slowly. Several mechanism are being proposed for IPv4 & IPv6 interoperability till the whole world changes to IPv6. In the mean time, to prevent the quick exhaustion of IPv4 addresses, IP subnetting and CIDR (ClassLess Inter Domain Routing) techniques are introduced.

IP addresses are globally unique. That means, no two host on the Internet will have the same IP address. But in real life, we may need to have private networks within the organisation. Depending upon the organisation policy, these private network may or may not be merged with the Internet in the future. In IPv4, it is not possible to assign a unique IP address to each and every host on the Planet. Thus the following network ids. are reserved for private networking purpose.

Anybody can use the above addresses, to setup their own private networks. On Internet you will never find these IP addresses; if you – then your router is not configured properly.

4.6 Domain Name System (DNS)

To easily remember the IP addresses, a new scheme called "Domain Name System" (DNS in short) was introduced. According to DNS, each IP address will be assigned an unique name called Domain Name. For e.g., the domain name of 202.54.6.20 is:

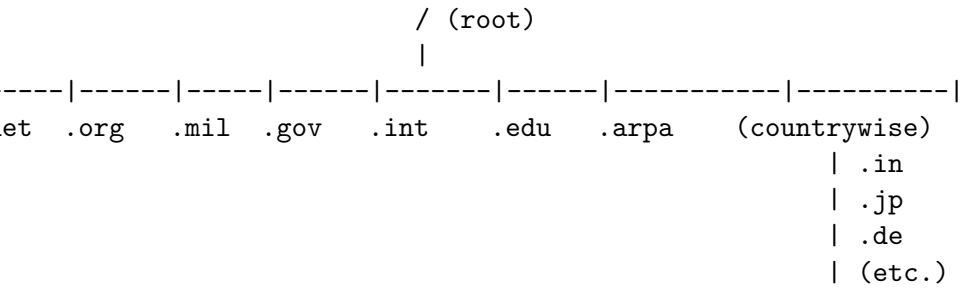
```
IP Addr: 202.54.6.20 <==> md2.vsnl.net.in  (Domain Name)
```

IP Addr. follows Left to Right hierarchy whereas the DN follows Right to Left hierarchy. The Domain Names can be made up of two or more words separated by dot. e.g. domain names are:

```
www.mitindia.edu
www.comp.nus.edu.sg
ftp.gnu.org
chn.vsnl.net.in
```

For Administrative and maintenance purposes, a set of Top Level Domains (TLD) were introduced.

DNS - Top Level Domains (TLD)




```
Organisational Classification
=============================

.com  - commercial domain
.net  - networking companies
.org  - non-profitable organisations
.mil  - military (US only)
.gov  - government (US only)
.int  - international organisations
.edu  - educational institutions
.arpa - a special domain (more about this later)

(More domains may be added in future like .store, .per etc..,)

Countrywise Classification
==========================

.in  - India
.jp  - Japan
.de  - Germany
```

```
.ru  - Russia
.sg  - Singapore
.my  - Malaysia
etc..,
```

It is the job of the Network Administrator to maintain a server called DNS server in each network. The DNS server will maintain a table of Domain Name & IP Address Pairs of all the hosts in that network. Normally the ISPs (Internet Service Provider) will maintain atleast two DNS servers namely Primary and Secondary (for backup purpose). DNS as a whole is a distributed database. No single host on the Internet will maintain the DN, IP Address pairs of all hosts exists on the Internet.

4.7 TCP/IP Layers

To standardise the networking protocols, the ISO (International Standardisation Organisation) cameup with a generic protocol model called "OSI Reference Model". OSI stands for Open System Interconnection. In a typical OSI reference Model, a protocol can be represented by 7 layers whereas in TCP/IP actually there are only 4 layers. Remember TCP/IP came into existence much before OSI Model and hence became a defacto standard for Internetworking. In TCP/IP Application Layer take care of the job of Presentation & Session Layer if needed & Link Layer partly take care of the job of Physical Layer.

Note: Some authors may logically represent the Physical Layer function as the fifth Layer in TCP/IP Model.

```
=========================================================================
7 Layer - OSI Model                    4 Layer - TCP/IP Model
=========================================================================
(7) * Application Layer                 * Application Layer
(6) * Presentation Layer                        "
(5) * Session Layer                             "
(4) * Transport Layer                   * Transport Layer
(3) * Network Layer                     * Network Layer
(2) * Data Link Layer                   * Link Layer
(1) * Physical Layer                            "
=========================================================================
```

The logical structure of the TCP/IP layers can be represented as shown in the figure. This figure was taken from RFC#1180 and slightly modified to include ICMP and IGMP.

4.8 Network Interface Card (NIC)

To connect each host to the network, we need a network interface card (NIC). Depending on the technology used at Data Link layer like Ethernet, IBM Token Ring etc... We can have a variety of network interface cards. However ethernet technology is most popular

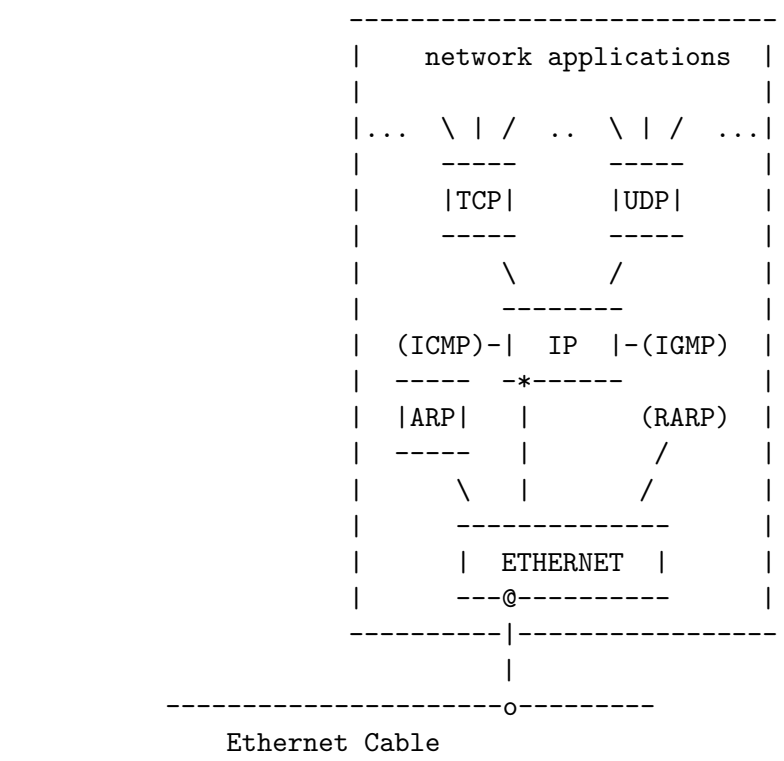


Figure: Basic TCP/IP Network Node

ACRONYMS USED:
==============

ftp: file transfer protocol
SNMP: Simple Network Management Protocol
DNS: Domain Name System
TCP: Transmission Control Protocol
UDP: User Datagram Protocol
ICMP: Internet Control Message Protocol
IGMP: Internet Group Management Protocol
ARP: Address Resolution Protocol
RARP: Reverse Address Resolution Protocol

Examples:

* Application Layer: ftp, SNMP, telnet, DNS, etc..,
* Transport Layer: TCP, UDP
* Network Layer: IP (ICMP, IGMP)
* Link: Ethernet (ARP, RARP)

and is used by the vast majority of LANs. Thus the NIC that uses Ethernet is commonly called as Ethernet Card. Each ethernet card comes with its own address called "Hardware Address or MAC Address" which consists of 6 bytes (48 bits) represented in Hexa Colon Notation. Here is an e.g. of a MAC address:

```
00:E0:00:5A:D1:2A
```

4.9 Routing

If you want to connect your host to more than one network – say, three network, then you will need three ethernet card to connect to the three networks. Thus, a host may contain more than one ethernet card. Such a host is called a "multi-homed" host. Sometimes, you want to pass data from one network to another network. For such a situation, you can use a multi-homed host by configuring the host to forward data between a pair of ethernet cards. Such a host is called a Router. Thus all routers are essentially a multihomed host but the reverse may not be true.

In a network, to reach a particular destination, there may be more than one path. To find out the optimum path, routers may use "Routing Algorithms". In a small network where the network topology rarely change, we can use fixed (static) routes. But for a large network, we need to use dynamic routing algorithms to update the routing table as soon as the route changes are detected.

4.10 Relation between IP Address and MAC address

Infact the IP address is the logical address assigned to the NIC, since TCP/IP can identify a NIC, only by an IP address. But to transfer the data, NIC use only the hardware address. So a translation mechanism called ARP was invented to translate the IP address into the coressponding MAC address. Sometime, given a MAC address we need to know the corresponding IP address. For this another protocol called RARP was introduced.

4.11 Data Flow from End-to-End

Assume that you are sending an email. Email uses a protocol called SMTP (Simple Mail Transfer Protocol) to deliver the email to the remote host. SMTP comes under Application Layer. The data will be passed to TCP (Transport Layer). TCP will split the data into small units. It will add TCP header information to each unit. Now this unit of data along with TCP Header is called a TCP-Segment. TCP sends it to IP (Network Layer). IP will add its own header information to the TCP-Segment. The unit of TCP-Segment along with the IP Header is called an IP-Datagram. IP sends the Datagram to Ethernet (Link Layer). Ethernet will add its own header information to the IP-Datagram. The unit of IP-Datagram along with the Ethernet Header is called an Ethernet-Frame.

Now, TCP may receive data from one or more applications. To identify each one of them uniquely it will use a number called PORT number (16-bit). Using 16-bit we can have port no. from 0 to 65535. Ports are classified as well-known ports (used by servers - ranging from 0 to 1023) and ephemeral ports (used by clients - greater than 1023). The

assignment of port numbers is handled by IANA (Internet Assigned Numbers Authority). A single server may provide more than one services each distinguished by the port number. For e.g. a server can provide ftp (Port#21), http (Port#80), telnet (Port#23) all at the same time. Thus a TCP Connection is identified by 4 parameters:

```
{Source IP Address, Source Port Number, Destn. IP Address, Destn. Port Number}
```

(IP Address, Port Number) pair is also called as SOCKET. Thus each TCP connection can be uniquely identified by two sockets: source socket and destination socket. Remember a SOCKET identifies one end of a TCP connection.

IP may receive data from one or more sources like TCP, UDP, ICMP, IGMP etc... To distinguish each one of them it uses a unique number called protocol number [8-bit protocol field in the IP Header]. In Unix machine you can find the protocol numbers in `/etc/protocols` file. For e.g. the value of protocol number for TCP is 6 and that of UDP is 17.

Ethernet may receive data from one or more sources like IP, ARP, RARP etc... To distinguish each one of them it uses a unique number called frame type [16-bit frame type field in Ethernet Header].

Thus, at the remote end, the Ethernet receive the frame, check its frame type and demultiplex the datagram to either IP/ARP/RARP. If IP receives the datagram, based on the 8-bit protocol field, it demultiplex the data to either TCP/UDP/ICMP/IGMP. If TCP receives the data, based on the 16-bit port number, it demultiplex the data to the corresponding application.

ICMP, IGMP make use of IP Datagram Delivery Service & ARP, RARP make use of Ethernet for sending ARP/RARP packets. Thus IP may receive data from TCP/UDP/ICMP/IGMP (other sources are also possible, but that is not important here).

4.12 Note on RFCs/STDs/FYIs/BCPs

The official standards in the Internet commmunity are published as "Request for Comments" (RFC) documents. However not all RFCs are official standards and they are just available for informational purposes. The RFCs which are officially announced as standards are also called as STDs. As of this writing, more than 3200 RFCs are available. RFCs are numbered from 1 and you can download it from several mirror sites on Internet. One of them is `http://www.ietf.org/rfc.html`. Apart from RFCs, "For Your Information" and "Best Current Practices" documents are also very valuable. The details about emerging technologies, associated problems and experience are also published as "Internet Drafts". You can obtain all the RFCs/STDs/FYIs/BCPs and Internet Drafts from `http://sunsite.cnlab-switch.ch/ftp/doc/standard/`.

5 Setting up a small network

Understanding the theory of networking will be more perfect only if we do some experiment with the real networks. So let us create a mini private network with just two host.

GNU/Linux supports a wide variety of Ethernet Cards. If you have the Ethernet Card in your computer, most distribution will automatically detect it during installation and include the appropriate driver modules. We just need to configure the ethernet cards with appropriate IP addresses. The ethernet card will be automatically assigned the device name "eth0" by the kernel. If you have more than one ethernet card in a host, then the remaining will be assigned with the device name eth1, eth2, etc... respectively. Whether you have ethernet card or not, by default all the host will be configured for local loopback interface. There is no any real hardware associated with loopback device. It is just a software interface. Those who don't have real network interface card, can make use of this software loopback interface. The kernel will assign the device name "lo" for this interface. By default all the GNU/Linux distribution will configure the local loopback interface with the IP address 127.0.0.1. Theoretically you can use any of the 127.x.x.x. To view/modify the configuration of an interface we can use the "ifconfig" command. First do a "man ifconfig".

```
bash-2.05# ifconfig lo
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
```

We can find out whether a interface is alive or not by using the "ping" command. It will send ICMP ECHO_REQUEST message to the destination host. If the destination host is alive, it will respond with an ICMP ECHO_REPLY message. Upon receiving the response from the destination host, ping will print the round trip time.

```
bash-2.05# ping -c1 127.0.0.1
PING 127.0.0.1 (127.0.0.1) from 127.0.0.1 : 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=0 ttl=255 time=1.763 msec

--- 127.0.0.1 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max/mdev = 1.763/1.763/1.763/0.000 ms
```

The option -c1 tells the ping command to send just one ICMP ECHO_REQUEST message.

Let us give the name feynman[1] and ramanujan[2] to our host respectively. Using the "hostname" command, you can set/view the hostname.

[1] Feynman is a Nobel Laurette in Quantum Electrodynamics. Feynman lectures in Physics are world famous

[2] Ramanujan is a great Indian mathematician and considered as a genius in number theory.

```
bash-2.05# hostname feynman
bash-2.05# hostname
feynman
bash-2.05#
```

To distinguish both the host, let us change the command prompt to reflect the username and hostname. The command prompt is stored under the environment variable name PS1 ("man bash" for more details). Here \ u represent username, \ h represent hostname, \ W represent current working directory.

```
bash-2.05# export PS1="[\u@\h \W]# "
[root@feynman root]#
```

You execute similar commands in the other host with the hostname "ramanujan". None of the settings you did will exist once you reboot the computer. To make the configuration permanent you need to include the configuration commands in the system startup script. On Redhat 7.x machines, these scripts are located under the directory `/etc/rc.d/`. It may be under different directory in other distributions. At any given time, all UNIX systems will be in one of the several possible runlevels. Redhat 7.x systems uses the following set of runlevels:

```
0 - halt (Do NOT set initdefault to this)
1 - Single user mode
2 - Multiuser, without NFS (The same as 3, if you do not have networking)
3 - Full multiuser mode
4 - unused
5 - X11
6 - reboot (Do NOT set initdefault to this)
```

You can set the default runlevel in the `/etc/inittab` file. For e.g. the line:

```
id:3:initdefault:
```

will set the default runlevel as 3. During bootup, first the system initialisation will be done using the script in `/etc/rc.d/rc.sysinit`. Then depending upon the runlevel, several initialisation scripts will be started by `/etc/rc.d/rc` with runlevel as argument. You can change the runlevel at anytime using the `init <runlevel>`. If you hate doing everything on command line, you can use the "linuxconf" utility to do most of the settings without bothering about the various configuration filenames. linuxconf will need root previlege. Hackers will do everything on command line :-)

We can configure the ethernet cards using "ifconfig". To run the command you will need root previlege. So login as root (or use su -) and type the following command:

```
[root@feynman root]# ifconfig eth0 192.168.1.1 netmask 255.255.255.0
[root@feynman root]# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:E0:00:5A:D1:2A
```

```
inet addr:192.168.1.1  Bcast:192.168.1.255  Mask:255.255.255.0
UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
RX packets:0 errors:0 dropped:0 overruns:0 frame:0
TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
collisions:0
RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
```

The output shows the interface is UP and running. To shutdown a interface you can use "down" option (e.g. ifconfig eth0 down).

Similarly you can configure the host "ramanujan" with the IP address 192.168.1.2. Now we can try to ping ramanujan from feynman:

```
[root@feynman root]# ping 192.168.1.2
connect: Network is unreachable
```

What happenned? We have not yet connected both the hosts physically using a ethernet cable. You need a piece of CAT-5 UTP (UnTwisted Pair) Ethernet Cable for each host, crimped on the both ends with a RJ-45 connector. You need one more device called a HUB (or a switch) to complete the network. A HUB is nothing but a electronic repeater. It will simply repeat the incoming data from a port to all the other ports. You can find 8-port, 16-port, 24-port hubs in the computer store. Connect one end of the cable to the ethernet card and other end to one of the ports in the HUB. Repeat the same for the other host. Now that the circuit is complete, you can ping ramanujan/feynman from each other.

```
[root@feynman root]# ping -c1 192.168.1.2
PING 192.168.1.2 (192.168.1.2) from 192.168.1.2 : 56(84) bytes of data.
64 bytes from 192.168.1.2: icmp_seq=0 ttl=255 time=1.566 msec

--- 192.168.1.2 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max/mdev = 1.566/1.566/1.566/0.000 ms
```

Congrats! You have successfully setup a TCP/IP network in GNU/Linux.

5.1 DNS Settings

For small network with very few hosts, we don't need a separate DNS server. Instead we can use the `/etc/hosts` file to store the IP addr. and domain name pairs. During installation, by default the entry for localhost will be added. Remember localhost refers the loopback interface.

```
[root@feynman root]# cat /etc/hosts
# Do not remove the following line, or various programs
# that require network functionality will fail.
127.0.0.1 localhost
[root@feynman root]#
```

Now add the `IP address <TAB> hostname` for the hosts ramanujan and feynman in both the systems.

```
[root@feynman root]# cat /etc/hosts
# Do not remove the following line, or various programs
# that require network functionality will fail.
127.0.0.1 localhost
192.168.1.1 feynman
192.168.1.2 ramanujan
[root@feynman root]#
```

Also add the following line in the `/etc/nsswitch.conf` which tells the order the DNS resolver will try to resolve the hostname (Most probably this line will be already there by default).

```
hosts: files dns
```

The above line tells the DNS resolver, to look for the file `/etc/hosts` first and then the DNS server.

From now onwards, you can simply use the hostname instead of specifying the IP address in most of the commands. If you have Internet access through a appropriate gateway in your network, then you need to set the domain name server entries in the file `/etc/resolv.conf`. A sample `/etc/resolv.conf` file will look like:

```
[mksarav@hanuman mksarav]$ cat /etc/resolv.conf
domain comp.nus.edu.sg
nameserver 137.132.90.2
search comp.nus.edu.sg
```

In the above e.g. the line "nameserver 137.132.90.2" represent the first name server that the system will use for any DNS query. You can add optional second and third entries if you want. The line "domain comp.nus.edu.sg" tells you that the example host (hanuman) belongs to the "comp.nus.edu.sg" domain.

5.2 Installing Client/Server programs

One way of installing whatever client and server programs you want is just select "Everything" option during the GNU/Linux Installation. It may be the good way for a beginner, but ultimately it is always good to compile a software from the source code.

RedHat comes with a package installation and management system called "RedHat Package Manager (RPM)" where each package filename ends with a .rpm extension. To install a .rpm package:

```
rpm -ivh <packagename.rpm>
```

e.g.

```
rpm -ivh telnet-server-0.17-20.rpm
rpm -ivh telnet-0.17-20.rpm
```

To upgrade a package use the -U option like:

```
rpm -Uvh wu-ftpd-2.6.1-18.rpm
rpm -Uvh ftp-0.17-12.rpm
```

To get a list of all packages installed in your system use:

```
rpm -qa | less
```

To uninstall (erase) a package use the -e option like:

```
rpm -e anonftp-4.0-9
```

To compile a software from the source code (.tar.gz), mostly you will need to perform the following steps:

```
1. login as root (or su -)
2. cd /usr/local/src
3. tar -zxvf path/package.tar.gz
4. cd package
5. ./configure
6. make
7. make install
```

will install the binaries in `/usr/local/bin`.

For example, let us install the packet sniffing utility called "ipgrab" following the above method:

```
[mksarav@feynman mksarav]$ su -
Password:
[root@feynman root]# cd /usr/local/src
[root@feynman src]# tar -zxvf /home/mksarav/download/ipgrab-0.9.8.tar.gz
[root@feynman src]# cd ipgrab-0.9.8/
[root@feynman ipgrab-0.9.8]# ./configure
[root@feynman ipgrab-0.9.8]# make
[root@feynman ipgrab-0.9.8]# make install
```

Installing and upgrading software packages with out any conflict is an art. For a thorough understanding of various methods of software installation and other details kindly refer the following HOWTOs:

- Software-Building-HOWTO
- Software-Proj-Mgmt-HOWTO

- Software-Release-Practice-HOWTO
- RPM-for-Unix-HOWTO
- RPM-HOWTO

For the rest of the tutorial we will assume that all the necessary client and server programs are already installed in the system. We will only explain the basic configuration details.

5.3 Internet Services

A wide variety of Internet Services are available in GNU/Linux. Some of them are:

- Telnet (remote login)
- File transfer protocol (ftp)
- Secure Shell (ssh)
- Name Server (e.g. bind)
- Web Server (e.g. Apache)
- Mail Server (e.g. Sendmail, exim, qmail)
- SMB Protocol (e.g. Samba - to share files across WIN and GNU/Linux systems)
- finger
- discard
- echo

etc... To quickly go through the various possible services have a look at `/etc/services` file. However all of the services listed in the file may not be available in your system. The entries in `/etc/services` tell you the port number that the particular service will listen for any incoming connection and the possible protocols (tcp/udp) that can be used.

Most of the network server programs can be run in two modes either as a standalone program or invoked by the inetd daemon (inetd stands for Internet services daemon). A daemon is nothing but a process running in the background. In the standalone mode, the server will be always running in the background and ready to serve for any incoming connections. In case of inetd, whenever some incoming connections request for a particular service, it will be lauched by the inetd daemon. In most of the UNIX systems, the inetd configuration will be in the file `/etc/inetd.conf`. However in Redhat 7.x systems, it has been put in a separate directory `/etc/xinetd.d` (xinetd stands for Extended Internet services). Here is an excerpt from the xinetd man page:

... xinetd performs the same function as inetd: it starts programs that provide Internet services. Instead of having such servers started at system initialization time, and be dormant until a connection request arrives, xinetd is the only daemon process started and it listens

on all service ports for the services listed in its configuration file. When a request comes in, xinetd starts the appropriate server. Because of the way it operates, xinetd (as well as inetd) is also referred to as a super-server. The services listed in xinetd's configuration file can be separated into two groups. Services in the first group are called multi-threaded and they require the forking of a new server process for each new connection request. The new server then handles that connection. For such services, xinetd keeps listening for new requests so that it can spawn new servers. On the other hand, the second group includes services for which the service daemon is responsible for handling all new connection requests. Such services are called single-threaded and xinetd will stop handling new requests for them until the server dies. Services in this group are usually datagram-based. So far, the only reason for the existence of a super-server was to conserve system resources by avoiding to fork a lot of processes which might be dormant for most of their lifetime. While fulfilling this function, xinetd takes advantage of the idea of a super-server to provide features such as access control and logging. Furthermore, xinetd is not limited to services listed in `/etc/services`. Therefore, anybody can use xinetd to start special-purpose servers. ..."

The `/etc/xinetd.conf` file will include the scripts in `/etc/xinetd.d`. You can enable/disable a particular service from the configuration file for that service under the directory `/etc/xinetd.d`. You can start/stop/restart the xinetd service using the script `/etc/rc.d/init.d/xinetd`.

```
[root@feynman root]# /etc/rc.d/init.d/xinetd <start>|<restart>|<stop>
```

5.4 Telnet

Telnet is the good old protocol, to connect and work in a remote system. To connect to the remote system, the telnet server must be running on the remote system. To enable the telnet server you need to change the line

```
disable = yes
to
disable = no
```

in the file `/etc/xinetd.d/telnet`. Restart the xinetd daemon using:

```
[root@feynman root]# /etc/rc.d/init.d/xinetd restart
Stopping xinetd:                                           [  OK  ]
Starting xinetd:                                           [  OK  ]
[root@feynman root]#
```

Now you can telnet to feynman from ramanujan. Follow similar steps in the host ramanujan to telnet from feynman.

```
[root@feynman root]# telnet ramanujan
Trying 192.168.1.2...
Connected to ramanujan (192.168.1.2).
Escape character is '\^]'.
```

```
Red Hat Linux release 7.2 (Enigma)
Kernel 2.4.7-10 on an i686
login:
```

5.5 ssh

Telnet has been in wide spread use for almost two decades now. However, it has a potential secuirty problem. Both the data and password transmitted over the network are clear text. That means, any intermediate node on the network can sniff the telnet packets and crack down your password. To avoid this potential problem, most of the installation started switching over to the secure shell (ssh) instead of telnet. "ssh" encrypts all traffic (including passwords) to effectively eliminate eavesdropping, connection hijacking, and other network-level attacks.

OpenSSH is a FREE version of the SSH protocol suite. Additionally, OpenSSH provides a myriad of secure tunneling capabilities, as well as a variety of authentication methods. The OpenSSH suite includes the ssh program which replaces rlogin and telnet, scp which replaces rcp, and sftp which replaces ftp. Also included is sshd which is the server side of the package, and the other basic utilities like ssh-add, ssh-agent, ssh-keygen and sftp-server. OpenSSH supports SSH protocol versions 1.3, 1.5, and 2.0. You can download openssh from `http://www.openssh.org`.

You can run "sshd" as a standalone daemon. As a root user start the sshd using

```
[root@ramanujan root]# /etc/rc.d/init.d/sshd start
Starting sshd:                                             [  OK  ]
[root@ramanujan root]#
```

A typical ssh session will look like:

```
[mksarav@feynman mksarav]$ ssh mksarav@ramanujan
The authenticity of host '192.168.1.2 (192.168.1.2)' can't be established.
RSA key fingerprint is 71:ca:ad:00:93:0f:10:f8:7b:65:ef:cd:86:17:4d:e7.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.1.2' (RSA) to the list of known hosts.
mksarav@192.168.1.2's password:
Last login: Tue Jan 22 14:44:01 2002
[mksarav@ramanujan mksarav]$
```

"ssh" comes with lot of other utilities. For more details kindly refer the manual pages.

5.6 File Transfer Protocol: ftp

"ftp" is a widely used Internet file transfer program, to transfer the files between different hosts.

In Redhat 7.x system, by default the ftp server is disabled and if you try to access it will show the following error message:

```
[mksarav@ramanujan mksarav]$ ftp feynman
ftp: connect: Connection refused
ftp> quit
[mksarav@ramanujan mksarav]$
```

To enable the ftp server you need to change the line

```
disable = yes
to
disable = no
```

in the file `/etc/xinetd.d/wu-ftpd`. Restart the xinetd daemon using:

```
[root@feynman root]# /etc/rc.d/init.d/xinetd restart
Stopping xinetd:                                           [  OK  ]
Starting xinetd:                                           [  OK  ]
[root@feynman root]#
```

Now you can ftp to feynman from ramanujan.

```
[mksarav@ramanujan mksarav]$ ftp 192.168.1.1
Connected to 192.168.1.1.
220 feynman FTP server (Version wu-2.6.1-16) ready.
530 Please login with USER and PASS.
530 Please login with USER and PASS.
KERBEROS_V4 rejected as an authentication type
Name (192.168.1.1:mksarav): mksarav
331 Password required for mksarav.
Password:
230 User mksarav logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> help
Commands may be abbreviated.  Commands are:

!               cr              mdir            proxy           send
$               delete          mget            sendport        site
account         debug           mkdir           put             size
append          dir             mls             pwd             status
ascii           disconnect      mode            quit            struct
bell            form            modtime         quote           system
binary          get             mput            recv            sunique
bye             glob            newer           reget           tenex
case            hash            nmap            rstatus         trace
ccc             help            nlist           rhelp           type
cd              idle            ntrans          rename          user
```

```
cdup            image           open            reset           umask
chmod           lcd             passive         restart         verbose
clear           ls              private         rmdir           ?
close           macdef          prompt          runique
cprotect        mdelete         protect         safe
ftp> quit
221-You have transferred 0 bytes in 0 files.
221-Total traffic for this session was 315 bytes in 0 transfers.
221 Thank you for using the FTP service on feynman.
[mksarav@ramanujan mksarav]$
```

Follow similar steps in the host ramanujan to ftp from feynman.

scp

Similar to telnet, ftp also send everything including your password as plain text on the network thus creating a major security problem. Nowadays most of the organisations started using the "scp" program that comes as a part of "ssh".

```
[mksarav@feynman mksarav]$ scp file1.dat mksarav@ramanujan:
mksarav@ramanujan's password:
file1.dat           100% |*****************************|  3041       00:00
[mksarav@feynman mksarav]$
```

The above command will copy the `file1.dat` file in feynman to the host ramanujan under the home directory of user mksarav. Don't forget to type the colon at the end.

5.7 Web service

Web service is provided by the "httpd" daemon. There are many free and commercial web server programs are available. However, the "Apache" web server is very popular and is available freely under the GNU GPL License. To start the web service:

```
[root@ramanujan /]# /etc/rc.d/init.d/httpd start
Starting httpd:                                            [  OK  ]
[root@ramanujan /]#
```

Now, if you type `http://ramanujan` from feynman, you will see the test page for the Apache Web Server. From where this file came? In Redhat 7.x, you can put all your files in `/var/www/html`. By default there will a `/var/www/html/index.html` which contains the test page that you saw. You can modify this file as you want. If you want to use any cgi scripts using perl or python or whatever language you choose, then you have to put your scripts in the `/var/www/cgi-bin/` directory.

By default, the httpd daemon will listen at port 80, for any incoming HTTP request. You change many of the default settings in the file `/etc/httpd/conf/httpd.conf`.

Many commercial sites, will host more than one website on the same server. Further for providing webservice on Internet, you have to get a proper global IP address and register

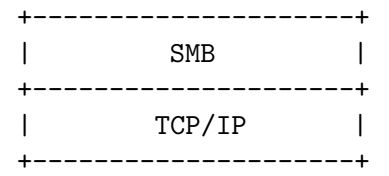


Figure: SMB (CIFS) implementation in WIN-2K and later versions

your domain name. You also need to add an DNS entry either with a commercial DNS service provider or you can run your own DNS server also. Only then your hostname will be resolved to the appropriate IP address from any part of the world. Explaining all these details is beyond the scope of this tutorial.

5.8 Samba

Samba is a set of tools originally written by Andrew Tridgell to share the resources such as disk and printers between UNIX and Windows hosts. IBM and Sytec developed a proprietary network system called "PC-Network" to share files between various PCs on a small LAN. This PC-Network used a small device driver known as NetBIOS (Network Basic Input/Output System). PC-Network came with its own proprietary hardware later being replaced by Ethernet. Lots and lots of software was written for use with the NetBIOS API (Application Programming Interface). Several vendors implemented the NetBIOS API on top of protocols such as IPX/SPX, and TCP/IP. NetBIOS over TCP/IP is often called NBT and has become the preferred NetBIOS transport. The workings of NBT are described in RFC#1001 and RFC#1002 (collectively known as Internet STD#19).

In the early 1980's, Intel and Microsoft came up with a protocol called SMB (Server Message Block) which was designed to run PC-Network LAN using the NetBIOS API to send and receive packets.

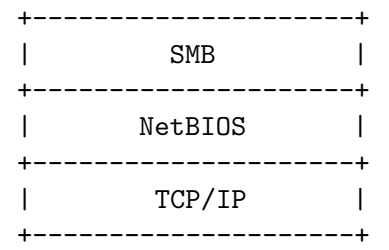


Figure: SMB implementation before WIN 2000

Microsoft used this in DOS, Win 3.1/9x. In Windows 2000, They modified the SMB protocol to run directly over the TCP/IP. However they maintained the backward compatibility. For later versions of SMB, Microsoft started using the name CIFS (Common Internet File System).

The main services of NetBIOS/TCP-IP are:

- Name Service
- Datagram Distribution Service
- Session Service

NetBIOS uses a name to identify each host on the network. Remeber NetBIOS names are nothing to do with Domain Name System and the whole thing happenned much before the DNS was invented. The NBT Name Service keeps track of which names are in use at which IP addresses, thus allowing the underlying IP network to find the nodes and transport NetBIOS messages between them. The Name Service runs on UDP port 137 whereas the Datagram Service (connectionless) and Session service (connection-oriented) runs on port 138 and 139 respectively.

```
[mksarav@feynman mksarav]$ grep netbios /etc/services
netbios-ns      137/tcp                         # NETBIOS Name Service
netbios-ns      137/udp
netbios-dgm     138/tcp                         # NETBIOS Datagram Service
netbios-dgm     138/udp
netbios-ssn     139/tcp                         # NETBIOS session service
netbios-ssn     139/udp
[mksarav@feynman mksarav]$
```

In Windows, NetBIOS Name Service is handled by WINS (Windows Internet Name Service). Again don't confuse this with the DNS. Both are different though the job is similar. If you sent a query like "Hey, who is the host NEWTON?" – then the WINS server will send a reply with the corresponding IP address.

Samba is the best known and most popular open source implementation of SMB (CIFS). Whatever resources like disks, printers etc... that you can share with other hosts in the network are called SMB SHARES. There are four basic things one can do with Samba:

1. Share a GNU/Linux drive with WIN machines.
2. Access an SMB share with GNU/Linux machines.
3. Share a GNU/Linux printer with Windows machines.
4. Share a WIN printer with GNU/Linux machines.

The `smbd` and `nmbd` daemons in the Samba package provides the necessary service. The later provides the NetBIOS nameserver support to clients. In Redhat 7.x systems, you can start the samba services using:

```
[root@feynman root]# /etc/rc.d/init.d/smb start
Starting SMB services:                                     [  OK  ]
Starting NMB services:                                     [  OK  ]
[root@feynman root]#
```

The whole Samba service is controlled by the settings in the `/etc/samba/smb.conf` file. There are too many options that you can use in `smb.conf` file which can't be covered in this tutorial. For this tutorial let us assume, you have WINDOWS in the host ramanujan. You reboot the host ramanujan into WINDOWS and configure it with the same TCP/IP settings as it was in GNU/Linux. For simplicity, let us use the same hostname ramanujan for the WINDOWS computer name. Further you should have installed the `Files and Printer Sharing for Microsoft Networks`. You can do this from `Control Panel` → `Network Connections` → `Local Area Connections`. If you don't know how to do this, ask a local WINDOWS Guru.

Accessing GNU/Linux home directory from WINDOWS

Let us explain the working of samba with a simple `smb.conf` file:

```
[mksarav@feynman mksarav]$ cat /etc/samba/smb.conf
[global]
        workgroup = mks
        interfaces = 192.168.1.1/24
        encrypt passwords = yes
        smb passwd file = /etc/smbpasswd
[homes]
        guest ok = no
        read only = no
```

The generic settings and the one which applies to all the SMB shares are done under the [global] section. Here we are creating a workgroup called "mks" and asking samba to use the interface 192.168.1.1. The /24 is to indicate the network id. Further we are forcing samba to use encrypted password and use the `/etc/smbpasswd` file instead of `/etc/passwd`. The [homes] section tells that no guest login is allowed and only users with valid account in the GNU/Linux machine is allowed to login.

Since we asked to use `/etc/smbpasswd` file, first we need to create Samba username and password. We can do this using:

```
[root@feynman root]# touch /etc/smbpasswd
[root@feynman root]# smbadduser mksarav:mksarav
------------------------------------------------------------
ENTER password for mksarav
New SMB password:
Retype new SMB password:
Added user mksarav.
```

If there is no file called `/etc/smbpasswd` initially then `touch /etc/smbpasswd` will create a empty file. The command `smbadduser mksarav:mksarav` tells the mapping between the username in `/etc/passwd` and Samba username. Here we want to use the same username in both. Since we have changed the configuration details, we have to restart the `smbd` daemon.

```
[root@feynman root]# /etc/rc.d/init.d/smb restart
Shutting down SMB services:                                [  OK  ]
Shutting down NMB services:                                [  OK  ]
Starting SMB services:                                     [  OK  ]
Starting NMB services:                                     [  OK  ]
[root@feynman root]#
```

Samba package comes with its own set of utilities like `smbclient`, `smbstatus`, `smbpasswd`, `smbprint`, `smbmount`, `smbumount` etc... to access SMB/CIFS resources on servers. `smbclient` is a ftp-like client to access SMB/CIFS resources. Let us first check whether we are able to access the home directories in the host feynman, using smbclient. i.e. we are essentially testing Samba from the same host.

```
[mksarav@feynman mksarav]$ smbclient -L feynman
added interface ip=192.168.1.1 bcast=192.168.1.255 nmask=255.255.255.0
Password:
Anonymous login successful
Domain=[MKS] OS=[Unix] Server=[Samba 2.2.1a]

        Sharename      Type      Comment
        ---------      ----      -------
        homes          Disk
        IPC$           IPC       IPC Service (Samba 2.2.1a)
        ADMIN$         Disk      IPC Service (Samba 2.2.1a)

        Server               Comment
        ---------            -------
        FEYNMAN              Samba 2.2.1a

        Workgroup            Master
        ---------            -------
        MKS                  FEYNMAN
[mksarav@feynman mksarav]$
```

The `-L` option allows you to look at what services are available on a server. When it ask for password just press enter. The output shows there is a SHARE called "homes" available on this server. Let's see whether we are able to access this share.

```
[mksarav@feynman mksarav]$ smbclient //feynman/homes -U mksarav
added interface ip=192.168.1.1 bcast=192.168.1.255 nmask=255.255.255.0
Password:
Domain=[MKS] OS=[Unix] Server=[Samba 2.2.1a]
smb: \> help
ls             dir            du             lcd            cd
pwd            get            mget           put            mput
```

```
rename          more            mask            del             open
rm              mkdir           md              rmdir           rd
prompt          recurse         translate       lowercase       print
printmode       queue           cancel          quit            q
exit            newer           archive         tar             blocksize
tarmode         setmode         help            ?               history
!
smb: \>
```

The `-U` option tells the SMB username. The SMB service name is specified with `//NetBIOS hostname/SMB share name`. Here the NetBIOS hostname of the system is `feynman` and the SMB share name is `homes`. The `smb: \>` interface will exactly look like ftp interface and you can get/put the files from/to the mksarav home directory. So far we have been testing everything from the same host.

Let us see what WINDOWS Network Neighbourhood (or from My Network places) shows. Remember you should have enabled the settings `NetBIOS over TCP/IP` in the TCP/IP properties of the host ramanujan (now booted with WINDOWS). The Network Neighbourhood will now show a workgroup called "Mks". If you double click it, it will show something like:

```
Samba 2.2.1a (Feynman)
```

If you double click Feynman, then it will prompt you for a username/password. Here i have login with my username "mksarav" (smbusername) and my password (smbpassword). Now you will see something like:

```
homes
mksarav
Printers and Faxes
```

Here both the shares `homes` and `mksarav` represent the home directory of the user mksarav in the host feynman. Now you can do anything with this share just like another windows folder.

Alternatively, you can also access the SMB service directly from the Internet Explorer browser by typing \\computername\sharename.

Accessing WINDOWS shared folder from GNU/Linux

Let us share the folder `C:\MP3>` in the host ramanujan and try to access it from feynman. Right Click the folder MP3 in ramanujan, you will see sharing and security options. In that enable the `Share this folder` option and give the share name as MP3. This process might be slightly different depending upon whether you are using WIN9x/WIN2K. If you are in doubt, you can ask any local WIN expert.

Now, we can try to access this share from the GNU/Linux host feynman.

```
[mksarav@feynman mksarav]$ smbclient -L ramanujan -U Administrator
```

```
added interface ip=192.168.1.1 bcast=192.168.1.255 nmask=255.255.255.0
Got a positive name query response from 192.168.1.2 ( 192.168.1.2 )
Password:
Domain=[WORKGROUP] OS=[Windows 5.1] Server=[Windows 2000 LAN Manager]

        Sharename       Type      Comment
        ---------       ----      -------
        IPC$            IPC       Remote IPC
        mp3             Disk

        Server               Comment
        ---------            -------
        RAMANUJAN

        Workgroup            Master
        ---------            -------
        MKS                  FEYNMAN
        WORKGROUP            RAMANUJAN
[mksarav@feynman mksarav]$
[mksarav@feynman mksarav]$ smbclient //ramanujan/mp3 -U Administrator
added interface ip=192.168.1.1 bcast=192.168.1.255 nmask=255.255.255.0
Got a positive name query response from 192.168.1.2 ( 192.168.1.2 )
Password:
Domain=[WORKGROUP] OS=[Windows 5.1] Server=[Windows 2000 LAN Manager]
smb: \> ls
  .                                   D        0  Thu Feb 14 21:44:12 2002
  ..                                  D        0  Thu Feb 14 21:44:12 2002
  Collections                         D        0  Thu Feb 14 21:51:40 2002
  new                                 D        0  Wed Feb 20 23:24:02 2002

                57175 blocks of size 262144. 44733 blocks available
smb: \> quit
[mksarav@feynman mksarav]$
```

In a big network, mostly you will find a WINS server running. In that case, you can include the option `wins server` in `/etc/samba/smb.conf`. Here is the file I use to access my university network SMB shares. For security reason the IP address has been blanked with x.

```
[global]
        workgroup = mks
        interfaces = 137.132.x.x/x
        encrypt passwords = yes
        smb passwd file = /etc/smbpasswd
        wins server = 137.132.x.x
```

```
[homes]
        guest ok = no
        read only = no
```

Here is how i will access my university network:

```
[mksarav@hanuman mksarav]$ smbclient -W nusstf -L nts27 -U dcssmk
Password:
```

Here the option `-W` is used to mention the DOMAIN.

Sharing the printers is not covered in this tutorial but i will add it in a later version of this tutorial. For more information on SMB you can read `SMB-HOWTO` and also go through `http://www.samba.org`. At samba site, you will find lot of documentation.

6 Network Troubleshooting Utilities

A variety of network troubleshooting utilites are availble for GNU/Linux. In this section, we can see some of the essential utilities.

6.1 ping

The most commonly used utility to find out whether a host is alive or not is the "ping". We saw how to do this in Sec.5 on page 12. Some of the useful option with ping are:

-R This option will set the RECORD_ROUTE option in the ECHO_REQUEST packet. When you use this option it will show the IP addresses of the intermediate routers that the packet is traversing. Since all these extra details are stored in the IP optional header space, at the maximum it can display only nine such routes. You can use `traceroute` instead of this.

-s Specifies the number of data bytes to be sent. The default is 56. Thus the size of the outgoing IP packet will be 20 (IP header) + 8 (ICMP header) + 56 = 84 data bytes.

-f Flood ping. Outputs packets as fast as they come back or one hundred times per second, whichever is more. This option will be very much useful for doing some performance measurements. For every ECHO_REQUEST sent a period "." is printed, while for every ECHO_REPLY received a backspace is printed. This provides a rapid display of how many packets are being dropped. Only the super-user may use this option. This will create excessive traffic on a network and should be used with caution.

Go through the man pages for more details and some of the pitfalls of ping.

6.2 tcpdump

If you want to see how the real packets will look like on the network, then `tcpdump` is your friend. It prints out the headers of packets on a network interface. You can specify which packets you want to see. A variety of boolean expression can be mentioned to filter and

see particular types of packets. The most commonly used option is `-i` to tell tcpdump to capture the packets from that particular interface. Let us ping ramanujan from feynman and see the tcpdump output. In one of the console you run the `tcpdump -i eth0 icmp` command as root and from other console you do `ping -c1 ramanujan`.

```
[root@feynman root]# tcpdump -i eth0 icmp
tcpdump: listening on eth0
16:09:58.134196 feynman > ramanujan: icmp: echo request (DF)
16:09:58.134354 ramanujan > feynman: icmp: echo reply

136 packets received by filter
0 packets dropped by kernel
[root@feynman root]#
```

The `icmp` option above will tell tcpdump to show only the ICMP packets.

Some of the useful options are:

-s With this option you can specify a particular number of bytes of data to be collected from each packet rather than the default of 68.

-w You have to mention a file name so that the raw packets will be written to that file rather than parsing and printing them out on the console. They can be later printed with the -r option.

Don't underestimate the use of `tcpdump`. You can do a variety of troubleshooting using it. **W. Richard Stevens** has used this tool extensively for his book **TCP/IP Illustrated - Vol.I**.

6.3 ipgrab

`ipgrab` is similar to tcpdump but much more user friendly. It will print out the packet headers in plain ASCII output if you want. Here is the "ipgrab" output of the `ping -c1 feynman`:

```
[root@ramanujan root]# ipgrab -ieth0 icmp

ipgrab 0.9.8
Listening on device eth0 (ethernet)

********************************************************************************
                        Ethernet (1014193584.000364)
--------------------------------------------------------------------------------
Hardware source:        00:60:97:db:8d:96
Hardware destination:   00:90:27:90:e6:6b
Type / Length:          0x800 (IP)
Media length:           98
--------------------------------------------------------------------------------
```

```
                        IP Header
-------------------------------------------------------------------------
Version:                4
Header length:          5 (20 bytes)
TOS:                    0x00
Total length:           84
Identification:         0
Fragmentation offset:   0
Unused bit:             0
Don't fragment bit:     1
More fragments bit:     0
Time to live:           64
Protocol:               1 (ICMP)
Header checksum:        46681
Source address:         192.168.1.2
Destination address:    192.168.1.1
-------------------------------------------------------------------------
                        ICMP Header
-------------------------------------------------------------------------
Type:                   8 (echo request)
Code:                   0
Checksum:               51972
Identifier:             23832
Sequence number:        0
*************************************************************************
                        Ethernet (1014193584.000527)
-------------------------------------------------------------------------
Hardware source:        00:90:27:90:e6:6b
Hardware destination:   00:60:97:db:8d:96
Type / Length:          0x800 (IP)
Media length:           98
-------------------------------------------------------------------------
                        IP Header
-------------------------------------------------------------------------
Version:                4
Header length:          5 (20 bytes)
TOS:                    0x00
Total length:           84
Identification:         24552
Fragmentation offset:   0
Unused bit:             0
Don't fragment bit:     0
More fragments bit:     0
Time to live:           255
```

```
Protocol:               1 (ICMP)
Header checksum:        55152
Source address:         192.168.1.1
Destination address:    192.168.1.2
-------------------------------------------------------------------------
                        ICMP Header
-------------------------------------------------------------------------
Type:                   0 (echo reply)
Code:                   0
Checksum:               54020
Identifier:             23832
Sequence number:        0


6 packets received
0 packets dropped by kernel
ARP:    0
IP:     2
ICMP:   2
IGMP:   0
UDP:    0
TCP:    0
IPX:    0
GRE:    0
IPv6:   0
ICMPv6: 0
OSPF:   0
RSVP:   0
AH:     0
ESP:    0
```

You can download ipgrab from `http://ipgrab.sourceforge.net`. First go through the manual pages which is available on the same site.

6.4 netstat

`netstat` will print network connection details, routing tables, interface statistics, and other details. You will appreciate the output of netstat if you know how the TCP/IP protocols work.

6.5 traceroute

Using the `ping -R` option you can't see much about the routes that a packet traverse to reach the destionation host. However `traceroute` will provide you the details of each

intermediate router that the packet traverse. However the packet need not has to go through the same route each time.

```
[mksarav@hanuman mksarav]$ /usr/sbin/traceroute mksarav.tripod.com
traceroute to mksarav.tripod.com (209.202.196.70), 30 hops max, 38 byte packets
 1  gw6509a-81a.comp.nus.edu.sg (137.132.81.6)  0.571 ms  0.474 ms  0.435 ms
 2  115-19.priv.nus.edu.sg (172.18.115.19)  0.668 ms  0.696 ms  0.640 ms
 3  core-pgp-vlan143.priv.nus.edu.sg (172.18.20.129)  0.711 ms  0.739 ms  0.703 ms
 4  core-cc-vlan21.priv.nus.edu.sg (172.18.20.5)  0.735 ms  0.685 ms  0.651 ms
 5  svrfrm1-cc-vlan165.priv.nus.edu.sg (172.18.20.90)  0.993 ms  0.786 ms  0.808 ms
 6  gk-pix-f1-821.nus.edu.sg (137.132.3.130)  2.383 ms  3.507 ms  5.626 ms
 7  165.21.48.101 (165.21.48.101)  178.774 ms  175.404 ms  184.515 ms
 8  GE-1-1-0.bedok.singnet.com.sg (165.21.12.1)  183.482 ms  185.186 ms  185.768 ms
 9  POS2-0.tp-core1.ix.singtel.com (202.160.250.53)  176.590 ms  178.955 ms  184.295 ms
10  POS0-1.above-core1.ix.singtel.com (202.160.250.34)  368.292 ms  363.799 ms 385.227 ms
11  POS2-0.paix-core1.ix.singtel.com (202.160.250.46)  379.003 ms  375.040 ms 377.120 ms
12  paix.exodus.net (198.32.176.15)  364.584 ms  374.864 ms  370.863 ms
13  bbr02-p3-0.sntc08.exodus.net (209.185.9.233)  383.027 ms  378.060 ms  376.887 ms
14  bbr01-p8-0.sntc04.exodus.net (206.79.9.186)  374.507 ms  369.439 ms  372.050 ms
15  bbr01-p1-0.ftwo01.exodus.net (209.185.9.110)  421.193 ms  417.206 ms  416.456 ms
16  bbr02-p3-0.ekgv01.exodus.net (206.79.9.54)  447.018 ms  437.996 ms  436.417 ms
17  bbr01-p2-0.okbr01.exodus.net (206.79.9.129)  433.585 ms  421.279 ms  436.054 ms
18  bbr02-p6-0.wlhm01.exodus.net (209.185.9.118)  448.084 ms  443.928 ms  442.889 ms
19  dcr03-g1-0.wlhm01.exodus.net (64.14.70.49)  451.590 ms  450.102 ms *
20  csr02-ve243.wlhm01.exodus.net (64.14.70.26)  447.934 ms *  455.011 ms
21  209.67.242.86 (209.67.242.86)  451.882 ms !X  447.694 ms !X  446.326 ms !X
[mksarav@hanuman mksarav]$
```

Find out from the `man traceroute` what the `!X` in the output means.

6.6 tracepath

`tracepath` traces path to a destination host discovering MTU (Maximum Transmission Unit) along the path. This will help you to identify the bottleneck link in the path to a destination.

```
[mksarav@hanuman mksarav]$ /usr/sbin/tracepath www.bbc.co.uk
 1?: [LOCALHOST]      pmtu 1500
 1:  gw6509a-81a.comp.nus.edu.sg (137.132.81.6)              2.072ms
 2:  115-19.priv.nus.edu.sg (172.18.115.19)                  2.282ms
 3:  core-pgp-vlan143.priv.nus.edu.sg (172.18.20.129)        2.384ms
 4:  core-cc-vlan21.priv.nus.edu.sg (172.18.20.5)            2.392ms
 5:  svrfrm1-cc-vlan165.priv.nus.edu.sg (172.18.20.90)       2.394ms
 6:  gk-pix-f1-821.nus.edu.sg (137.132.3.130)                5.078ms
 7:  165.21.48.101 (165.21.48.101)                         181.171ms
 8:  GE-1-1-0.bedok.singnet.com.sg (165.21.12.1)           asymm  9 173.820ms
 9:  POS2-0.tp-core1.ix.singtel.com (202.160.250.53)       174.541ms
10:  POS1-0.tp-core2.ix.singtel.com (202.160.250.150)      384.624ms
11:  POS2-0.paix-core1.ix.singtel.com (202.160.250.46)     393.607ms
12:  198.32.176.151 (198.32.176.151)                       392.192ms
```

```
13:  rt0-POS3-0.sjc.bbc.co.uk (212.58.255.173)          396.120ms
14:  rt1-S2-1.111ny.bbc.co.uk (212.58.255.169)          486.564ms
15:  rt0.111ny.bbc.co.uk (212.58.255.137)               488.105ms
16:  rt0-POS5-0.thny.bbc.co.uk (212.58.255.41)          477.914ms
17:  www2.thny.bbc.co.uk (212.58.240.32)                asymm 18 487.517ms reached
     Resume: pmtu 1500 hops 17 back 18
[mksarav@hanuman mksarav]$
```

The above output shows that you can't send a packet with size > 1500 to reach the destination. If you exceed this limit, the IP packet will get fragmented into smaller packet size (< 1500). Let us see the effect with an example. Using `ping -s` option let us first ping the host `www.bbc.co.uk` with 1472 data bytes (so that the packet size will be 1472+8+20=1500):

```
[mksarav@hanuman mksarav]$ ping -c2 -s1472 www.bbc.co.uk
PING www.bbc.net.uk (212.58.240.31) from 137.132.81.78 : 1472(1500) bytes of data.
1480 bytes from www1.thny.bbc.co.uk (212.58.240.31): icmp_seq=0 ttl=238 time=457.837 msec
1480 bytes from www1.thny.bbc.co.uk (212.58.240.31): icmp_seq=1 ttl=238 time=456.507 msec

--- www.bbc.net.uk ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max/mdev = 456.507/457.172/457.837/0.665 ms
```

Now let us repeat the experiment with 1473 data bytes (packet size now will be 1473+8+20=1501):

```
[mksarav@hanuman mksarav]$ ping -c2 -s1473 www.bbc.co.uk
PING www.bbc.net.uk (212.58.240.31) from 137.132.81.78 : 1473(1501) bytes of data.

--- www.bbc.net.uk ping statistics ---
2 packets transmitted, 0 packets received, 100% packet loss
[mksarav@hanuman mksarav]$
```

Why it shows 100% packet loss? In the IP header there is a bit called "Don't fragment bit" to tell the intermediate routers not to fragment the packet. ping sends all the packets with the "Don't fragment" bit set in the IP header but unfortunately the first host MTU is 1500 and it needs to fragment the packet. But ping told not to fragment it. So it will simply discard the packet.

7 Before THE END

You may also go through the basic GNU/Linux exercises by the author at `http://mksarav.tripod.com`. This entire tutorial is prepared using LaTeXwith `vim` editor. The first two digrams were drawn using the GNU pic. I thank the entire Open Source and Free Software Community for providing a whole lot of tools and OS, completely free for the entire community. Especially I am very much inspired by Donald E Knuth for his TeXand RMS (Richard M Stallman) for contributing his entire life for the Free Software.