Linux Networking

Biniam Gebremichael
Kidane Yemane

Technical report 2000-002
January 2000
ISSN 1404-3203

Department of Information Technology
Uppsala University

Linux Networking

Uppsala University

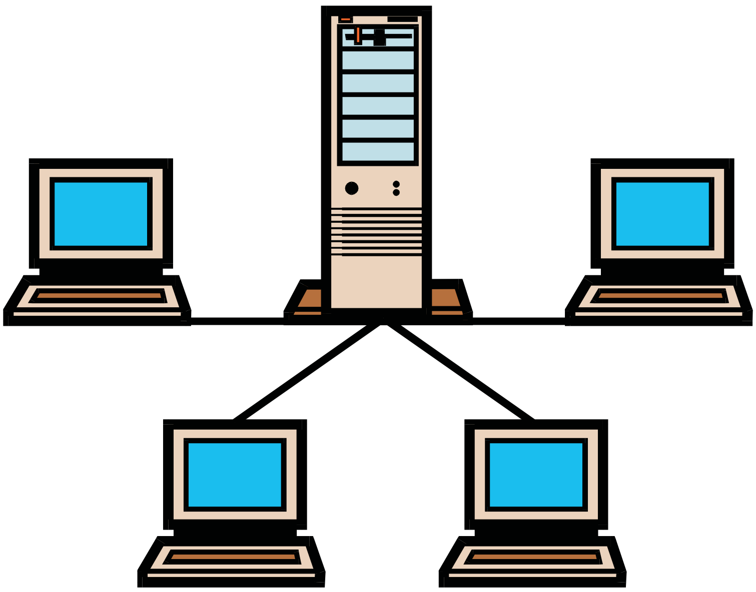
Department of Computer Systems

Uppsala, Sweden

Biniam Gebremicheal
biniam@asmara.csd.uu.se

Kidane Yemane
kidane@asmara.csd.uu.se

20th January 2000



Contents

Abstract

Educational institutions and other groups with limited resources can enjoy the full feature of the Unix operating system by using the free operating system Linux. In this thesis we present a brief discussion on the most popular operating systems in use with a special emphasis on Linux. We provide information and guidelines starting from Linux installation all the way to setting up a Local Area Network (LAN). Included is a discussion and configuration of a dial-up network, a wireless network and LAN security.

1 Motivation

Today, knowledge and information are widely recognized as prime engines of progress driving improvement in education, leadership, economic and social well development . Nations without the means to harness these dynamic forces will surely be left behind. Much of the rapid development in industrialized countries has been achieved through cutting-edge information technology. At the same time, nations through out the developing world are eager to gain access to the same technologies in an effort to keep pace. Simply, because:

1. Computer networks, which have become cheaper to acquire and easier to use, make available ever increasing amount of information. Such networks, by providing *access to Internet*, can help overcome the isolation that scientists in developing countries often experience.

2. For countries like Eritrea, Computer network, besides its feasibility to Internet connectivity, has additional importance as far as computer demand and their costs are concerned. For instance computers in Eritrea are very rare and only few percent of the population has access to them. Thus, *resource sharing* would be a good solution for increasing accessibility.

3. With the help of network operating systems, like Linux, a computer is able to serve as many users as possible and provides some degree of *security and privacy* in sharing the communal machine. An important feature that the traditional stand-alone operating systems can not provide.

Yet, what is the best way to get on board the high tech bandwagon in an effort to keep pace?

Due to limited financial support, insufficient basic infrastructure and other problems that developing countries facing. The task of establishing a reliable computer network gets harder. Despite these facts, Computer networks, which serve as the backbone of such an effort, must be designed, built, operated and maintained throughly. Similarly, the availability of well-tested realistic documents, that provides guidelines in establishing such networks, are a prerequisite to success. By "realistic document", we mean:

1. One that puts the above constraints under consideration.

2. Provides a way of taking advantage of today's high-tech communications by allowing the process to bypass older techniques that relay on hard-to-maintain systems.

3. Guides to the establishment of networks compatible to the existing worldwide standards. Avoids future modification.

With these criteria, in mind, this thesis presents different ways of tackling the problem. Linux, today's most attractive Network Operating System (the tool that is used here) serves very effectively for such a purpose. It provides all network functionality, while unrevealing behind the screen events for better training and qualifications. The thesis explains in detail all necessary configuration of "Linux TCP/IP network on top of Ethernet", Dial-Up Networking, Wireless networking, ... We hope that our work will be a significant contribution toward global connectivity in developing countries.

2 Introduction

A decade ago, Computer Networking in general and Internet in particular was classified as a luxury by many institutions and communities. But few years later, it turns out to be the fundamental infrastructure to business, research, education, health and other sectors. Computer networking has made the traditional data processing to be accomplished more easily and cheaply. Hardware and software are *sharable* without regards to the physical location of the resource and the user. High reliability is attained by having *alternative sources of supply*. *Money is saved* by keeping data on one powerful server and make it available for clients. The ability to increase system performance gradually (*scaling*) is granted as the workload grows.

In achieving these goals, various software and hardware was designed and different connection topologies invented. Linux is one of the leading multi-user operating systems that have been produced for this purpose. It is free, reliable and efficient. This thesis talks about installing, administrating and maintaining a Linux-based network. All the experiments presented here have been run on a one-desktop-three-laptop network and the configuration files provided refers to this actual network.

The thesis is organized as follows: in the first introductory part, chapters 1 and 2, the leading network operating systems (NOS) are compared and the advantages of employing Linux for efficient networking on inexpensive hardware is listed. The hardware and disk space requirement are discussed, disk partitioning and installing more than operating system is exemplified.

Chapters 3-7 are about setting up a local area network (LAN) among Linux machines. Network interface , IP masaquarding, Network Information System (NIS), Network File System (NFS) and Domain Name Service (DNS) are presented and explained in detail. Using a running example all procedures are illustrated.

Dial-Up networking , which is mostly used for Internet Service Providing (ISP), is presented in Chapter 8. A Point-to-Point Protocol(PPP) Dial-out and Dial-in configuration for a Linux box is shown.

In Chapter 9, wireless networking is presented as another technology for connecting computers or a network of computers with another network.

Wireless networking, like radio communication, is an important alternative of establishing a network of computers for areas with unstable telecommunication. It can be inexpensively operated and maintained, as shown in this thesis.

3 Network Operating Systems

3.1 UNIX

UNIX is one of the most popular operating systems worldwide because of its large support base and distribution. It was originally developed in the mid-1970s as a multitasking system for minicomputers and mainframes. What makes UNIX so important is not only its popularity for multiuser machines, it is also the foundation for the majority of the free software world. If you have access to the Internet, nearly all of the free software available there is written specifically for UNIX system. (The Internet itself is largely UNIX-based). This makes UNIX a superior network operating system than the other NOS brands like Windows NT and $OS/_2$. Specifically for universities and other teaching institutes, UNIX is the best (and may be the only one), for learning and experiencing the behind-the-screen of network operations. It is highly automatable, scalable and offers a better reliability and flexibility; simply because it is designed to do so[7].

Versions of UNIX exist for many systems-ranging from personal computers to supercomputers. Most versions of UNIX for personal computers are quite expensive and cumbersome. In a commercial UNIX development house, the entire system is developed with a rigorous policy of quality assurance, source and revision control systems, documentation, and bug reporting and resolution. These houses are large enough to employ hundreds(if not thousands) of programmers, testers, documenters, and administrative personal which makes the product very expensive for users with less financial support. For large and sensitive businesses and companies such expensive documentation and proofs might be indispensable, but for ordinary users, who does not care much about an extensive source control system but cares more about its cost, another alternative is often required.

3.2 Linux

While UNIX remains to be the superior, Its cost makes developers to think about another free UNIX implementation(Linux) through which the cost-sensitive community can be part of the computing world. Linux was first written from scratch by Linus Torvalds at the University of Helsinki in Finland. Since its birth, Linux was (and still is) developed by a group of volunteers on the Internet from all over the world. Across the Internet and beyond, anyone with enough know-how has the opportunity to aid in developing and debugging the kernel, porting new software, writing documentation, or helping new users. There is no single organization responsible for developing the system.

Today, Linux has all the feature you would expect in a modern fully-fledged Unix, including true multitasking, virtual memory, shared libraries, demand loading, shared copy-on write executables, proper memory management and TCP/IP networking. You might be amazed that such a non-structured group of volunteers, programming and debugging a complete system, could get anything done at all, let alone producing a UNIX-like operating system. However, it turns out to be one of the most efficient and motivated development effort ever employed. The entire Linux kernel was written form scratch, without employing any code from proprietary sources. A great deal of work was put forth by volunteers to port all of the free software under the sun to the Linux system, libraries were written and ported, file systems developed, and hardware drivers written for many popular devices.

While feature-wise, Linux remains quite similar to other implementations of UNIX for PC, there are some differences between Linux and commercial versions of UNIX for PC.

1. Linux supports a different range of hardware for commercial implementations. In general, Linux supports the most well-known hardware devices, but support is still limited to that hardware which developers actually have access to. However, commercial UNIX vendors generally have a wider support base, and tend to support more hardware.

2. Commercial implementations of UNIX usually come bundled with a complete set of documentation as well as user support from the vendor. In contrast, most of the documentation for Linux is limited to documents available on the Internet and few books. But with the growing popularity of Linux, it will be soon supported with enough

documentation[5].

3. Linux is still (and will always be) under development, and certain features (such as TCP/IP networking) are less stable(in 1997). But as far as general stability and robustness are concerned, many users have reported that Linux is at least as stable as commercial UNIX systems.

4. The most important factor to consider for many users is price. The Linux software is free. If you have access to the Internet you can downloaded it. You can also purchase it via mail order on diskette, tape, or CD-ROM. Linux is not distributed on a "single machine" license and you can copy or share it with a friend.

Along with the price of commercial UNIX one usually pays for documentation, support, and assurance of quality. These guarantees are very important factors for large institutions, but personal computer users may not require these benefits. In any case, many businesses and universities are finding that running Linux on a lab of inexpensive personal computers is preferable to running a commercial version of UNIX in a lab of workstations. Linux can provide the functionality of a workstation on PC hardware at a fraction of the cost, and many personal UNIX users are moving to Linux.

3.3 Windows NT

Windows NT is another network operating system. NT, like UNIX versions and Linux, can communicate with many different types of computers, it can secure data and keep unauthorized users off the network and fulfills all the requirement for an operating system functioning in a networked environment[2].

The two main characteristics that makes windows NT different from UNIX and its clones are: its graphical user interface and its centralized system.

1. NT's GUI: NT has long enjoyed an intuitive user interface for managing single systems, largely benefiting from the exceptional familiarity of the windows look-and-feel, adopted by the NT GUI. Since many users

where already familiar with the single system GUI of Microsoft products, many users still believes NT's GUI is better. However, as users begin to deploy large numbers of servers, and geographically-dispersed servers, some of NT's architectural shortcomings for system management have become more apparent, deriving primarily from its design as a single-user system. The multi-user design of UNIX supports remote access at multiple levels, including the ability to login with a character session, via telnet, to edit configuration files, running GUI tools over the network-enabled X Window System, and now through Java versions of system management tools. NT currently enjoys almost none of these features [7]. Besides, unlike the UNIX/Linux user interface, where you are notified about what is going on, NT's GUI hinders users from following what is happening behind the screen.

2. Centralized system: Unlike UNIX and its clones, windows NT is a propriety system. The interface and design are controlled by a single corporation(Microsoft), and only this corporation may implement that design. The advantage of a centralized design system is that, it sets a strict standard for programming and user interface. NT is NT wherever you go. While for Linux, because of its diversity and uncentralized system and the many needs which the software is attempting to meet, running the system is not an easy task. It will take a considerable amount of effort and attention on the administrators part to keep the system running and take care of things.

Despite of these facts Linux remains to be the best choice as far as minimum cost maximum functionality and scalability is needed. If a company is small to medium-sized, has few mission-critical processes to be run, is willing to hire additional administrators for their Microsoft Exchange as well (mail service) and Internet Information Server(s), and has a substantial budget for Microsoft's "per server" or "per seat" licensing scheme, then NT would be the operating system of choice [7]. Apart from this environment, NT will remains the last chosen multitasking operating system.

4 Installation of Linux

Linux installation manual comes together with the operating system, or can be found electronically in the online documentation, mostly in */usr/doc/LDP/*(for Red Hat Linux). This section discusses about the hardware requirements and disk space partition for installing Linux. Installing two operating systems(Linux and Windows 9x) on one hard disk is discussed at the end of this section, and configuring the LInux LOading(LILO) is also discussed. For complete Linux installation please refer to the installation manual [8].

4.1 Hardware Requirements

4.1.1 On what hardware does it run?

Linux was first developed for 386/486-based PCs. These days it also on DEC Alphas, SUN Sparcs, M68000 machines(like Atari Amiga), MIPS and Power-PC.

4.1.2 Memory requirements

Linux requires very little memory to run compared to other advanced operating systems. You should have at the very least 2 megabytes of RAM; however, it is strongly suggested that you have 4 megabytes. The more memory you have the faster the system will run. Linux can support the full 32-bit address range of the 386/486; in other words, it will utilize all of your RAM automatically.

4.1.3 Hard drive space requirements

The amount of hard drive space that you will require depends greatly on your needs and amount of software that you're installing. You could run a complete system in 10 to 20 megabytes of space on your drive. (you can run a minimal system completely from floppy, with all the functionality of networking). However, if you want to have a room for expansion, and for larger packages such as X-Windows, you will need more space. The following can be taken as an example.

For a normal installation

One partition for /	with 80 to 100MB
one partition for swap	with 16 to 64MB
one partition for /usr	with 400 to 600MB
one partition for /home	with 20 to ??? MB per user

For Server installation

one partition for /	with 40 to 60MB
one partition for swap	with 32 to 64MB
one partition for /tmp	with 80 to 100M
one partition for /var	with 40 to 80MB
one partition for /var/spool/mail	with 5 to ??MB per user
one partition for /var/spool/news	with 100 to ???MB depending on news traffic
one partition for /httpd	with 40 to ??MB depending on number of www pages.
one partition for /usr	with 400 to 600MB
one partition for /home	with 20 to ??MB per user.

The order is important this means the order in the file /etc/fstab represents the order in which this partitions will be mounted. Installing Linux, especially from some distribution, like Red hat is very easy and highly robust. All it takes is answering some question about your hardware profile, and choosing various software packages to configure.

4.2 Installing more than one OS onto one hard disk

If you are planing to install Linux operating system, but didn't want to wipe out your old operating system, in which you have enough expertise. You can keep both operating systems on different partitions of the same hard disk. This will be further illustrated by way of an example, but first a brief introduction to LILO is inorder. **LILO** is the most used **LI**nux **LO**ader for the x86 flavour of Linux; When LILO boots the system, it uses BIOS calls to load the Linux kernel off the disk (IDE drive, floppy or whatever). Therefore, the kernel must live in some place that can be accessed by the bios.

At boot time, LILO is not able to read filesystem data, and any pathname you put in /etc/lilo.conf is resolved at installation time (when you invoke /sbin/lilo). Installation time is when the program builds the tables that list which sectors are used by the files used to load the operating system. As a consequence, all of these files must live in a partition that can be accessed by the BIOS (the files are usually located in the /boot directory, this means that only the root partition of your Linux system needs to be accessed via the BIOS).

Another consequence of being BIOS-based is that you must reinstall the loader (i.e., you must reinvoke /sbin/lilo) any time you modify the LILO setup. Whenever you recompile your kernel and overwrite your old image you must reinstall LILO.

Intention: You want to install Linux together with Dos/Windows 95 /windows NT and want both systems to boot from the hard drive.

precondition: There must be a primary partition as a boot partition both for Dos/Windows 95/windows NT and Linux. These partitions must reside below the 1024-cylinder boundary. If LILO does not reside below 1024- cylinder boundary. You will experience error messages like " Li-" or another incomplete part of the word LILO, during booting from the hard drive this is because LILO can not find the boot kernel.

cause: At boot-up time LILO has only the BIOS drivers at its disposal in order to access the hard drives. Thus the reachable area is limited to

- the cylinders 0 to 1023 of the hard drives within reach.
- the first two physical hard drives
- (E)IDE-drives, if available, It will then be impossible to boot from SCSI-hard drives. This is especially bad since on SCSI-hard drives the 1024-cylinder limitation becomes visible only at a much higher capacity [normally 8 GB] than with (E)IDE-hard drives [504 MB to 2 GB].

The last two limitations are less severe with modern PCs:

Some recent BIOSses allow the access to further equipment, such as up to four EIDE-hard drives (with EIDE-controller) instead of two. Some really modern combinations of BIOS/SCSI host adapter will allow you to boot from a SCSI-hard drive when EIDE-disks are built into the machine. However these features are not widespread and standardized enough to specify general rules. However there is no way to get around the 1024-cylinder limitation at the moment!

The following procedure can be used to boot DOS/Windows95/Windows98 and Linux systems with LILO which must be installed into the Master Boot Record (MBR) for this purpose.

- First install the other system. If you have to repartition the drive : Pay attention that there is still some place so the Linux boot partition can reside below the 1024 cylinder boundary. Normally the Linux root-partition will go into that primary partition. However, if there is not much space below 1024 cylinders, a few megabytes are sufficient for this partition. These will contain /boot.

- Install Linux. Create the Linux boot partition. According to the size of the partition, use it either for / (root-partition) or only for /boot, when specifying the mount-points.

- If your Linux boot-partition is mounted as /boot : Copy the kernel(s), that you want to boot (usually vmlinuz in a new system) to /boot at the end of the installation.

- Prepare the installation of LILO into the MBR : Create the following /etc/LILO.conf. Of course you have to adapt it to the names of your partitions and kernels. Please read the comments ! If you want to use more than one kernel or further systems, you can of course add more sections for further boot partitions at the end of the file. Please also note that by default the system at the top will be booted if nothing else is specified at the LILO boot-prompt. If necessary rearrange the different sections.

  ```
  # LILO configuration file:   LILO in the MBR
  ```

```
# Start LILO global Section
boot=/dev/hda  # Where LILO will be installed to: MBR
               # Please adapt this: must be sda with SCSI instead of hda !!

# Not necessary, but recommended :
# Make a backup of the old MBR :
backup=/boot/MBR.hda.yymmdd  # backup file for the old MBR
                             # from date dd.mm.yy
                             # Please adapt this !

# Remove the comment signs if you want to have a message
# from LILO every time you boot the system.
# This message must be stored in
# /boot/message before :
# message=/boot/message       #  LILO's greeting

prompt

# Remove the comment signs if you want to provide security for
# the boot procedure.
# password = my_password  # Please adapt this !

timeout=100     # Wait for 10 s at the prompt warten, before the
                # default systes  is booted.
vga = normal    # normal text modus (80x25 characters)
# End LILO global section

# configure Linux boot partitions
image = /vmlinuz      #  Please adapt this : The kernel
root = /dev/hda3      #  Please adapt this : The Linux root partition !!
label = Linux
# configuring the Linux boot partition : end

# configuring DOS/Win/WinNT boot partition
other = /dev/hda1    #  Please adapt this :  start partition !!
  label = DOS        #  whatever you like
  table = /dev/hda   #  Please adapt this : must be sda with SCSI
                     #  instead of hda !!
```

```
# configuring DOS/Win/WinNT boot partition: end
```

- Install LILO into the MBR:

```
[root@eritrea]♯ /sbin/lilo
```

5 LINUX LAN'ing

The idea of networking is probably as old as telecommunications itself. A computer network is a collection of hosts that are able to communicate with each other, often by relying on the services of a number of dedicated hosts that relay data between the participants. Hosts are very often computers, but need not be; one can also think of X-terminal or intelligent printers as hosts. Communication is impossible without some sort of language or code. In computer networks, these languages are collectively referred to as *protocols*. The protocols used in computer networks are nothing but very strict rules for the exchange of messages between two or more hosts.

There are several types of computer networks based on different protocols. TCP/IP is the leading protocol among its several dozen competitors, and in this thesis, we will describe this leading protocol in detail.

5.1 TCP/IP networking

TCP/IP and UDP/IP are sets of protocols for the transfer of data across networks of computers. Two of these protocols underlie most of the other application protocols like, www, ftp, DNS ...

TCP Transport Control Protocol, a "reliable stream service" for the unreliable Internet network protocol.

UDP User Datagram Protocol, "unreliable datagram service" for quick and less overhead services.

IP Internet Protocol, which sets the basic rules for formatting packets of data to go out over a network. TCP rides on top of IP.

TCP/IP traces its origins to a research project funded by the United States DARPA (Defense Advanced Research Projects Agency) in 1969. This was an experimental network, the ARPANET, which was converted into an operational one in 1975, after it had proven to be a success.

In 1983, the new protocol suite TCP/IP was adopted as a standard, and all hosts on the network were required to use it. When ARPANET finally grew into the Internet, the use of TCP/IP had spread to networks beyond the Internet itself. Most notable are local area networks, but in the advent of fast digital telephone equipment, such as ISDN, it also has a promising future as a transport for dial-up networks.

In the remaining part of the thesis we will show how we can configure TCP/IP to connect hosts as a LAN. But first, lets see the most important components of TCP/IP and its services. This part gives you a brief background you need in order to understand what your setup requires, in the next chapter. For a detailed explanation of networking issues refer to [1,9].

5.1.1 IP Addressing and Naming

The addresses understood by the IP-networking protocol are 32-bit numbers. Every Network interface in the machine must be assigned a number unique to the networking environment. If you are running a local network that does not have communication traffic with other networks, you may assign these numbers according to RFC 1918. RFC 1918 provides a number of guidelines on how to select and allocate these IP addresses. There are two reasons why we should follow this guidelines: First, it will be easy to connect to the Internet when the need arises. Second, if the IP number allocation is done outside the RFC 1918 guidelines there might be confusion in the Network Address Translator (NAT). However, for sites on the Internet, numbers are assigned by a central authority, the Network Information Center, or NIC.

Depending on the size of the network, there are several classes of network addresses, defining different splits of IP-addresses. Class A, B, C for networks with bigger, medium and small number of hosts respectively. And other IP classes are D, E, F assigned for multicast and future uses. IP-networking is structured in an autonomous way, similar to that of postal service. This type of structure splits IP-addresses into *host* and *network* part. Hosts that belong to the same network(called *subnet*) have similar figures in the network part of their IP address. This number of bits that are interpreted as the subnet number is given as the so-called *subnet mask*, or *netmask*.

A subnet has few nodes that are directly connected to different other subnets. All other hosts can be accessed only through these nodes called *gateways*. A gateway is a host that is connected to two or more physical networks simultaneously and is configured to switch packets between them. Thus, a gateway is assigned one IP-address per network it is on. These addresses along with the corresponding netmask are tied to the interface the subnet is accessed through.

5.1.2 TCP/IP services

TCP/IP underlies various services for various kinds of message exchanges. The common services are:

1. **Traditional services:** like
   - FTP(file transfer protocol), to send or receive files from one computer to another computer.
   - TELNET(The network terminal protocol). to log in on any other computer on the network, starting a remote session by specifying the computer address to connect to.
   - Email. Sending messages to users on other computers.

2. **Client/Server model of network services:** Nowadays, the way in which networks are used has been changing. Now many computer networks has "server/client" model services. A server is a system that provides a specific service for the rest of the network, A client is another system that uses that service. Those "server/client" model services can all be provided within the framework of TCP/IP.

Some of those services, which we are going to deal with in detail, are: Network File Systems(NFS), Network Information Systems(NIS), remote execution, nameservers (or DNS), terminal servers, network-oriented window systems.

6 Network Configuration

In the previous section we have list all the things one needs to build a network and we have also briefly discussed the concepts behind TCP/IP networking. In this section we present the configuration of the network. It will be illustrated using four PCs, five Ethernet Cards, a hub, two modems and 10bT twisted wires. After considering the requirements made on the network, Linux Red Hat distribution was chosen as the operating system software. We refer to the reader to chapter three to see why we have chosen Linux as the operating system over other network operating systems, especially Microsoft Windows NT.

6.1 Configuring a Network Interface

Hooking up a small LAN of UNIX machines is easy. It simply requires an Ethernet adapter in each machine and the appropriate Ethernet cables(such as: 10bT twisted wire) plus a hub. In our case the stationary PC has two Ethernet cards: one is used to establish a Local Area Network with our there labtops, and the other Ethernet card is used to connect the LAN to the Internet. Unlike many Unix operating systems that their network devices have appearance in the /dev directory in Linux the network devices are created dynamically in software thus don't require device files to be present. It is automatically created by the device driver while it is initializing and has located your hardware. For example, the Ethernet device driver creates $eth0$, and $eth1$ interfaces sequentially for the stationary PCs and $eth0$ for each laptop PCs. After the device driver has created the interface which can be checked by issuing $ifconfig$ command, the network interface can be configured, which essentially means assigning appropriate address to a network devices and to setting appropriate value for other configurable parameters of a network devices. The program most commonly used to do this; is the $ifconfig$ command.

```
[root@eritrea]♯ ifconfig eth0 192.168.1.1 netmask 255.255.255.0 up
```

In this case, we are assigning IP address 192.168.1.1(refer to section 3.1.4 why we have chosen this address) to Ethernet interface '*eth*0' and a network mask of '255.255.255.0' the *up* option tells the interface that it should became active. To shutdown an interface, you can just call '*ifconfig eth*0 down'.

6.2 Configuring loop back interface

The " loop back" interface is a special type of interface that allows you to make a connection to yourself. By convention the IP address '127.0.0.1' has been assigned specifically for loop back.
loop back interface is usually configured by the standard initialization scripts. You cam make sure by doing

```
[root@eritrea]♯ ifconfig lo 127.0.0.1
[root@eritrea]♯ route add -host 127.0.0.1 lo
```

6.3 Routing

IP routing is the process by which a host with multiple network connections decides where to deliver IP packets it has received. To accomplish this each host keeps a special list of routing rules, called *a routing table.* This table contains rows which typically contain at least three fields, the first is a destination address the second is the name of the interface to which the datagram to be routed and the third the optionally the IP address of another machine which will carry the datagram on its next step through the network. The table can be seen using one of the following command.

```
[root@eritrea]♯ cat /proc/net/route
[root@eritrea]♯ /sbin/route -n, or
[root@eritrea]♯ netstat -r
```

A special command called *route* can be used to manipulate the routing table it takes command line arguments and converts them into kernel system calls that request the kernel to add, delete, or, modify entries in the routing table. In the on going example we have one network segment. We use a class C network with an address of 192.168.1.0. We name one of the cards in the stationary PC as (*eritrea*) with IP address 192.168.1.1, which it will serve as router. And the other two PCs are assigned an IP address of 192.168.1.2, and 192.168.1.3 respectively. The first step is to configure the interfaces as described earlier.

[root@eritrea]♯ ifconfig eth0 192.168.1.2 netmask 255.255.255.0 up

You now need to add an entry into the routing table to tell the kernel that datagrams for all hosts with address that match 192.168.1.* should be sent to the Ethernet device.

[root@eritrea]♯ route add -net 192.168.1.0 netmask 255.255.255.0 eth0

This route will enable you to establish IP connections with all of the hosts on your Ethernet system and are other IP hosts that are not in your Ethernet segment will use the default route. The idea of default route is simply to enable you say 'every thing else should go here'.

[root@eritrea]♯ route add default gw 192.168.1.1 eth0

7 NFS Configuration

NFS is a product of Sun Microsystems that permits users to execute files without knowing the locations of these files. They may be local or remote in respect to the user. Users can create, read, or remove a directory .Files themselves can be written to or deleted. NFS provide a distributed file system that permits a user to capitalize on access capabilities beyond their local file system.

The idea behind NFS is having one copy of a file on one server that all users on a network can access it. The consequence of this is that software

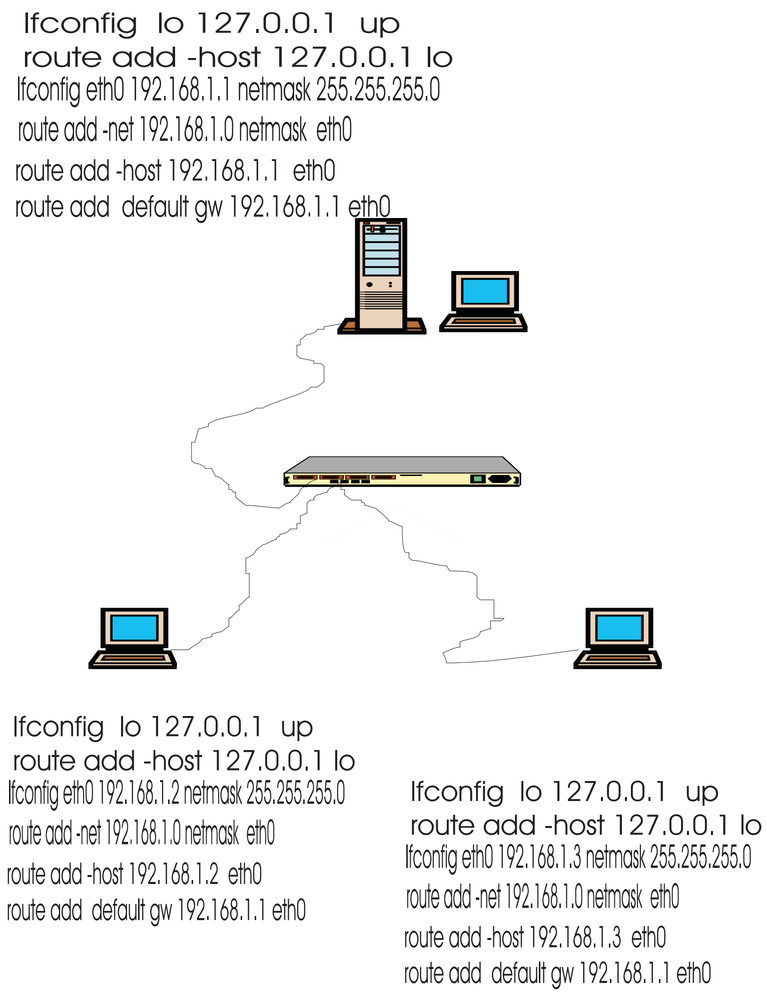


Figure 1: all commands

(and updates) can be installed on one server and not on multiple hosts in a networked environment. NFS is based on a client/server model. However, with NFS, a single NFS server can function to serve the requests of many clients.

7.1 Setting up a NFS server

A good starting point is to make sure the connection is up between the machines that are going to be used as server and clients. You should be able to telnet back and forth between the machines. In order to set up NFS we need to set up a number of programs.

7.1.1 The portmapper

The portmapper on Redhat Linux is called *portmap* and it is found in the */usr/sbin* directory. It manages RPC connections, which are used by protocol such as NFS and NIS. The portmap server must be running on machine which act as a servers for protocols which make use of the RPC mechanism. You can start it manually, but it will need to be started every time you boot your machine, so you need to include in the *rc* script(*/etc/rc.d/init/d*).

7.1.2 Mountd and nfsd

All the file system we want to make available on a client machine say we want file system */home* which is on the machine eritrea to be available to the machine called massawa, then I'd put this in */etc/exports* on eritrea.
/home *massawa(rw)*
The above line gives massawa read and write access to /home. There are other options we refer the reader to man page of exports for detail. You should always run exportfs after editing */etc/experts*. Now we can start *rpc.mountd* and *rpc.nfsd*. The command *rpcinfo -p* can be used to check if *rpc.mountd* and *rpc.nfsd* are running properly

[root@eritrea]♯ rpcinfo -p

should show something like:

```
program vers proto   port
   100000    2   tcp    111  rpcbind
   100000    2   udp    111  rpcbind
   100004    2   udp    934  ypserv
   100004    1   udp    934  ypserv
   100004    2   tcp    937  ypserv
   100004    1   tcp    937  ypserv
   100007    2   udp    948  ypbind
   100007    2   tcp    950  ypbind
   100021    1   udp   1026  nlockmgr
```

```
100021    3   udp   1026  nlockmgr
100021    1   tcp   1024  nlockmgr
100021    3   tcp   1024  nlockmgr
300019    1   tcp    773  amd
300019    1   udp    774  amd
100024    1   udp    757  status
100024    1   tcp    759  status
100011    1   udp    767  rquotad
100011    2   udp    767  rquotad
100005    1   udp    777  mountd
100005    1   tcp    779  mountd
100005    2   udp    782  mountd
100005    2   tcp    784  mountd
100005    3   udp    787  mountd
100005    3   tcp    789  mountd
100003    2   udp   2049  nfs
```

after checking that the portmapper reports the services, you can proceed to set up the NFS client.

7.2 Setting up a NFS client

Assuming that your kernel is compiled with NFS file system or available as a module you can, at root prompt issue appropriate mount command and the file system will appear. Continuing the example from the above viz. to make file system */home* of eritrea to be available on a machine massawa, we use the command:

[root@eritrea]♯ mount -o rsize=1024, wsize=1024, eritrea:/home /home

Now you can *cd* there, and *ls* in it, and look at the individual files. You can get rid of the file system by using the command:

[root@eritrea]♯ umount /home

You can include in the file */etc/fstab* to get the file system mount upon boot.

8 NIS Configuration

NIS stands for Network Information Service. Its purpose is to provide information, that has to be known throughout the network, that has to be known throughout the network, to all machines on the network. Information likely to be distributed by NIS is:

login names /passwords/home directories (*/etc/passwd*)

group information (*/etc/group*)

If for example, your password entry is recorded in the NIS passwd database, you will be able to login on all machines on the network which have the NIS client programs running.

NIS works as follows: within a network there must be at least one machine acting as a NIS server. You can also have multiple NIS server, where one act as master NIS server and the other as slave server. The importance of having NIS slave depends on the number of machines in your network and reliability of your network, and the NIS client connected to that server, will have access to find one that is up or faster.

8.1 Setting up NIS

In the network we must have at least on NIS server, we decided eritrea should be the NIS server and keren and massawa to be NIS clients.

The NIS server software is shipped with Linux, otherwise you can download it from several Linux-sites. Compile the software, if necessary, to generate the *ypserv* and *makedbm* programs. In order to run NIS server in eritrea as a master, we need to determine what files we wish to be available via NIS and then add or remove the appropriate entries to the "all" rule in */var/yp/Makefile*.

you should always look at the Makefile and edit the options at the beginning of the file. edit the file */etc/ypserv.conf* if you want to restrict access to ypserv.

```
#
# ypserv.conf In this file you can set certain options for the NIS server,
# and you can deny or restrict access to certain maps based
# on the originating host.
```

```
#
# See ypserv.conf(5) for a description of the syntax.
#

# Some options for ypserv. This things are all not needed, if
# you have a Linux net.

sunos_kludge: no
dns: no

# The following, when uncommented,  will give you shadow like passwords.
# Note that it will not work if you have slave NIS servers in your
# network that do not run the same server as you.

# Host                       : Map              : Security   : Passwd_mangle
#
# *                          : passwd.byname    : port       : yes
# *                          : passwd.byuid     : port       : yes

# Not everybody should see the shadow passwords, not secure, since
# under MSDOG everbody is root and can access ports < 1024 !!!
*       : shadow.byname    : port       : yes

# If you comment out the next rule, ypserv and rpc.ypxfrd will
# look for YP_SECURE and YP_AUTHDES in the maps. This will make
# the security check a little bit slower, but you only have to
# change the keys on the master server, not the configuration files
# on each NIS server.
# If you have maps with YP_SECURE or YP_AUTHDES, you should create
# a rule for them above, that's much faster.
*                           : *                : none
```

make sure the portmap(portmap(8)) is running, and start the server. The command

[root@eritrea]♯ rpcinfo -u localhost ypserv

should output something like,

```
program vers proto   port
 100000    2   tcp    111  portmapper
 100000    2   udp    111  portmapper
 100007    2   udp    637  ypbind
 100007    2   tcp    639  ypbind
```

```
or
```

```
program vers proto   port
 100000    2   tcp    111  portmapper
 100000    2   udp    111  portmapper
 100007    2   udp    758  ypbind
 100007    1   udp    758  ypbind
 100007    2   tcp    761  ypbind
 100007    1   tcp    761  ypbind
```

Now generate the NIS(YP) database on the master, run

[root@eritrea]♯ /usr/lib/yp/ypinit -m

Now the server is up and running. If you need to update a map, run make in the */var/yp* directory on the NIS master. This will update a map if the source file is newer. If you want to restrict access for users to your NIS server, you can edit the */var/yp/Makefile* and set NIS to use another source password file typically */etc/yp*. Note that if you do this the normal tools to administer the password file such as passwd, chfn, adduser will not work any more.

8.2 On the clients

Once you obtained the software, *ypbind* and $yp-tool$, follow the instruction that come with it to compile it. A suitable place for the *ypbind* daemon to install is the directory */usr/sbin*, you should do this as a root and the other binaries (*ypwhich*, *ypcat*, ...) should go in a directory accessible by all users, normally */usr/bin*. *ypbind* have a configuration file */etc/yp.conf* where we can hard code a NIS server there

```
# /etc/yp.conf - ypbind configuration file
# Valid entries are
#
#domain NISDOMAIN server HOSTNAME
# Use server HOSTNAME for the domain NISDOMAIN.
#
#domain NISDOMAIN broadcast
# Use  broadcast  on  the local net for domain NISDOMAIN
#
#ypserver HOSTNAME
# Use server HOSTNAME for the  local  domain.  The
# IP-address of server must be listed in /etc/hosts.
#
```

We do the following to test *ypbind* before incorporating in the start up files.

- Make sure you have your YP-domain name set. If it is not set then issue the command:

  /bin/domainname nis.domain

- Start up "*/usr/sbin/portmap*" if it is not already running.

- Create the directory "*/var/yp*" if it does not exist.

- Start up "*/usr/sbin/ypbind*".

Use the command "$rpcinfo - plocalhost$" to check if ypbind was able to register its service with the portmapper.

The output should look like:

```
program vers proto   port
 100000    2   tcp    111  portmapper
 100000    2   udp    111  portmapper
 100007    2   udp    637  ypbind
 100007    2   tcp    639  ypbind
```

```
or
```

```
program vers proto   port
 100000    2   tcp    111  portmapper
 100000    2   udp    111  portmapper
 100007    2   udp    758  ypbind
 100007    1   udp    758  ypbind
 100007    2   tcp    761  ypbind
 100007    1   tcp    761  ypbind
```

Depending on the ypbind version you are using. You may also run "$rpcinfo - ulocalhostypbind$". This command should produce something like:

```
program 100007 version 2 ready and waiting
```

```
or
```

```
program 100007 version 1 ready and waiting
program 100007 version 2 ready and waiting
```

At this point you should be able to use NIS client programs like ypcat, etc... For example, "ypcat passwd.byname" will give you the entire NIS password database.

IMPORTANT: If you skipped the test procedure then make sure you have set the domain name, and created the directory */var/yp* This directory MUST exist for ypbind to start up successfully. the command /bin/domainname can be used to check if the ypdomain name is set.

If the test worked you may now want to change your startup files so that ypbind will be started at boot time and your system will act as a NIS client. Make sure that the domainname will be set before you start ypbind. Reboot the machine and watch the boot messages to see if ypbind is actually started.

9 Automount

Automouting is the process where mounting (and unmounting) of certain filesystems is done automatically by a daemon. If the filesystem is unmounted, and a user attempts to access it, it will be automatically (re)mounted. This is especially useful in large networked environments and for crossmounting filesystems between a few machines (especially ones which are not always online). It may also be very useful for removable devices, or a few other uses, such as easy switching between a forced-on ASCII conversion mount of a dos filesystem(fs) and a forced-off ASCII conversion mount of the same DOS fs.

There are two types of automounters in Linux; AMD and autofs. We shall only discribe Autofs which is a newer system assisted by the kernel, meaning that the kernel's filesystem code knows where the automount mount points are on an otherwise normal underlying fs, and the automount program takes it from there.

9.1 Installation

Because autofs is implemented in kernel-space, your kernel must have support compiled in. In kernel 2.0.xx it is an experimental option, but appears to be quite stable. In 2.1.xx (and presumably 2.2.xx) it is not experimental.

The automount program and its configuration files are also necessary; using the rpms (from RedHat, as part of the install) is a great way to go. The automount program should be started by an "rc" script under the */etc/rc.d/init.d* directory. The *rpm* installs this, but you will need to make sure it gets started, either by linking it from your "rc?.d" directory, using Redhat's control-panel, or on another distribution by getting the thing started anyway you care to. Non-rpm distros will have to do whatever's applicable to their system. And don't look too hard at what the "rc" script does.

9.2 Configuration

Installing the RPM's will get you to this point easily enough, but here's the part you might not be sure about if you haven't done this before.

There are two files in /etc, one called auto.master and one called auto.misc. My auto.master looks like this:

```
/auto   /etc/auto.misc  --timeout 60
```

The first entry is not the mount point. It's where the set of mount points (found in the second entry) are going to be. The third option says that the mounted filesystems can try to unmount themselves 60 seconds after use. They can't unmount if being used, of course.

Auto.misc is a "map file". Multiple map files can be defined in auto.master. My auto.misc looks like this:

```
kernel          -ro,soft,intr          ftp.kernel.org:/pub/linux
cd              -fstype=iso9660,ro     :/dev/cdrom
zip             -fstype=auto           :/dev/hdd4
floppy          -fstype=vfat           :/dev/fd0
```

The first column (the "key") is the mount point. In this case it would be */auto/floppy* or whatever. The middle set are the options; read the *mount* manpage for details on this. And the last column specifies where the fs comes from. The "kernel" entry is supposed to be an NFS mount. The colon sign ":" on all the other lines means its a local device.

Automount can also be configured with NIS (yp) to serve users log on with their PCs via the NIS server, with a home directory automatically mounted from the NIS server.

The following hypothetical example can serve as an illustration: Assume that we have directories scattered around on many different UNIX boxes. The user's home directories are on their local machines in /home/username. They are automounted to /users/hostname/username. The user's home directory as specified in the NIS exported passwd is also /users/hostname/username. The home directories are automounted on the local machine even if the real directory resides on the local machine. This is important.

We are assuming that you have configured the NIS client on your Linux box so that you can do an NIS pull using *ypcat*. You can start the NIS client in one of your system startup files:

[root@eritrea]♯ domainname yourNISdomain
[root@eritrea]♯ ypbind

At this point, you should be able to read the **NIS** map for the passwd group and auto.users (or whatever your user NIS map is named) using ypcat:

[root@keren]♯ ypcat -k passwd
[root@keren]♯ ypcat -k group
[root@keren]♯ ypcat -k auto.users

Assuming that your **NIS** home directory map is called auto.users (If this is not the case, substitute the proper map name), the output from $ypcat - kauto.users$ is:

```
* -rsize=8192, wsize=8192, rw, intr &:/home
```

The directory underwhich the mounts are made must exist:

[root@keren]♯ mkdir /users

Then you start the automount daemon provided by autofs and tell it to use the **NIS** map for directories under $/users$. You should put this in one of your system startup scripts called $/usr/sbin/automount/usersypauto.users$

Of course this must be run after NIS client is running. Now, if you have a user named yordanos and her home directory resides on a machine called keren, the passwd map has yordanos' home directory as /users/keren/yordanos. This directory is really at keren:/home. you can *cd* to yordanos' home directory from your Linux box:

[root@keren]♯ cd yordanos

```
[yordanos@keren /users/keren/yordanos]$
```

[yordanos@keren /users/keren/yordanos]$ mount

```
...
automount(pid109) on /users type autofs
(rw,fd=5,pgrp=109,minproto=2,maxproto=3)
...
keren:/home on /users/keren type nfs
(rw, rsize=8192, wsize=8192, intr, addr=10.122.164.111)
```

10 DNS Configuration

Domain Name System(DNS) is one of the main player in the Internet success. When a network grows from few hosts to hundreds of them, addressing becomes a non trivial problem. For the Internet, a network of millions of computers, the ability to address each host uniquely is a crucial thing that the network has to fulfill. To this end, a mechanism called Domain Name System is built.

DNS is a hierarchical domain-based naming scheme and a distributed database system for implementing this naming scheme[1,9]. The top of the hierarchy is labeled as "." (pronounced as 'root'). under "." there are a number of Top Level Domains(TLDs). Each domain is partitioned into subdomains and these are further partitioned, and so on. Fig 2 illustrates a portion of the Internet domain space.

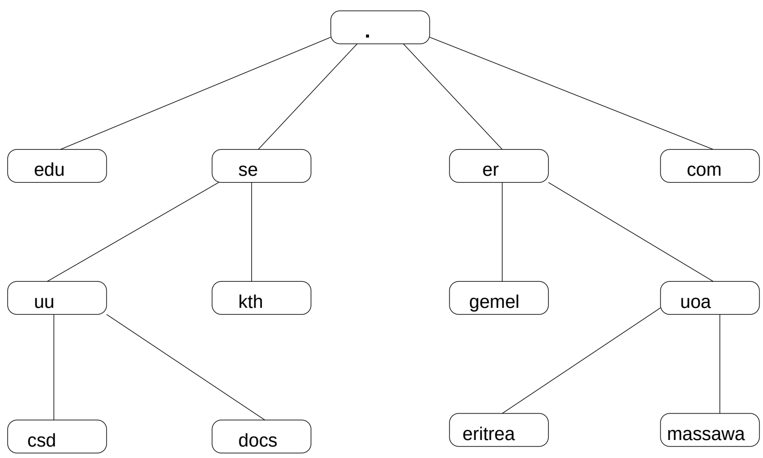


Figure 2: Domain Space Tree

This way a user can connect from a host in one place to another host in another place without knowing the actual IP-address of the peer. A network administrator can organize the network in a structured and organized way, preserving the uniqueness of his/her domain space.

In what follows we will create our own domain and explain how the DNS configuration process proceeds. For more information on nameserver and DNS domain space please refer to [1,9].

10.1 Creating your own domain

Now, we are going to make our own domain and define machines in it. We use *csd.uoa.er* to define the local area network of the Computer Science department at the University of Asmara in Eritrea. The configuration requires running named in the machine that serves as a DNS server called nameserver and editing six files, namely: */etc/names.conf, /etc/resolve.conf, /var/named/zone/0.0.127, /var/named/csd.uoa.er, /var/named/192.168.1.* *Named* is started with the command 'ndc start' and can be restarted by 'ndc restart' when ever you need to restart it after making change to the configuration. Once named is started the file */var/named/zone/*0.0.127 is created automatically and the default nameserver becomes the machine itself (localhost=127.0.0.1), and gets ready for any request. Requests, to translate host names to/from their IP-addresses, can be issued to the nameserver by invoking *nslookup*. The translation is done either by the machine itself, if the host we are looking for a host under the LAN where this nameserver is located, or by a neighbor nameserver if it is outside the LAN. The IP number of the internal machines and their corresponding given names are in */var/named/csd.uoa.er* and */var/named/*192.168.1 . The former file is for translating name to IP(called forward zone file), while the later is for translating IP to name(revers zone file). The format of these files is given below. A detailed syntax explanation of the format can be found in [5]. Recall, from section 4.6, that our network address is 192.168.1 and the host address for eritrea, keren, massawa are 1,2,3 respectively. *eritrea.csd.uoa.er* is the chosen machine to be nameserver.

;

```
; the forward zone file: /var/named/csd.uoa.er
;
@       IN      SOA     csd.uoa.er. root.csd.uoa.er. (
                        199609206       ; serial, todays date + todays serial >
                        8H              ; refresh, seconds
                        2H              ; retry, seconds
                        1W              ; expire, seconds
                        1D )            ; minimum, seconds
                NS      csd.uoa.er.
                MX      10 csd.uoa.er.  ; Primary Mail Exchanger

localhost       A       127.0.0.1

router          A       192.168.1.1

csd.uoa.er.     A       192.168.1.1

eritrea         A       192.168.1.1

www             CNAME   csd.uoa.er.
ftp             CNAME   csd.uoa.er.

@               TXT     "Computer Science Department, UoA"

;
;       Workstations
;
keren           A       192.168.1.3
                MX      10 csd.uoa.er.   ; Primary Mail Host
massawa         A       192.168.1.2
                MX      10 csd.uoa.er.   ; Primary Mail Host
```

www, ftp, router, eritrea are other aliased names for the same machine, which helps to use http://www.csd.uoa.er/ to access web pages that sites in the server.

```
;
; reverse zone file: /var/named/192.168.1
```

```
;
@               IN      SOA     csd.uoa.er. root.csd.uoa.er. (
                                199609206       ; Serial
                                28800   ; Refresh
                                7200    ; Retry
                                604800  ; Expire
                                86400)  ; Minimum TTL
                        NS      csd.uoa.er.
;
;       Servers
;
1       PTR     eritrea.csd.uoa.er.
;
;       Workstations
;
3     PTR     keren.csd.uoa.er.
2     PTR     massawa.csd.uoa.er.
```

The */etc/name.conf* file is used to organize this information and to tell named which files are responsible for which tasks. The zones $0.0.127.in-addr.arpa$ and *1.168.192.in-addr.arpa* are reverse zones for the localhost and the internal hosts respectively. Observe that they are named with the network address reversed. The address 1.168.192 is the inverse of our network address 192.168.1. A responsible file name is associated with each zone. For example zone/csd.uoa.er is responsible for translating host names that end with csd.uoa.er. As a matter of fact, it could also be responsible for host names that does not have any extensions besides their name, if the search path in file(/etc/resolve.conf)is csd.uoa.er.

```
// Boot file for csd.uoa.er nameserver

options {
        directory "/var/named";
};

zone "." {
        type hint;
```

```
        file "root.hints";
};

zone "0.0.127.in-addr.arpa" {
        type master;
        file "zone/127.0.0";
};

zone "csd.uoa.er" {
        type master;
        file "zone/csd.uoa.er";
};

zone "1.168.192.in-addr.arpa" {
 type master;
        file "zone/192.168.1";
};
```

The */etc/resolve.conf* looks like this. It is recommended to put more nameservers in case some nameservers are down or not working properly. By making the search option 'search csd.uoa.er' will make possible to use only *keren* instead of *keren.csd.uoa.er*.

```
# /etc/resolve.conf
domain csd.uoa.er
search csd.uoa.er csd.uu.se
nameserver 127.0.0.1
nameserver 130.238.12.42
nameserver 130.238.12.182
```

10.2 Using the local nameserver

Let us check how the named will operate under our configuration, To start or restart named use 'ndc start' or 'ndc restart' then invoke nslookup. In the nslookup prompt enter hostnames and IP addresses as follows. A correctly configured nameserver should out put similar.

[root@eritrea]♯ nslookup

and the output looks like:

```
Default Server:  localhost
Address:  127.0.0.1

> keren
Server:  localhost
Address:  127.0.0.1

Name:    keren.csd.uoa.er
Address:  192.168.1.3

> www.asmarino.com
Server:  localhost
Address:  127.0.0.1

Non-authoritative answer:
Name:    asmarino.com
Address:  209.125.130.198
Aliases:  www.asmarino.com

> 192.168.1.2
Server:  localhost
Address:  127.0.0.1

Name:    massawa.csd.uoa.er
Address:  192.168.1.2

> exit
```

In the workstations the only thing we have to do is. To modify the file /var/resolv.conf, to look like this.

```
//resolv.conf file for a workstation computer
search csd.uoa.er
nameserver 192.168.1.1
```

and the output of nslookup in the workstation should look like.

[root@keren]♯ nslookup

```
Default Server:  eritrea.csd.uoa.er
Address:  192.168.1.1

> massawa
Server:  eritrea.csd.uoa.er
Address:  192.168.1.1

Name:    massawa.csd.uoa.er
Address:  192.168.1.2

> exit
```

11 Dial-Up Networking

Dial-Up networking is another way of establishing connectivity to your server. With just a modem and a telephone line a remote computer can be part of the network. The serial line protocol known as Point-to-Point Protocol is the the cheapest and easiest way of establishing a dial-up network. In this chapter we guide you on configuring the PPP connection scripts for both client and server.

11.1 PPP Dial-out

Setting up your Linux box as a PPP client can be done in four steps as follows:

1. Editing the ppp-on, ppp-off and ppp-on-dialer. These ppp files are usually in the $/usr/doc/ppp{-}2.3.3/scripts/$ directory. It is a good idea to copy them in a new *ppp* directory (eg. $/etc/ppp/$). Next modify the ppp-on and supply the appropriate server telephone number, account name and password. If you have a stationary IP assigned to your PC give the local and remote IP numbers and the netmask. But if the

connection is dynamic-IP-number-allocation, set them to 0.0.0.0 . In the pppd arguments: the speed of your modem and the connection device has to be set correctly. the 'noipdefault' and 'defaultroute' are important for dynamic IP assignment and in making the dial-up server a default router. They have to be included in the argument only if they are needed. Remember that the *pppd* command has to be accessible by users.

```
# the ppp-on file
#!/bin/sh
#
# Script to initiate a ppp connection. This is the first part of the
# pair of scripts. This is not a secure pair of scripts as the codes
# are visible with the 'ps' command.  However, it is simple.
#
# These are the parameters. Change as needed.
TELEPHONE=004800707     # The telephone number for the connection
ACCOUNT=bice4577        # The account name for logon
PASSWORD=jupg]Van       # The password for this account
LOCAL_IP=0.0.0.0        # Local IP address if known provide the number.
                        # 0.0.0.0 for dynamic
REMOTE_IP=0.0.0.0       # Remote IP address if desired. Normally 0.0.0.0
NETMASK=255.255.255.0   # The proper netmask if needed
#
# Export them so that they will be available at 'ppp-on-dialer' time.
export TELEPHONE ACCOUNT PASSWORD
#
# This is the location of the script which dials the phone and logs
# in.  Please use the absolute file name as the $PATH variable is not
# used on the connect option.  (To do so on a 'root' account would be
# a security hole so don't ask.)
#
DIALER_SCRIPT=/etc/ppp/ppp-on-dialer
#
# Initiate the connection
#
# I put most of the common options on this command. Please, don't
# forget the 'lock' option or some programs such as mgetty will not
```

```
# work. The asyncmap and escape will permit the PPP link to work with
# a telnet or rlogin connection. You are welcome to make any changes
# as desired. Don't use the 'defaultroute' option if you currently
# have a default route to an ethernet gateway.
#
exec /usr/sbin/pppd debug lock modem crtscts /dev/ttyS1 115200 \
        asyncmap 20A0000 escape FF kdebug 0 $LOCAL_IP:$REMOTE_IP \
        noipdefault netmask $NETMASK defaultroute connect $DIALER_SCRIPT
```

and the ppp-on-dialer script looks like:

```
#!/bin/sh
#
# This is part 2 of the ppp-on script. It will perform the connection
# protocol for the desired connection.
#
exec chat -v                                                \
        TIMEOUT         3                                   \
        ABORT           '\nBUSY\r'                          \
        ABORT           '\nNO ANSWER\r'                     \
        ABORT           '\nRINGING\r\n\r\nRINGING\r'        \
        ''              \rAT                                \
        'OK-+++\c-OK'   ATH0                                \
        TIMEOUT         30                                  \
        OK              ATDT$TELEPHONE                      \
        CONNECT         ''                                  \
        name:--name:    $ACCOUNT                            \
        assword:        $PASSWORD                           \
        >               ppp
```

sometimes dial-up servers use different login processes. By default the ppp-on-dialer expects Login: and Password: before sending the login name and the password. If it is different (eg Username: instead of Login:) you need to edit this file and modify it as needed.

2. Editing your host files. Make sure that your /etc/hosts file looks something like:

```
127.0.0.1       localhost               localhost
0.0.0.0         keren.student.uu.se     keren
```

3. Modify the /etc/resolv.conf file to set the domainname and nameservers. Enter what is correct for your connection provider or ISP. For example, Uppsala University dial up service providers have a domain name and nameservers as shown below.

   ```
   #/etc/resolv.conf
   search student.uu.se
   domain student.uu.se
   nameserver 127.17.1.2
   nameserver 127.17.1.5
   nameserver 192.168.255.10
   ```

4. Finally, make ppp-on and ppp-off runnable and start ppp-on. After a minute or two you will have your interface and your routing table configured. Issue /sbin/ifconfig and /sbin/route to check whether the script did it all correctly. For debugging purpose view the /var/logs/message file.

11.2 PPP Dial-in

To provide the above dial-out services for home and business dialers a reliable PPP server should be located and stand alert to respond calls. The process a server goes through to establish a PPP link to users includes:

1. Receiving calls.
2. Verifying user account name and password pair.
3. Starting PPP link and configuring the client dynamicaly.

The Demon process that deals with all this procedures is called *mgetty* (which we are going to use) or the non-smart one called *getty*. The first step in making *mgetty* operational is compiling it for 'AUTO-PPP' which can be done by editing the Makefile in mgetty source code (in /usr/src/redhat/SOURCE/mgetty/ for Red hat Linux). The CFLAGS flag should look like 'CFLAGS=-O2 -Wall -pipe -DAUTO_PPP' instead of 'CFLAGS=-O2 -Wall -pipe'. Next, copy policy.h-dist to policy.h, Compile and install mgetty. Modify /etc/inittab to run mgetty on your serial ports. Add one line per modem you have connected and operational, otherwise mgetty will produce unnecessary errors. On our system the *mgetty* serial setup (in /etc/inittab) is

```
s1:2345:respawn:/sbin/mgetty -D /dev/ttyS1
```

Note that any path used in this document is specific to our system, for other systems specially for non-Red Hat systems the path may be different. Please pay attention to provide the right path for your own system.

For PPP data transfer (no voice or fax) mgetty.config should primely include the following lines.

```
#the modem is located in /dev/ttyS1, it could be another port
port ttyS1
#debug level, log saved in /var/log/mgetty.ttyS1
debug 6
#dont do voice/fax
data-only y
```

and in the login.config files for Auto-PPP, you have to:

1. Comment out the Taylor UUCP line
2. Comment out the /FIDO/ line
3. Add the line:

   ```
   /AutoPPP/ - ppp /usr/sbin/pppd auth -chap +pap login modem crtscts lock
   ```

With what we have done so far a user can call the server and log to the system in a terminal mode (example: using minicom). For enabling the server, to allocate IP-number dynamically, configuring the client's network interface and adding to its routing table, we have to do some more. Here is the procedures.

1. Making pppd accessible for remote users

   ```
   chmod u+s /usr/sbin/pppd
   ```

2. Creating a ppp short cut

   ```
   alias ppp="exec /usr/sbin/pppd -detach"
   ```

   or you can create a bash file that have the same purpose

3. Creating /etc/ppp/options.ttyS1 (ttyS1 is the port the modem connected) This holds the server and remote IP numbers respectively.

   ```
   192.168.1.3:192.168.1.10
   ```

   The first IP number is the PPP server IP number and the second one is an IP number to be assigned dynamically for the dialing client. As many files (of this sort) as the modems are required, if you have more than one modems in the rest of the *ttyS* ports.

4. The server should really have its own nameserver or the nameserver to be used should be easily locate-able and accessible.

5. In /.ppprc the individual user can have his own options that is applicable only for himself. This can also be used to restrict or give privileges for this user.

More information can be read from the *mgetty* and *pppd* man pages and from [5].

12 Wireless Networking

A wireless network is a flexible data communication system implemented either as a wireless LANs(WLAN) or as a point-to-point connection among distant LANs. Using electromagnetic waves (radio or infrared) wireless networks transmit or receive data over the air, and it runs in the range of 1-10Mbps. It provides all the functionality of a wired network, but without physical constraint of the wired network itself. In other words, a wireless network is simply a communication technology that combines data connectivity with user mobility. Users are taking advantage of this new technology by accessing shared information without looking for a place to plug in, thus, increasing scalability and not having to pull cables through walls and ceilings. Another attractive feature, that has been learned recently is, the reducing cost of wireless network hardware makes it economically preferable than wired network [6,10].

A typical WLAN configuration is implemented as an extension to, or as an alternative for, a wired LAN within a building or campus. Based on the thickness of walls and other barriers within the building, a spread spectrum adapter runs from 100 to 500m long. While a Point-to-Point WLAN such as LAN-LAN bridging runs longer distance, provided that the two antennas can see each other clearly. Lucent Technology (a wireless hardware manufacturer) claims, of having an adapter that runs on a distance up to 30km. Off course, the distance that an outdoor wireless network can travel is affected by barrier trees, winds and rain, consequently, it is recommended to use a LAN-LAN bridging configuration to nearby buildings.

12.1 Benefits of Wireless Network

Global connectivity in developing countries is in an early stage due to installation costs, insufficient basic infrastructures, low quality of available telecommunication services and limited financial support. The application of wireless technology is an effective choice to overcome some of these problems, at least within smaller areas. This is true even if transmission speeds are lower than the ones achieved by wired networks. Within university campus, it is easier to install a radio link system than to place cables or expensive optical fibers in the ground. Furthermore, radio installations are easier to protect from external natural phenomena such as floods, landslides, etc. At first glance, wireless LANs looks more expensive than wired LANs, but in the long term they have been noticed of having lower maintenance costs and are relatively easy to configure. The use of Linux and standard radio-communication technologies, in conjunction with the many Linux software applications, makes this task even easier. With this idea in mind, this thesis discusses on wireless configuration and how it works. At the end of the chapter an example is provided. 'Wire less network in Africa' an experience on establishing wireless network using Linux at a Nigerian University [6].

12.2 How Wireless LAN works

Wireless LANs uses electromagnetic waves to communicate information from one point to another without relying on any physical connection. Most wireless LAN systems often built on spread-spectrum technology for on-air communication with the peer host. The so-called "spread-spectrum" is a digital coding method in which the signal is transformed or spread so that it cannot be received by any receiver except the designated one that understands the transmitted signal code[6]. It minimizes interference to other users and normally does not require an operation license in the ISM (International Scientific and Medical Band), depending on the regulation adopted by the country. Radio waves are often referred as radio carriers because they simply perform the function of delivering energy to a remote receiver. The data being transmitted is superimposed on the radio carrier so that it can be accurately extracted at the receiving end. This is generally referred to as modulation of the carrier by the information being transmitted. Once data is superimposed (modulated) onto the radio carrier, the radio signal occupies more than a single frequency, since the frequency or bit rate of the modulating information adds to the carrier.

Multiple radio carriers can exist in the same space at the same time without interfering with each other if the radio waves are transmitted on different radio frequencies. To extract data, a radio receiver or augment networks without installing or moving wires. Wireless LANs tunes in (or selects) one radio frequency while rejecting all other radio signals on different frequencies.

In a typical WLAN configuration, a transmitter/receiver (transceiver) device, called an access point, connects to the wired network from a fixed location using standard Ethernet cable. At a minimum, the access point receives, buffers, and transmits data between the WLAN and the wired network infrastructure. A single access point can support a small group of users and can function within a range of less than one hundred to several hundred feet. The access point (or the antenna attached to the access point) is usually mounted high but may be mounted essentially anywhere that is practical as long as the desired radio coverage is obtained [10].

Wireless LANs should not be confused with wireless metropolitan-area network (WMANs), packet radio often used for law enforcement or utility applications, or with wireless wide-are networks (WWANs), wide-area data transmission over cellular or packet radio [10]. These systems involve costly infrastructures and provide much lower data rates compared to WLANs.

12.3 WLAN Configuration

End users access the WLAN through wireless-LAN adapters, which are implemented as PC cards in notebook computers, ISA or PCI cards in desktop computers, or integrated within hand-held computers. WLAN adapters provide an interface between the client network operating system (NOS) and the airwaves (via an antenna). The nature of the wireless connection is transparent to the NOS. Meaning, except for the hardware and driver installation, which can be done with the help of installation manual shipped together with the hardware, the network configurations remains the same as was shown in the previous chapters.

1. The simplest WLAN configuration is an independent LAN that connects a set of PCs with wireless adapters. Any time two or more wireless adapters are within range of each other, they can set up an independent network. These on-demand networks typically require no administration or pre-configuration. Access points can extend the range of existing LANs by acting as a repeater, effectively doubling the distance between wireless PCs.

2. In infrastructure WLANs, see figure 3, multiple access points link the WLAN to the wired network and allow users to efficiently share network resources. The access points not only provide communication with the wired network but also mediate wireless network traffic in the immediate neighborhood.

3. Multiple access points see figure 4 can provide wireless coverage for an entire building or campus. Microcells and Roaming Wireless communication is limited by how far signals carry for given power output. WLANs use cells, called microcells, similar to the cellular telephone

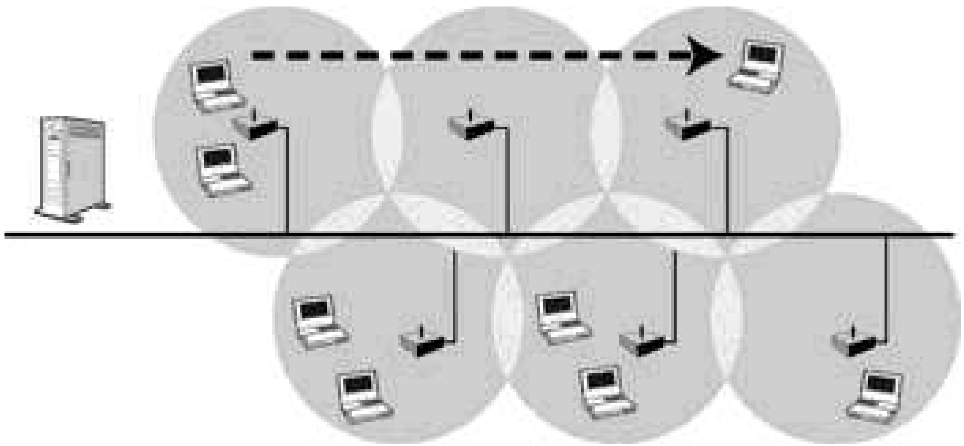


Figure 3: Infrastructure WLAN

system to extend the range of wireless connectivity. At any point in time, a mobile PC equipped with a WLAN adapter is associated with a single access point and its microcell, or area of coverage. Individual microcells overlap to allow continuous communication within wired network. They handle low-power signals and "hand off" users as they roam through a given geographical area.

12.4 Case Study: A Wireless Network at a Nigerian University

The "Programme of Training and System Development on Networking and Radio Communications" at Abdus Salam ICTP, Trieste, Italy, and the Obafemi Awolowo University (OAU) of Ile-Ife in Nigeria establish a Pilot Educational and Research Computer Network at OAU. Such a network, based principally on personal computers running Linux, today provides connectivity between several faculties and departments on the campus.

As shown in Figure 5, the wireless campus network (OAUNET) is based on a radio system; it initially involved three separate buildings and had the capacity to be rapidly extended to other university structures. The wireless link uses a spread-spectrum, direct-sequence technique providing data transmission at 2Mbps.

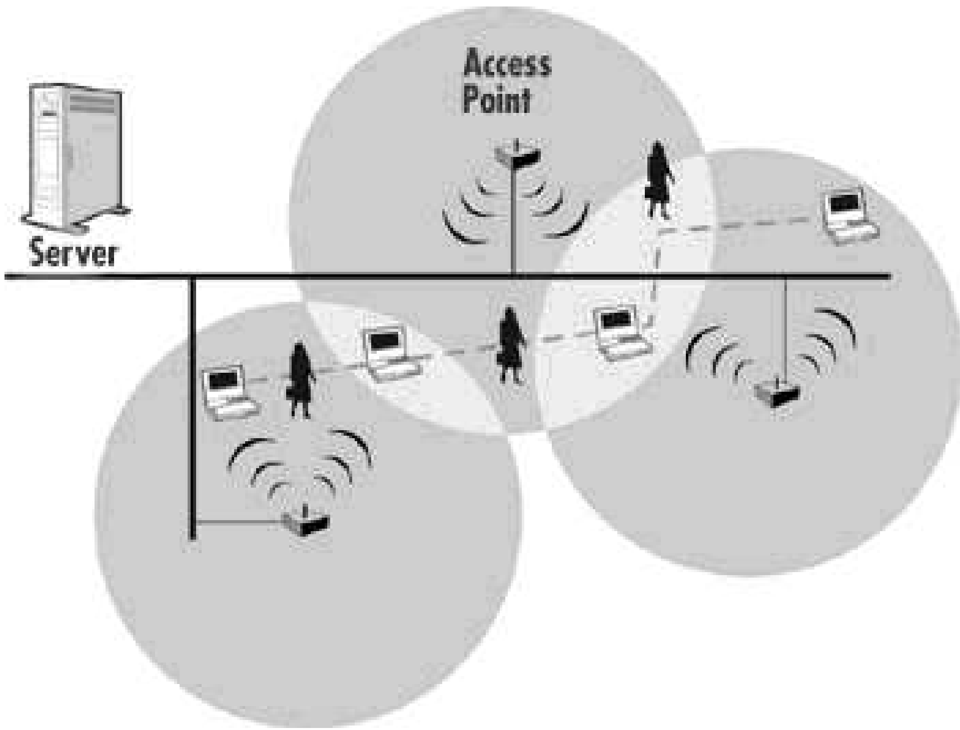


Figure 4: Handing off the WLAN Connection Between Access Points

Inside each building, an Ethernet 10-BASE 2 cabling structure is installed in order to keep the initial costs as low as possible (i.e., no hubs, less cable) and to ensure the local availability of spares (BNC), etc. In each of these buildings, a Linux PC acts as "faculty server" and provides mail services for the local users and does routing to the backbone. This strategy has been selected to keep the user-generated traffic local and reduce the access to the main backbone. All services are TCP/IP-based to keep the system as standard as possible with Internet protocols, avoiding future modifications when full connectivity might be provided to the university.

The academic network gateway and the main mail host are located at the Department of Computer Science at the university. Due to national regulations and the lack of a permanent connection to the Internet, the gateway is linked on a dial-up base (uucp) using an international direct-dialing line to the ICTP computer network in Trieste, Italy. Software was developed by OAU staff with some assistance from ICTP to refine the basic

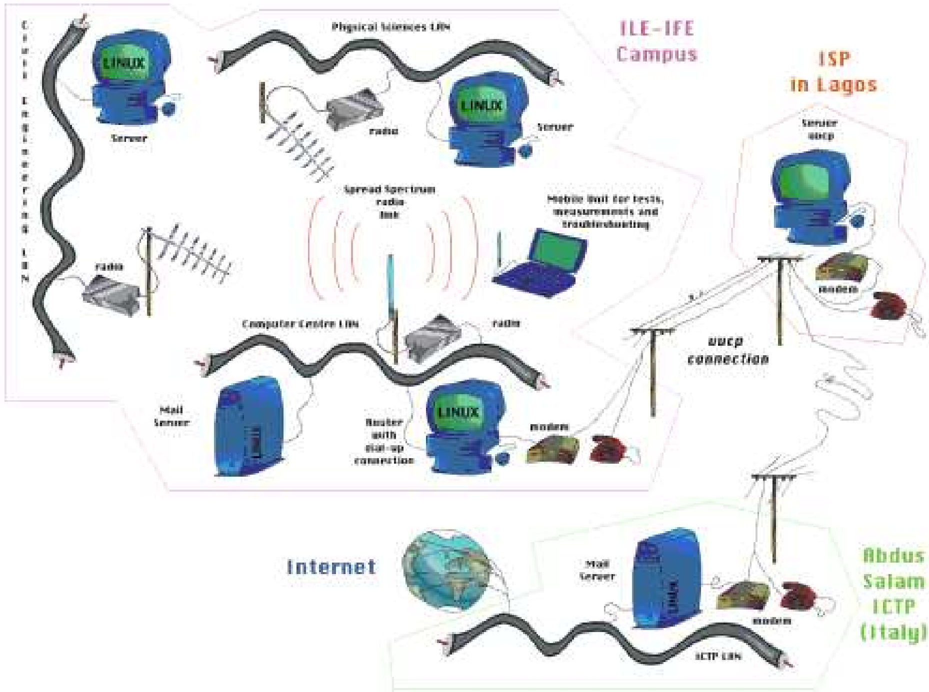


Figure 5: Wireless Network at OAU

uucp mail transfer: a custom sendmail delivery program batches mail in intermediate-sized, BSMTP (batch simple mail transfer protocol) formatted files; these files are compressed as much as possible before being transferred over uucp. To cope with telephone line instabilities, a uucp relay was placed in Lagos; the uucp configuration takes care of selecting the path either directly to Trieste or through the Lagos relay, automatically choosing the one that works.

Campus-wide services such as e-mail, FTP, WWW and NFS are available today within the OAUNET. As connection to the rest of the world is done on a dial-up line (since 1996), only e-mail exchange is provided freely to local users. There is no limitation on the amount and size of the information being transferred on campus, and it has been in operation since June 1996 without any major problems and has proven to be highly beneficial for the academic life at the university. The next phase of the wireless network requires the installation of other Internet services on the OAUNET. For example, arrangements are being made to provide connectivity to the library and other faculties such as the faculty of Agriculture.

13 Network Security

For the first few decades of their existence, computer networks were primarily used by university researchers for sending email, and by corporate employees for sharing printers[1]. Under these conditions, security did not get a lot of attention. But now, as millions of ordinary citizens are using networks for banking, shopping, and filing their tax returns, network security is looming on the horizon as a potentially massive problem. The following section explains the achievement of network security on Linux. In the Internet paradigm, local network security is done by the network layer protocol called $firewalling$. A firewall is a set of related programs, located at a network gateway(fig. 6) server, that protects the resources of a private network from users of another network[11]. (The term also implies the security policy that is used with the programs.) An enterprise with an intranet that allows its workers access to the wider Internet installs a firewall to prevent outsiders from accessing its own private data resources and for controlling what outside resources its own users have access to. The services of firewall should not be confused with SSL and TSL security protocol. While firewall secures local communication; Secure Sockets Layer and Transport Layer Security (SSL/TSL) Protocol provides privacy between two communicating applications (a client and a server). Banking, online shopping and other transactions that demand encryption and decryption of data require SSL/TSL security .In this chapter we only deal with “protecting local networks”.

Basically, a firewall, working closely with a router program, and it has to different types:

1. filters all network packets to determine whether to forward them toward their destination or not, called *Filtering Firewalls*
2. Makes network requests on behalf of workstation users, called *Proxy servers*

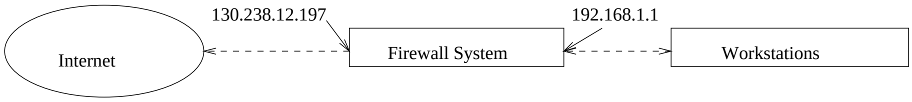


Figure 6: Firewall design

13.1 Packet Filtering Firewalls

Packet Filtering is the type of firewall built into the Linux kernel intended to keep the flames of Internet out of your private LAN. Or, to keep the members of your LAN pure and chased by denying them access the all the evil Internet temptations.

A filtering firewall works at the network level. Data is only allowed to leave the system if the firewall rules allow it. As packets arrive at the gateway(figure 6)they are filtered by their type, source address, destination address, and port information contained in each packet. Many network routers have the ability perform some firewall services. Filtering firewalls can be thought of as type of router. Because of this you need a deep understanding of IP packet structure to work with one. Because very little data is analyzed and logged filtering firewalls take less CPU and create less latency in your network.

13.2 Proxy Servers

Proxies are mostly used to control, or monitor, outbound traffic. Some application proxies cache the data requested. This lowers bandwidth requirements and decreases the access time for the next user how access the same data. It also gives unquestionable evidence of what was transferred. There are two types of proxy servers.

13.2.1 Application Proxy

A typical application proxy server is an Internet data caching. Software like *Squid* does this by accepting requests for objects that people want to download and handling their requests in their place. In other words, if a person wants to download a web page, (s)he asks this proxy to get the page for him/her. Then connects to the remote server and requests the page. It then transparently streams the data through itself to the client machine, but at the same time keeps a copy. The next time someone wants that page, the proxy server simply reads it off disk, transferring the data to the client machine almost immediately.

Because proxy servers are handling all the communications, they can log every thing they/you do. For HTTP (web) proxies this would include very URL they/you see. For FTP proxies this includes every file you download. They can even filter out "inappropriate" words from the sites you visit or scan for viruses.

Application proxy servers can authenticate users. Before a connection to the outside is made, the server can ask the user to login first. To a web user this would make every site look like it required a login.

13.2.2 SOCKS Proxy

A SOCKS server is a lot like an old switch board. It simply cross wires your connect through the system to another outside connection[5]. Most SOCKS server only work with TCP type connections. And like filtering firewalls they don't provide for user authentication. They can however record where each user connected to.

13.3 How Packets Traverse the Filters

The kernel starts with three lists of rules; these lists are called *firewall chains* or just *chains*. The three default chains are *input*, *output* and *forward*. When a packet comes in (say, through the Ethernet card) the kernel uses the input chain to decide its fate. If it survives that step, then the kernel decides where to send the packet next (this is called *routing*). If it is destined for another machine, it consults the forward chain. Finally, just before a packet is to go out, the kernel consults the output chain.

A chain is a checklist of rules. Each rule says 'if the packet header looks like this, then here's what to do with the packet'. If the rule doesn't match the packet, then the next rule in the chain is consulted. Finally, if there are no more rules to consult, then the kernel looks at the chain policy to decide what to do. In a security-conscious system, this policy usually tells the kernel to reject or deny the packet.

13.4 Using *ipchains*

First, check that the version of *ipchains* you have is 1.3.4 or above. There are several things you can do with *ipchains* in managing the set of rules in the chains. creating(-N) a new chain, deleting a rule(-D), listing the rules in a chain(-L), listing currently masqueraded connections (-M -L), are among the important commands you need to learn inorder to manage your firewall. *ipchains* has a fairly detailed manual page (*man ipchains*), and we refer the reader to this manual.

13.5 Filtering Specifications

- **Specifying source and destination IP addess:** Source(-s) and destination (-d) IP addresses can be specified in four ways. The most common way is to use the full name, such as 'localhost' or 'keren.csd.uoa.er'. The second way is to specify the IP address such as '127.0.0.1'.

  The third and fourth way allow specification of a group of IP addresses, such as '199.95.207.0/24' or '199.95.207.0/255.255.255.0'. These both specify any address from 199.95.207.0 to 199.95.207.255 inclusive; the digits after the '/' tell which parts of the IP address are significant.

- **Specifying Protocol:** The protocol can be specified with the '-p' flag. Protocol can be a number (if you know the numeric protocol values for IP) or a name for the special cases of 'TCP', 'UDP' or 'ICMP'. Case doesn't matter, so 'tcp' works as well as 'TCP'.

- **Specifying UDP and TCP Ports:** For the special case where a protocol of TCP or UDP is specified, there can be an extra argument indicating the TCP or UDP port, or an (inclusive) range of ports. A range is represented using a ':' character, such as '6000:6010', which covers 11 port numbers, from 6000 to 6010 inclusive. If the lower bound is omitted, it defaults to 0. If the upper bound is omitted, it defaults to 65535. So to specify TCP connections coming from ports under 1024, the syntax would be as '-p TCP -s 0.0.0.0/0 :1023'.

- **Specifying ICMP Type and Code:** ICMP also allows an optional argument, but as ICMP doesn't have ports, (ICMP has a type and a code) they have a different meaning. You can specify them as ICMP names (use *ipchains -h* icmp to list the names) after the '-s' option, or as a numeric ICMP type and code, where the type follows the '-s' option and the code follows the '-d' option.

- **Specifying an Interface:** The '-i' option specifies the name of an interface to match. An interface is the physical device the packet came in on, or is going out on. You can use the *ifconfig* command to list the interfaces which are 'up' (ie. working at the moment).

- **Specifying TCP SYN Packets Only:** It is sometimes useful to allow TCP connections in one direction, but not the other. For example, you might want to allow connections to an external WWW server, but not connections from that server. The naive approach would be to block TCP packets coming from the server. Unfortunately, TCP connections require packets going in both directions to work at all.

  The solution is to block only the packets used to request a connection. These packets are called SYN packets. By disallowing only these packets, we can stop attempted connections in their tracks. The '-y' flag is used for this: it is only valid for rules which specify TCP as their protocol. For example, to specify TCP connection attempts from 192.168.1.1: '-p TCP -s 192.168.1.1 -y'.

13.6 Manipulating the Type Of Service (TOS)

There are four seldom-used bits in the IP header, called the Type of Service (TOS) bits. They effect the way packets are treated; the four bits are "Minimum Delay", "Maximum Throughput", "Maximum Reliability" and "Minimum Cost". Only one of these bits is allowed to be set. The most common use is to set telnet and ftp control connections to "Minimum Delay" and FTP data to "Maximum Throughput". This would be done as follows:

```
[root@eritrea]♯ ipchains -A output -p tcp -d 0.0.0.0/0 telnet -t 0x01 0x10
[root@eritrea]♯ ipchains -A output -p tcp -d 0.0.0.0/0 ftp -t 0x01 0x10
[root@eritrea]♯ ipchains -A output -p tcp -s 0.0.0.0/0 ftp-data -t 0x01 0x08
```

The '-t' flag takes two extra parameters, both in hexadecimal. These allow complex twiddling of the TOS bits: the first mask is ANDed with the packet's current TOS, and then the second mask is XORed with it. If this is too confusing, just use the following table:

TOS Name	Value	Typical Uses
Minimum Delay	0x01 0x10	ftp, telnet
Maximum Throughput	0x01 0x08	ftp-data
Maximum Reliability	0x01 0x04	snmp
Minimum Cost	0x01 0x02	nntp

Example: You might want to deny all TCP-packets coming from the web site *www.BadInfo.com* through port 80(*www*). First, you may check the existence of this site from your Internet browser. Or use telnet on port *www*.

```
[root@eritrea]♯ telnet www.BadInfo.com www
```

```
Trying 209.224.235.152...
Connected to www.badinfo.com.
Escape character is '^]'.
```

It's working. Now refuse the *www.BadInfo.com* www-packets to enter your network.

```
[root@eritrea]♯ ipchains -A output -d www.BadInfo.com www -p tcp -j DENY
```

You can list your updated rule as.

```
[root@eritrea]♯ ipchains -L output
```

```
Chain output (policy ACCEPT):
target     prot opt     source               destination          ports
ACCEPT     all  ---f--  anywhere             anywhere             n/a
```

```
acctout    all  ------  anywhere              anywhere              n/a
acctboth   all  ------  anywhere              anywhere              n/a
out        all  ------  anywhere              anywhere              n/a
DENY       tcp  ------  anywhere              209.224.235.152       any ->  www
```

Thus, the client *keren* will not access any web page at www.BadInfo.com.

[root@keren]♯ telnet www.badinfo.com www

```
Trying 209.224.235.152...
```

delete rule number 5 from the input chain as.

[root@eritrea]♯ ipchains -D output 5

Similarly, a firewall can be configured to secure more complex and safety-critical systems in the Internet. A detailed example can be found in [5].

14 Summary

In establishing a reliable and cost-effective computer network, this thesis talks on all the different phases of establishment. As in figure 7, these phases are: building, equipping and securing the network.

14.1 Building the Network

The first thing to do in building a network is to install the network operating system (Linux in our case) on both the clients and the server. In this thesis we have seen what are the required hardwares and how disk can be partitioned for proper installation of single or multiple operating systems.

Beside to the intuitive wired network, two other alternative technologies for building a network are also presented. There are several communication protocols used to connect two computers with the help of a telephone and a modem on both ends. One of these Dial-up protocols, which we have adopted, is the Point-to-Point Protocol(PPP). A PPP-server should always be ready and waiting for calls. Once a call arrives, it should verify the user name and password pair for authentication purpose. Finally, it should assign an IP-number to the calling machine. Where as, A PPP-client initiate

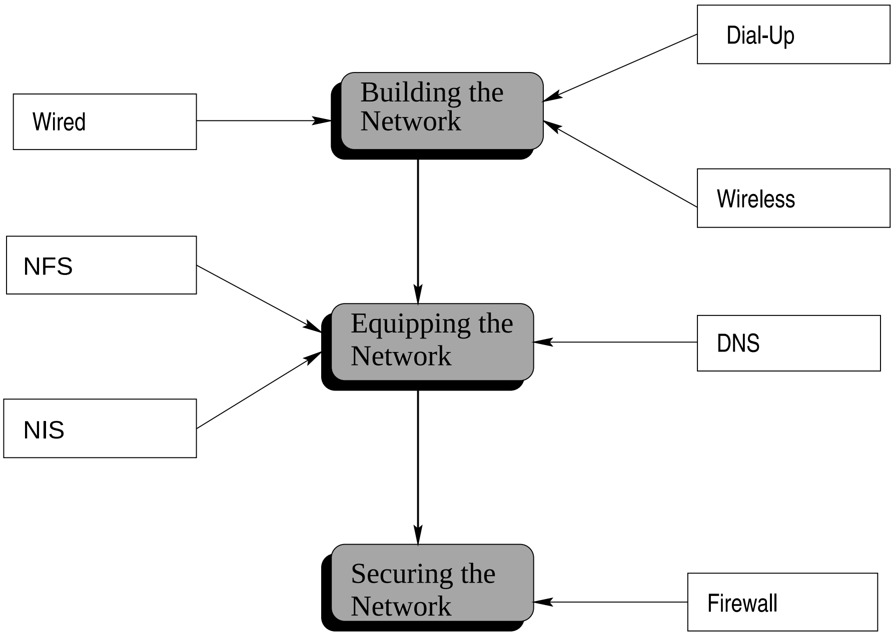


Figure 7: Summary of the thesis

the connection and supplies the correct user name and password pair to the server. Chapter 10 discusses these dial-in and dial-out procedures of PPP on Linux. The second alternative, that has been presented here is the wireless network. Its decreasing cost of installation and maintenance, wireless networking, attracts many institutions. Its effectiveness has been proven in different areas around the world. As an example we have presented the case of Obafemi Awolowo University(OAU) of Ile-Ife in Nigeria. Since 1996, Linux-based wireless network on personal computers provides connectivity between several faculties and departments on the campus.

14.2 Equipping the Network

As it was discussed in the introductory part of this thesis. The main goals for having a computer network is to share resource efficiently and to have some degree of security. To that end a network should provide several services to enable its users access the available resources where ever in the network they sit on.

Network File system(NFS) is one of this services. It permits users to execute files without knowing the locations of these files. The NFS server has a list of allowed clients and file systems that are needed to be accessible. When NFS is running from the background a client can mount the permitted file systems on demand, and it can use them as if they are local files.

Another complimentary service to NFS is Network Information Service(NIS). The purpose of NIS is to provide information that has to be known throughout the network. Information that are likely to be distributed by NIS is user authentication informations: login name, password and group information. With the help of NIS a user can log to the network from any machine in the network. Thus both NIS and NFS makes all machines in the network look identical to the user.

Domain Name System(DNS) is another important service that has to be available in the network. Every host in a TCP/IP network has a numerical unique identification number called *IP number*. But the problem with *IP number* is that, it is hard to remember and with respect to big networks a plane 32-bit number is so difficult to organize and keep its uniqueness property. To solve these problems a DNS service associate a unique name for every *IP number* by classifying them into domains and subdomains. Chapter 9 explains on how to use DNS and how to create your own DNS database for your network.

14.3 Securing the Network

When we have the network operating well, and all services are running properly; asking for the security of the services would be the appropriate question. We could have information in our local network which is private to the local users, and we might want to keep it inside. There are so many tempting informations in the Internet that are harmful to youngsters and even adults from different cultures and society, that we have to keep them from coming inside our local network. And there could be so many security issues that we might want to deal with. These matters are jointly handled in the TCP/IP network by a package called *Firewall*. The Last chapter is dedicated for the Linux firewall configuration.

For a large scale network, the thesis can be expanded by including configurations of modem polls for a busy dial-up connections. Services like NIS are not secured and they are vulnerable for various attacks. It would be interesting to investigate some other encryption enhanced services for secured network. It is also interesting to compare and contrast the cost/efficiency of Windows NT versus Linux under the features that have been discussed in this thesis.

Acknowledgement

We would like to thank our advisor Dr. Ivan Christoff and the departments system administrator Joel Frederick for various discussions, suggestions and feedbacks, especially Frederick for introducing us to Linux operating system. We would also like to thank Prof. Sten Kaijser and Yordanos G. Beyene for their various helps.

References

[1] Andrew S. Tanenbaum: *Computer Networks*, 3rd ed. NJ, Prentice-Hall, 1995.

[2] Drew Heywood: *Using Windows NT Server 4*, Que Corporation, 1998.

[3] Drew Heywood: *Using Linux* , Que Corporation, 1998.

[4] ED TAYLOR: *TCP/IP Complete*, McGraw-Hill, 1998.

[5] Eric S. Raymond:*Linux Undercover*, Linux Secrets as Revealed by the Linux Documentation Project, 1998.

[6] F. Postogna, et al.:*Wireless Networking in Africa*,Linux Journal pp 56, 42, 1998.

[7] Kirch, John Networking Consultant and Microsoft Certified Professional, *UNIX-Verses-Windows NT server 4*: 23 March 1999 ; http://www.unix-vs-nt.org/kirch/.

[8] Red Hat Software, Inc.: *Linux Installation Guide*, Red Hat Software, Inc. 1999.

[9] S. Keshav: *An Engineering Approach to Computer Networking*, ATM networks, the Internet, and the telephone network: Addison Wesley, 1997.

[10] The Wireless LAN Alliance ($WLANA$):*Introduction to Wireless LANs*, 1996.: http://www.wlana.com.

[11] Wack, John P. and Carnahan, Lisa J,: *Keeping Your Site Comfortably Secure: An Introduction to Internet Firewalls*: NIST Special Publication 800-10. US DEPARTMENT OF COMMERCE, National Institute of Standards and Technology. [http://csrc.ncsl.nist.gov/nistpubs/800-10/main.html]

Index

Numbers written in italic refer to the page where the corresponding entry is described, the ones underlined to the definition, the rest to the pages where the entry is used.