1 Networking Overview **4Hrs**

1.1 History
1.2 Protocol Standards – *networking protocols overview*
1.3 Reference Model (OSI, TCP/IP)
1.3.1 Overview, comparison OSI and TCP-IP
1.4 Windows and Linux Networking Basics -*overview*
1.5 Switching and Routing basics – *Circuit/packet/message switching, Unicast/multicast/broadcast/anycast routing overview*

What is a Network?

A network consists of two or more computers that are linked in order to share resources (such as printers and CDs), exchange files, or allow electronic communications. The computers on a network may be linked through cables, telephone lines, radio waves, satellites, or infrared light beams.

Two very common types of networks include:

- Local Area Network (LAN)
- Wide Area Network (WAN)

You may also see references to a Metropolitan Area Networks (MAN), a Wireless LAN (WLAN), or a Wireless WAN (WWAN).

Networking is just what it says; setting up a network where you can meet people working in your field of interest. If you ask people around you if they know someone in the field you are interested in, you will be surprised by the number of referrals you can get. Calling someone at a company that you have a connection with through a friend makes it easier to get your foot in the door of that company.

Local Area Network

A Local Area Network (LAN) is a network that is confined to a relatively small area. It is generally limited to a geographic area such as a writing lab, school, or building.

Computers connected to a network are broadly categorized as servers or workstations. Servers are generally not used by humans directly, but rather run continuously to provide "services" to the other computers (and their human users) on the network. Services provided can include printing and faxing, software hosting, file storage and sharing, messaging, data storage and retrieval, complete access control (security) for the network's resources, and many others.

Wide Area Network

Wide Area Networks (WANs) connect networks in larger geographic areas, such as Florida, the United States, or the world. Dedicated transoceanic cabling or satellite uplinks may be used to connect this type of global network.

Using a WAN, schools in Florida can communicate with places like Tokyo in a matter of seconds, without paying enormous phone bills. Two users a half-world apart with workstations equipped with microphones and a webcams might teleconference in real time. A WAN is complicated. It uses multiplexers, bridges, and routers to connect local and metropolitan networks to global communications networks like the Internet. To users, however, a WAN will not appear to be much different than a LAN.

Brief history of networking

Making devices talk to each other for the purposes of communication is nothing new. Early forays into telephony such as the telegraph and telephone have since evolved into more complicated devices, and now a computer can be networked to the Internet, another PC, or even a home stereo. In the early 1960s, individual computers had to be physically shared, making the sharing of data and other information difficult. Seeing this was impractical, researchers developed a way to "connect" the computers so they could share their resources more efficiently. Hence, the early computer network was born.

Through the then-new communication protocol known as packet switching, a number of applications, such as secure voice transmission in military channels became possible. These new circuits provided the basis for the communication technologies of the rest of the 20th century, and with further refinement these were applied to computer networks.

These networks provided the basis for the early ARPANET, which was the forerunner of the modern Internet. The Advanced Research Projects Agency (ARPA) submitted the proposal for the project on June 3, 1968 which was approved a few weeks later. This proposal entitled "Resource Sharing Computer Networks" would allow ARPA not only the further sharing of their data, but would allow them to further their research in a wide variety of military and scientific fields. After being tested in four locations, the network spread and the new protocols created for its use evolved into today's World Wide Network.

In 1977, early PC-based Local Area Networks, or LANs (Local Area Networks) were spreading, and while initially restricted to academics and hobbyists, they eventually found their way into the workplace and in homes, although the explosion into the latter two arenas is a relatively recent phenomenon. LAN variants also developed, including Metropolitan Area Networks (MANs) to cover large areas such as a college campus, and Wide Area Networks (WANs) for university-to-university communication. With the widespread use of computers in the corporate world, the speed and convenience of using them to communicate and transfer data has forever altered the landscape of how people conduct business.

Networks have have become an integral part of the corporate world. Ubiquitous computing and Internet-capable cellular phones have allowed people to remain connected, even if the individual is away from a fully wired office environment. With the Internet, the daily functions of corporate life have been mutated and improved, allowing for files, information, and other information to be transmitted at near instant speeds. However, with it also comes the simultaneous requirement to keep data and communications secure.

Market leaders

Much of the technological advances in networking come from a wide variety of sources, but a number of companies continue to innovate and lead by providing the infrastructure and necessary hardware. While Microsoft dominates the field of operating systems on workstations and on many servers with Windows, the open source Apache Web server provides the foundation for more Web servers than any competing product by a tremendous margin. Numerous companies continue to develop and invent new technology, such as hardware from Cisco Systems. Known for their routers and countless other products, they also provide diverse wireless networking solutions through their Linksys brand. Other networking market leaders include Nortel Networks, Novell, Lucent Technologies and Juniper Networks.

The future of networking

While the initial concept behind networking computers was to see every person on the planet being "wired", the evolution of the technology aims to do just the opposite. Wireless technologies are emerging as a popular, cable- free alternative to traditional wired networks. By 2009, it is predicted that wearable computers – which will replace the personal digital assistant or PDA – will be fully integrated in the workplace, with the ability to connect to both wired and wireless networks. Other emerging technologies include smart appliances - products that have enhanced capabilities and / or the ability to access the Internet, to fully automated homes that have all appliances, heating and cooling systems, entertainment and other home needs connected via a LAN. Recent drivers of networking trends include Voice-Over Internet Protocol (VoIP) and convergence.

protocol

In information technology, a protocol is the special set of rules that end points in a telecommunication connection use when they communicate. Protocols specify interactions between the communicating entities.

Protocols exist at several levels in a telecommunication connection. For example, there are protocols for the data interchange at the hardware device level and protocols for data interchange at the application program level. In the standard model known as Open Systems Interconnection (OSI), there are one or more protocols at each layer in the telecommunication exchange that both ends of the exchange must recognize and observe. Protocols are often described in an industry or international standard.

The TCP/IP Internet protocols, a common example, consist of:

- Transmission Control Protocol (TCP), which uses a set of rules to exchange messages with other Internet points at the information packet level
- Internet Protocol (IP), which uses a set of rules to send and receive messages at the Internet address level
- Additional protocols that include the Hypertext Transfer Protocol (HTTP) and File Transfer Protocol (FTP), each with defined sets of rules to use with corresponding programs elsewhere on the Internet

There are many other Internet protocols, such as the Border Gateway Protocol (BGP) and the Dynamic Host Configuration Protocol (DHCP).

The word *protocol* comes from the Greek *protocollon*, meaning a leaf of paper glued to a manuscript volume that describes the contents.

Standard Protocol

A standard protocol is a mandated, fixed procedure for completing a task. Often designated by flow chart or spelled out in textual steps, standard protocol may be nationally recognized or de facto--accepted, but not nationally recognized. There are as many types of standard protocol as there are tasks associated with them.

1. **Computer Communication**

    - Electronics like telephones, computers and DVD recorders communicate by way of data lines. These single lines provide device-to-device communication through interconnection. Its standard protocol creates data packets that are scrambled through the lines and reassembled so the receiver can understand the message.

2. **Medical Protocol**

    - Body fluid spills in the emergency room require that environmental specialists follow specific protocol to clean up the mess so patients and hospital personnel are not subjected to infection and viruses such as AIDS. Occupational Safety and Health Administration clean-up protocol falls under the standard protocol "Universal Precautions," while hospital clean-up policy may fall under de facto standard protocol.

3. **Decision-Making Protocol**

    - A type of de facto standard protocol manages decision-making processes. These often become industry standards if the selected protocol works for the group or for a project. Quality teams brainstorming and choosing what protocol makes sense for a project falls under de facto protocol.

What is a Network Protocol

A protocol is a set of rules that governs the communications between computers on a network. These rules include guidelines that regulate the following characteristics of a network: access method, allowed physical topologies, types of cabling, and speed of data transfer.

Types of Network Protocols

The most common network protocols are:

- Ethernet
- Local Talk
- Token Ring
- FDDI
- ATM

Network Protocol Overview

The OSI model, and any other network communication model, provides only a conceptual framework for communication between computers, but the model itself does not provide specific methods of communication. Actual communication is defined by various communication protocols. In the context of data communication, a protocol is a formal set of rules, conventions and data structure that governs how computers and other network devices exchange information over a network. In other words, a protocol is a standard procedure and format that two data communication devices must understand, accept and use to be able to talk to each other.

In modern protocol design, protocols are "layered" according to the OSI 7 layer model or a similar layered model. Layering is a design principle which divides the protocol design into a number of smaller parts, each part accomplishing a particular sub-task and interacting with the other parts of the protocol only in a small number of well-defined ways. Layering allows the parts of a protocol to be designed and tested without a combinatorial explosion of cases, keeping each design relatively simple. Layering also permits familiar protocols to be adapted to unusual circumstances.

The header and/or trailer at each layer reflect the structure of the protocol. Detailed rules and procedures of a protocol or protocol group are often defined by a lengthy document. For example, IETF uses RFCs (Request for Comments) to define protocols and updates to the protocols.

A wide variety of communication protocols exists. These protocols were defined by many different standard organizations throughout the world and by technology vendors over years of technology evolution and development. One of the most popular protocol suites is TCP/IP, which is the heart of Internetworking communications. The IP, the Internet Protocol, is responsible for exchanging information between routers so that the routers can select the proper path for network traffic, while TCP is responsible for ensuring the data packets are transmitted across the network reliably and error free. LAN and WAN protocols are also critical protocols in network communications. The LAN protocols suite is for the physical and data link layers of communications over various LAN media such as Ethernet wires and wireless radio waves. The WAN protocol suite is for the lowest three layers and defines communication over various wide-area media, such as fiber optic and copper cables.

Network communication has slowly evolved. Today's new technologies are based on the accumulation over years of technologies, which may be either still existing or obsolete. Because of this, the protocols which define the network communication are highly inter-related. Many protocols rely on others for operation. For example, many routing protocols use other network protocols to exchange information between routers.

In addition to standards for individual protocols in transmission, there are now also interface standards for different layers to talk to the ones above or below (usually operating system specific). For example: Winsock and Berkeley sockets between layers 4 and 5; NDIS and ODI between layers 2 and 3.

The protocols for data communication cover all areas as defined in the OSI model. However, the OSI model is only loosely defined. A protocol may perform the functions of one or more of the OSI layers, which introduces complexity to understanding protocols relevant to the OSI 7 layer model. In real-world protocols, there is some argument as to where the distinctions between layers are drawn; there is no one black and white answer.

To develop a complete technology that is useful for the industry, very often a group of protocols is required in the same layer or across many different layers. Different protocols often describe different aspects of a single communication; taken together, these form a protocol suite. For example, Voice over IP (VOIP), a group of protocols developed by many vendors and standard organizations, has many protocols across the 4 top layers in the OSI model.

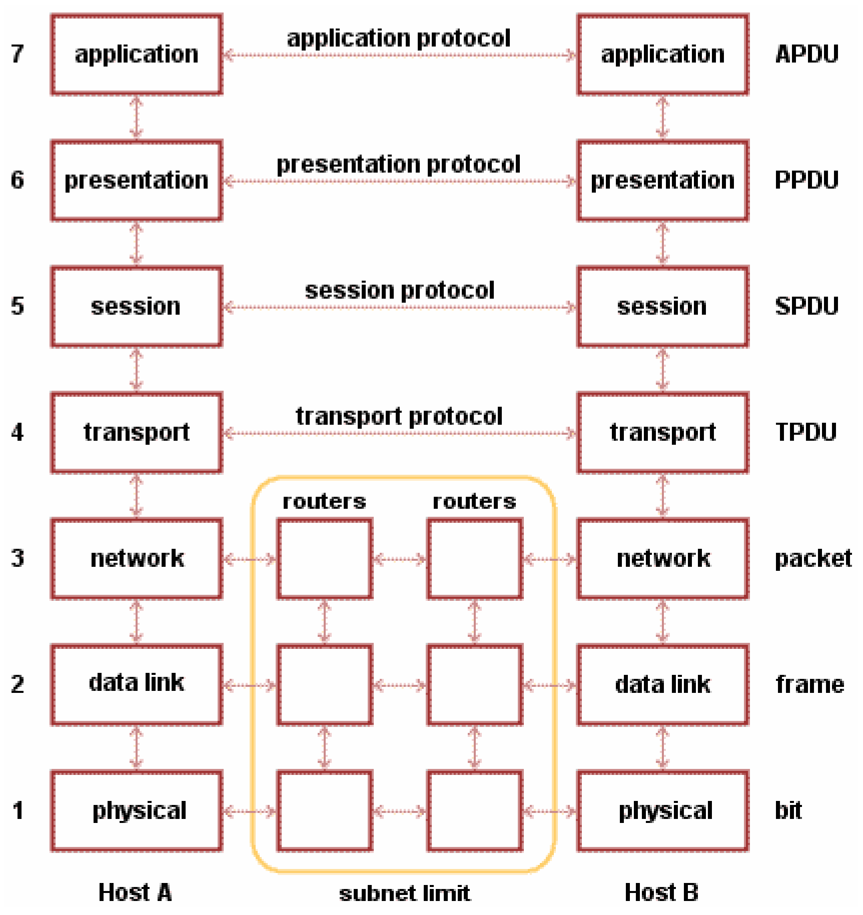
Protocols can be implemented either in hardware or software or a mixture of both. Typically, the lower layers are implemented in hardware, with the higher layers being implemented in software.

Protocols could be grouped into suites (or families, or stacks) by their technical functions, or origin of the protocol introduction, or both. A protocol may belong to one or multiple protocol suites, depending on how you categorize it. For example, the Gigabit Ethernet protocol IEEE 802.3z is a LAN (Local Area Network) protocol and it can also be used in MAN (Metropolitan Area Network) communications.

Most recent protocols are designed by the IETF for Internetworking communications and by the IEEE for local area networking (LAN) and metropolitan area networking (MAN). The ITU-T contributes mostly to wide area networking (WAN) and telecommunications protocols. ISO has its own suite of protocols for internetworking communications, which is mainly deployed in European countries.

OSI Reference Model

Open Systems Interconnection (OSI) is a standard reference model for communication between two end users in a network. The model is used in developing products and understanding networks. Also see the notes below the figure.



OSI divides telecommunication into seven layers. The layers are in two groups. The upper four layers are used whenever a message passes from or to a user. The lower three layers are used when any message passes through the host computer. Messages intended for this computer pass to the upper layers. Messages destined for some other host are not passed up to the upper layers but are forwarded to another host. The seven layers are:

Layer 7: The application layer ...This is the layer at which communication partners are identified, quality of service is identified, user authentication and privacy are considered, and any constraints on data syntax are identified. (This layer is *not* the application itself, although some applications may perform application layer functions.)

Layer 6: The presentation layer ...This is a layer, usually part of an operating system, that converts incoming and outgoing data from one presentation format to another (for example, from a text stream into a popup window with the newly arrived text). Sometimes called the syntax layer.

Layer 5: The session layer ...This layer sets up, coordinates, and terminates conversations, exchanges, and dialogs between the applications at each end. It deals with session and connection coordination.

Layer 4: The transport layer ...This layer manages the end-to-end control (for example, determining whether all packets have arrived) and error-checking. It ensures complete data transfer.

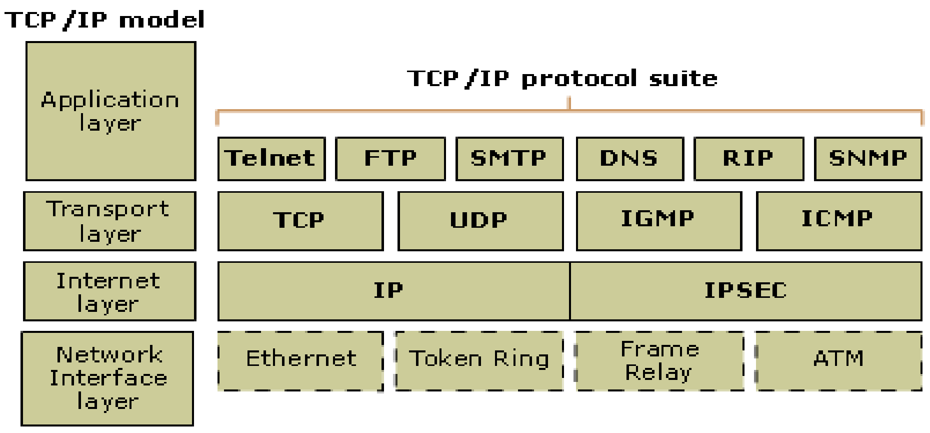
Layer 3: The network layer ...This layer handles the routing of the data (sending it in the right direction to the right destination on outgoing transmissions and receiving incoming transmissions at the packet level). The network layer does routing and forwarding.

Layer 2: The data-link layer ...This layer provides synchronization for the physical level and does bit-stuffing for strings of 1's in excess of 5. It furnishes transmission protocol knowledge and management.

Layer 1: The physical layer ...This layer conveys the bit stream through the network at the electrical and mechanical level. It provides the hardware means of sending and receiving data on a carrier.

TCP/IP REFERENCE MODEL

The TCP/IP reference model, is based on a suite of protocols in which each protocol solves a particular network communications problem. The TCP/IP model can be used in a heterogeneous environment that has equipment from many different vendors. There are four layers on the TCP/IP reference model.



Layers:

- Layer 1: The Network Layer
- Layer 2: The Internet Layer
- Layer 3: The Transport Layer
- Layer 4: The Application Layer

Layer 1: Network Layer

The network access layer is responsible for exchanging data between a host and the network and for delivering data between two devices on the same network. Node physical addresses are used to accomplish delivery on the local network. Functions performed at this level include encapsulation of IP datagrams(i.e. the packet format defined by Internet Protocol.) into the frames transmitted by the network, and the mapping of IP addresses into the physical addresses used by the network. The TCP/IP Network Access Layer can encompass the function of all three lower layers of the OSI reference model (Network Layer, Data Link Layer, and Physical Layer.)

Layer 2: Internet Layer

The internet layer is responsible for sending source packets from any network on the internetwork and have them arrive at their destination regardless of the path they took. The Internet Protocol (IP) is used in this layer and it provides the packet delivery service on which the TCP/IP is based. The IP protocol implements a system of logical host addresses called IP addresses. The IP addresses are used by the internet and higher layers to identify devices and to perform internetwork routing.

Layer 3: Transport Layer

The transport layer is responsible for the reliability, flow control, and error correction of data being sent across the network.Its main protocol is called the transmission control protocol (TCP). TCP provides reliable data delivery service with end-to-end error detection and correction.User Datagram Protocol (UDP) is another protocol used which provides slow-overhead, connectionless datagram delivery service.UDP is unreliable but enhances network throughput when error correction is not required at the host-to-host-layer. Both protocols deliver data between the Application Layer and the Internet Layer. The users may choose either best suited to their means.

Layer 4: Application Layer

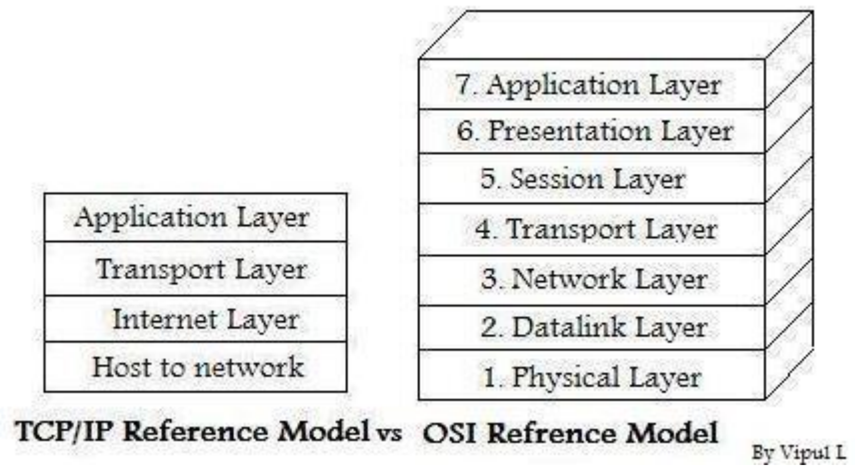
The application layer is responsible for handling high-level protocols, issues of representation, encoding and dialog control. This layer is broadly equivalent to the application, presentation and session layers of the OSI model. It gives an application access to the communication environment. Examples of protocols found at this layer are Telnet, FTP (File Tranfer Protocol), SNMP (Simple Network Management Protocol), HTTP (Hyper Text Transfer Protocol) and SMTP (Simple Mail Transfer Protocol).

TCP/IP Model Vs OSI Model

This article is about TCP/IP model vs OSI model. It is meant to highlight the differences between the two set standards of the industry. TCP/IP model and the OSI model have been the two protocol suites on which the communication industry heavily relies on.

Both TCP/IP model and OSI model work in a very similar fashion. But they do have very subtle differences too. Knowing these differences is very crucial to learning computer networking. This article will try to show the comparison between TCP/IP model and OSI model.

Background
Advance research project agency (ARPA) created a OSI reference model so that they could logically group the similarly working components of the network into various layers of the protocol. But after the advent of the Internet, there arose the need for a streamlined protocol suite, which would address the need of the ever-growing Internet. So the Defense Advanced Research Project Agency (DARPA) decided to create the TCP/IP protocol suite. This was going to address many, if not all the issues that had arisen with the OSI reference model.



TCP/IP Model Layers

TCP/IP is a suite of protocol which is named after its most significant pair of protocols. That is Transmission Control Protocol and Internet Protocol. TCP/IP are made up of layers. Each layer is responsible for a set of computer network related tasks. Every layer provides service to the layer above it. In all, there are four layers in the TCP/IP reference model.

Application Layer: This is the topmost layer of the TCP/IP suite. This is responsible for coding of the packet data.

Transport Layer: This layer monitors end-to-end path selections of the packets. It also provides service to the application layer.

Internet Layer: This layer is responsible for sending packets through different networks.

Link Layer: It is the closest layer to the network hardware. It provides service to Internet layer.

OSI Model Layers

In OSI reference model there seven layers of protocols. Again, in OSI reference model, each layer provides services to the layer above it. There are in all seven layers of in OSI. They are

Physical Layer: It is the lower most layer of the OSI reference model. It is layer which is responsible for direct interaction of the OSI model with hardware. The hardware provides service to the physical layer and it provides service to the datalink layer.

Datalink Layer: There may be certain errors which may occur at the physical layer. If possible, these errors are corrected by the datalink layer. The datalink layer provides the way by which various entities can transfer the data to the network.

Network Layer: It does not allow the quality of the service to be degraded that was requested by the transport layer. It is also responsible for data transfer sequence from source to destination.

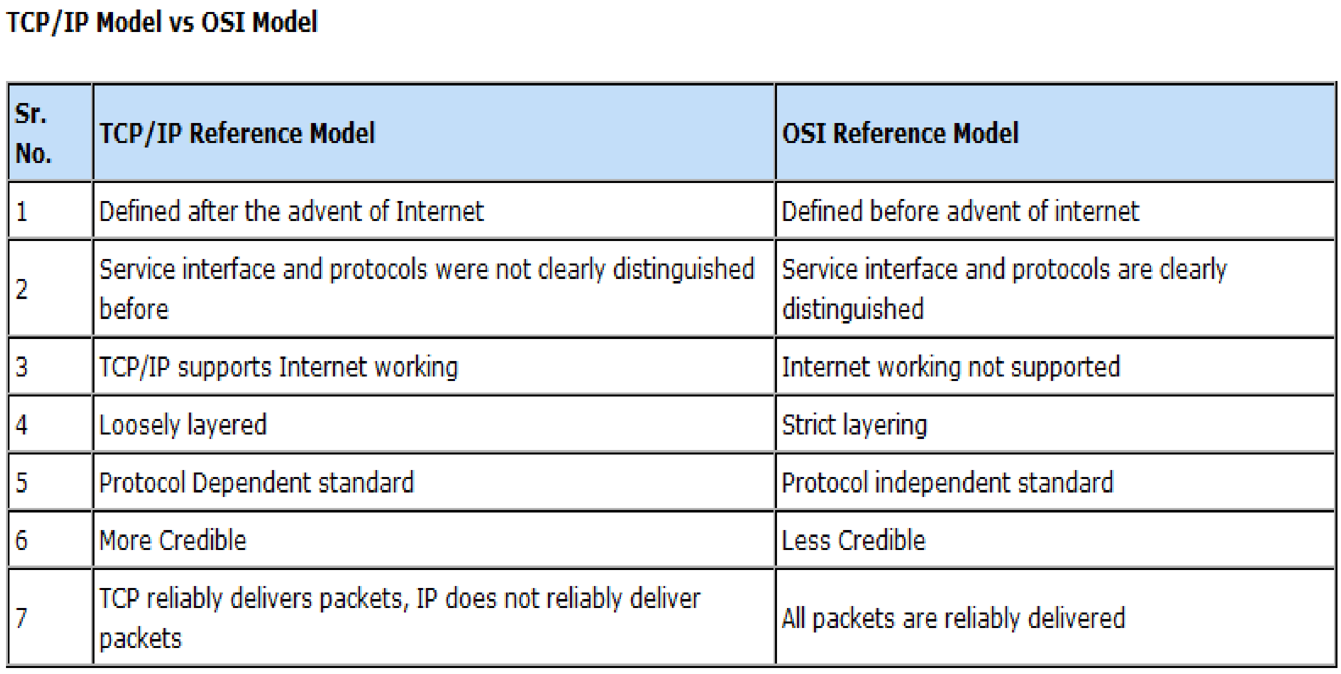
Transport Layer: The reliability of the data is ensured by the transport layer. It also retransmits those data that fail to reach the destination.

Session Layer: The sessions layer is responsible for creating and terminating the connection. Management of such a connection is taken care of by the sessions layer.

Presentation Layer: This layer is responsible for decoding the context (syntax and semantics) of the higher level entities.

Application Layer: Whichever software application that implements socket programming will communicate with this layer. This layer is closest to the user.

TCP/IP Model vs OSI Model



TCP/IP Model vs OSI Model

Sr. No.	TCP/IP Reference Model	OSI Reference Model
1	Defined after the advent of Internet	Defined before advent of internet
2	Service interface and protocols were not clearly distinguished before	Service interface and protocols are clearly distinguished
3	TCP/IP supports Internet working	Internet working not supported
4	Loosely layered	Strict layering
5	Protocol Dependent standard	Protocol independent standard
6	More Credible	Less Credible
7	TCP reliably delivers packets, IP does not reliably deliver packets	All packets are reliably delivered

The entire communication industry stands on the backbone of TCP/IP and OSI reference model. It is very vital to learn the above differences, if anyone wants to be an expert in the field of communication.

Basic Windows networking principals

By default, all Windows operating systems use the TCP/IP protocol suite to communicate with each other through network devices. Any computer network adaptor using TCP/IP requires 3 things to communicate with other computers: An IP address, a subnet mask for that IP address, and a default gateway. These terms will be defined in a moment. First, though, an idea of how a TCP/IP network works logically.

When you give a computer an IP address, you identify the network which it is a member of, and give it an identification number within that network. A computer in a given network can communicate with any other computer that is local to it (in the same network), provided there is a way for information to pass between them (network cables, wireless network, etc.).

Computers in a network cannot, however, communicate with computers in a different network (remote network) directly, even if they are physically connected to each other via cables.

This is where the **default gateway** comes in. A gateway is defined as a path out of the local network to other remote networks. A gateway can be a number of things physically, such as a DSL/cable router for your local network, a Windows server computer with multiple network adaptors split between different networks, etc. Gateways must all share one thing in common though. They are connected to at least 2 networks, and have the ability to pass traffic between them.

The default gateway assigned to a network adaptor is sent all traffic that does not belong in the local network. As an example of this, say you have a DSL Internet connection. When you connect to the Internet, you are provided with a default gateway assigned by your service provider. When you attempt to connect to a site on the Internet, the URL you type in (say www.pcstats.com) is converted into an IP address by your Internet provider.

Since this address is not going to be in your local network, the network adaptor in your computer forwards the request for the web page to its default gateway, your service provider.

From that point, your request will be passed from network to network through the internet until it reaches the local network of www.pcstats.com and the data needed to display the web page starts its way back through the internet to your IP address.

Fundamentals of Linux Networking

As you know, Linux itself is just a kernel. Linux can be configured as a networked workstation, a DNS server, a DHCP server, a web server, a mail server, a file and print server, database server, a firewall, a gateway router and many more. Is Linux can be all that? Yes it can. In this section, user will learn the true meaning of Linux, which is a network operating system or server system, to be all that mention above.

The first thing you should learn about networking is ip addresses. A server and a client computer must have an ip address so that it can be reach in network environment.

Linux server can serves networking 365 days a year without any problem. It's a proof that Linux is a very stable and secure network operating system when properly configured and maintained. All that starts with setting up a network card and configure an ip address for the server.

Fundamental configurations are:

- **TCP/IP networking in Linux** - identify the characteristics of basic TCP/IP networking in Linux.
- **IP subnets and Linux** - identify the underlying principles of IP subnets.
- **TCP/IP configuration files in Linux** - identify the Linux files you use to configure TCP/IP.
- **Configuring network interfaces in Linux** - configure network interfaces using the ifconfig program.
- **Routing in Linux** - view or configure the routing table.
- **Monitoring a network using Linux tools** - monitor and troubleshoot a TCP/IP network using Linux tools.
- **Configuring TCP/IP in Linux** - configure TCP/IP.
- **Configuring PPP in Linux** - identify how to set up PPP for use on a Linux system.
- **Using PPP in Linux** - use PPP to initiate, terminate, and troubleshoot PPP connections.
- **Establishing PPP connections in Linux** - use different options available when establishing a PPP connection for a given scenario.

Fundamentals of networking under CentOS

In CentOS, all the communication occurs between configured software interfaces and the network devices connected to the System. all network configuration files for interfaces are located in ***/etc/sysconfig/network-scripts/ .***

Three categories of files are located in this directory:

1. interface configuration files
2. interface control scripts
3. network function files

All these file work together to make various network devices work together.

Here is the primary network configuration files:

1. ***/etc/hosts*** -> To resolve the hostnames that can't be resolve any other way
2. ***/etc/resolv.conf*** -> Specifies the IP Address of the DNS server
3. ***/etc/sysconfig/network*** -> specifies routing and host information for all network interfaces
4. ***/etc/sysconfig/network-scripts/ifcfg-<interface-name>*** -> For each network interface, there is a corresponding interface configuration script

Some scripts to configure network in CentOS

To put IP Address on the network

```
# ifconfig eth0 192.168.0.10 netmask 255.255.255.0
```

or

```
# ip addr add 192.168.0.10 dev eth0
```

configuring network with Setup

```
# setup
```

restarting the network service

```
# /etc/init.d/network restart
```

it will show the IP Address of the server also used to check the internet connectivity

```
$ nslookup linuxheaven.net
```

What is Switching ?

Networking concept has two basic concepts and they are Switching and Routing . They are fundamental concepts in Networking, other topics like network security are based on these concept. Routing and Switching are the base packet or data delivering methods in network and each device like router , switch , hub , firewall , proxy , cash server, modem are using one of these methods.

What is switching ?

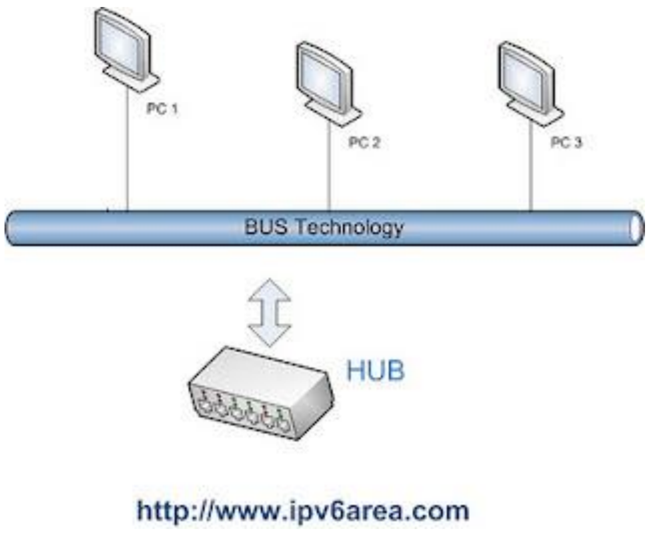
When we are talking about the switching, the protocols and concepts are related to the layer 2 and data packets with in this layer that are called frame.

What is Switch?

As you know the switch refers to the device that can select one condition from 2 or more conditions. For example an electrical switch can select **0** as turn off and select **1** for turn on. Better switches have more choices than 2 conditions and because of this capacity, they are bigger and bigger and also more expensive.

In Networking there is a long history for switches. At first time the direct connection between two or more computers used to make a network and such technology like 10base2, 10base5, and 10base-T for cabling were used.

In those technologies, the base topology was **BUS Technology** and the most advanced of this technology is HUB. A bus or data bus refers to the one **link (shared link)** between 2 or more devices for sending and receiving data, and when a device wants to send data for another device, this bus should be empty and none of devices should not use the bus. In figure below we can see a logical view to a hub and the meaning of bus.



If PC1 wants to send data to PC3 , at first checks the bus condition to avoid collision (this is collision detection) and if bus is empty then it will send the data for PC3. But actually it will broadcast it on the bus and each device in this technology will receive the data , PC3 will take the data with the hardware address that is on the frame .

The data packets in this layer called **frame**, and contains upper level data and in this layer the source and destination address are called MAC addresses.

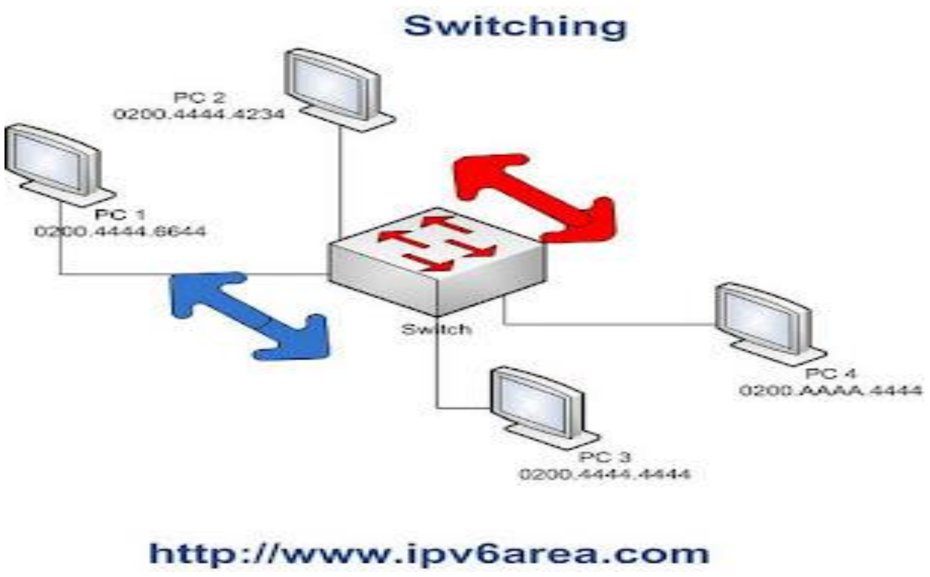
What is MAC address?

The MAC address is unique address (like FFFF.FFFF.FFFF) for each interface in network . PC1 creates a frame with destination MAC address of PC3 and , PC3 after receive checks the MAC address .

The bad point in this technology is that all devices in the bus will receive data or listen to the data .We can see this is a shared resource and in time one device just can use the bus and the others are listener and the bandwidth is divided to the users .

In this term we have other concept like bridges and repeaters .

The switch has some different technology for connecting the devices . The base topology in switches is **Star** and as like a hub all devices connected to the switch . But the basic difference is that each port in switch can connect to another port in direct. In figure below we can see a sample of switches:



PC1 ,PC2 ,PC3 , PC4 are connected to the switch and each of them has MAC address . PC1 and PC3 have connection together and in this time also PC2 and PC4 are sending and receiving data . The switch provides the connectivity and give them the real speed (For example if port speed is 100 mbps they can use this bandwidth).
Switch knows that in each port what is the MAC address in other hand when a device connect to the switch , switch in moment will find the MAC address and when a frame send for this MAC address switch just give the frame to that port that have MAC address and other ports never listen to the data. Yes the better security and data distribution . Switches check all the frames for destination MAC .

Like a router for checking the destination IP addresses , in switch there is a **MAC Table** that contains all MAC addresses that are active. For example this switch (in above figure) has at least four MAC addresses in MAC table.

Switch has a CPU (at least) and this CPU has a MAC address also , and this is the MAC address of switch . Yes like routers they have address but in layer 2 .

This architecture is good , switch has MAC table and with this list will send data just to the destination and not for other ports , this is meaning of filtering with switch.The operation is running in hardware based CPU , this means frame forwarding or switching is a hardware operation and in high speed . A company when creates a switch will say the switching capacity in Gigabit per second(good rate) or higher !!!!!!

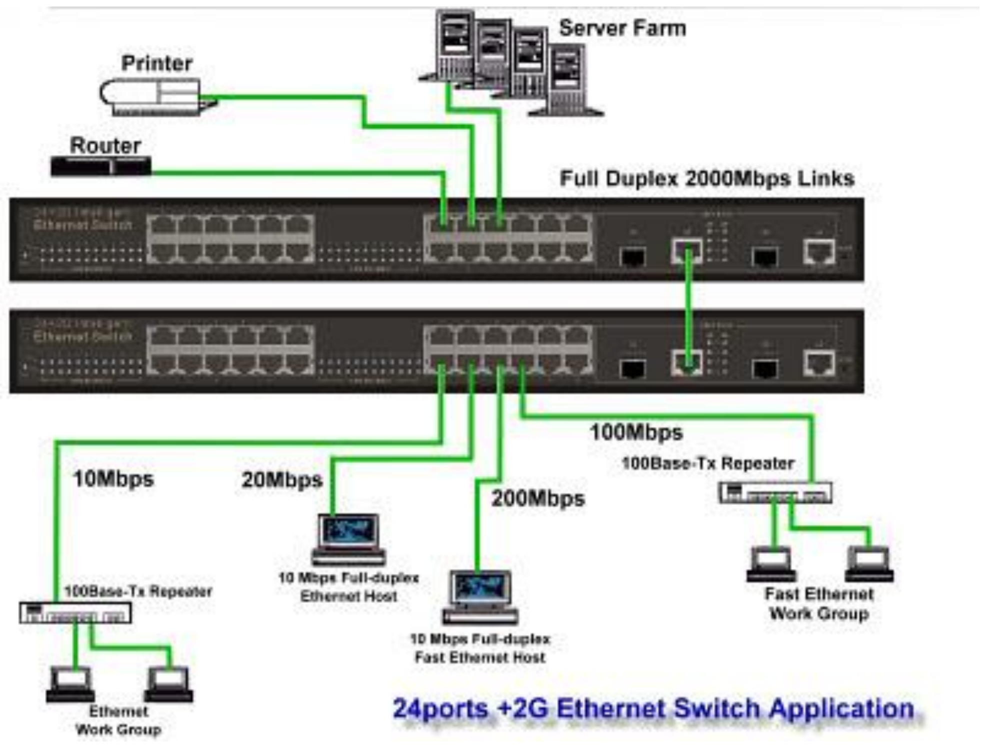
These two switches are samples of good and famous brands in world with high capacity and performance. Depend on the capacity and numbers of ports and design , switches have been selecting . Cisco and Brocade

Cisco 24 port 2960

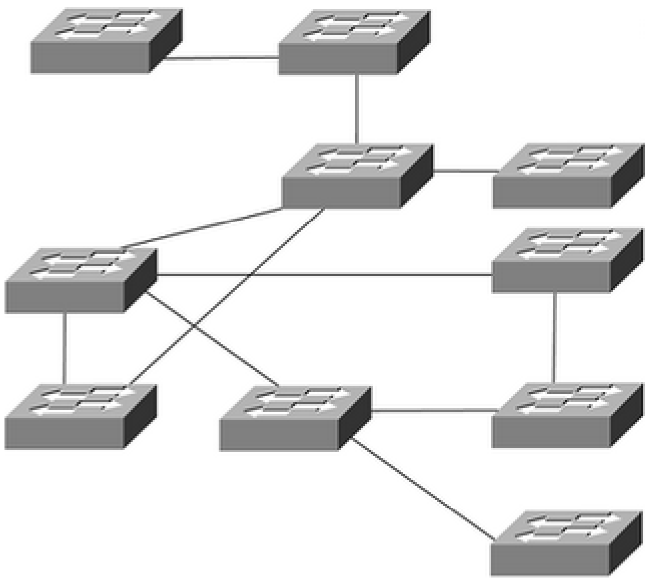
Foundry -Brocade switch 24 port

Simple LAN design with switches:

We can connect the switches together with one or more ports and making the biggest network. In figures below you can see simple diagrams of connectivity with switches .


Server Farm
Printer
Router
Full Duplex 2000Mbps Links
100Mbps
100Base-Tx Repeater
10Mbps
20Mbps
200Mbps
100Base-Tx Repeater
10 Mbps Full-duplex Ethernet Host
10 Mbps Full-duplex Fast Ethernet Host
Fast Ethernet Work Group
Ethernet Work Group
24ports +2G Ethernet Switch Application

Simple design with 2 switches



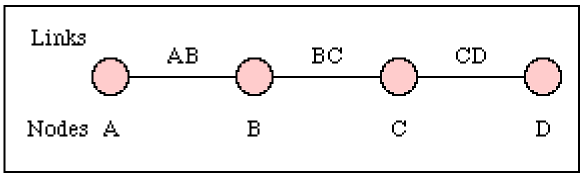
Standards and Protocols :

The basic standards for layer 2 is IEEE and the special standard for Ethernet is IEEE 802. This standard contains details for wired and wireless , switching and VLAN s, STP,,....

Some of the switching schemes are:

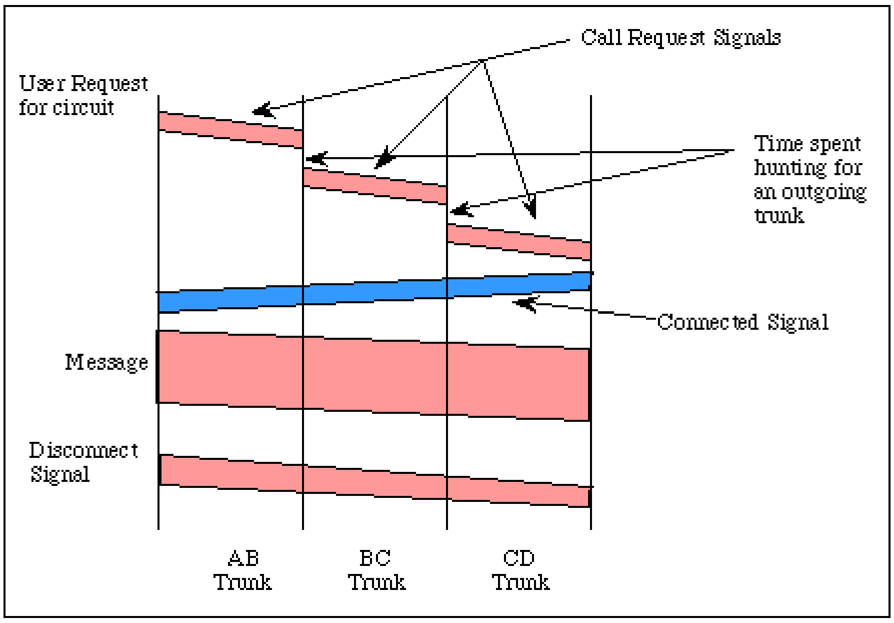
Circuit Switching

Circuit switching is the most familiar technique used to build a communications network. It is used for ordinary telephone calls. It allows communications equipment and circuits, to be shared among users. Each user has sole access to a circuit (functionally equivalent to a pair of copper wires) during network use. Consider communication between two points A and D in a network. The connection between A and D is provided using (shared) links between two other pieces of equipment, B and C.



A connection between two systems A & D formed from 3 links

Network use is initiated by a connection phase, during which a circuit is set up between source and destination, and terminated by a disconnect phase. These phases, with associated timings, are illustrated in the figure below.



A circuit switched connection between A and D

(Information flows in two directions. Information sent from the calling end is shown in pink and information returned from the remote end is shown in blue)

After a user requests a circuit, the desired destination address must be communicated to the local switching node (B). In a telephony network, this is achieved by dialing the number.

Node B receives the connection request and identifies a path to the destination (D) via an intermediate node (C). This is followed by a circuit connection phase handled by the switching nodes and initiated by allocating a free circuit to C (link BC), followed by transmission of a call request signal from node B to node C. In turn, node C allocates a link (CD) and the request is then passed to node D after a similar delay.

The circuit is then established and may be used. While it is available for use, resources (i.e. in the intermediate equipment at B and C) and capacity on the links between the equipment are dedicated to the use of the circuit.

After completion of the connection, a signal confirming circuit establishment (a connect signal in the diagram) is returned; this flows directly back to node A with no search delays since the circuit has been established. Transfer of the data in the message then begins. After data transfer, the circuit is disconnected; a simple disconnect phase is included after the end of the data transmission.

Delays for setting up a circuit connection can be high, especially if ordinary telephone equipment is used. Call setup time with conventional equipment is typically on the order of 5 to 25 seconds after completion of dialing. New fast circuit switching techniques can reduce delays. Trade-offs between circuit switching and other types of switching depend strongly on switching times.

Packet switching

Packet switching is a digital networking communications method that groups all transmitted data – regardless of content, type, or structure – into suitably sized blocks, called *packets*. Packet switching features delivery of variable-bit-rate data streams (sequences of packets) over a shared network. When traversing network adapters, switches, routers and other network nodes, packets are buffered and queued, resulting in variable delay and throughput depending on the traffic load in the network.

Packet switching contrasts with another principal networking paradigm, circuit switching, a method which sets up a limited number of dedicated connections of constant bit rate and constant delay between nodes for exclusive use during the communication session. In case of traffic fees (as opposed to flat rate), for example in cellular communication services, circuit switching is characterized by a fee per time unit of connection time, even when no data is transferred, while packet switching is characterized by a fee per unit of information.

Two major packet switching modes exist; (1) connectionless packet switching, also known as datagram switching, and (2) connection-oriented packet switching, also known as virtual circuit switching. In the first case each packet includes complete addressing or routing information. The packets are routed individually, sometimes resulting in different paths and out-of-order delivery. In the second case a connection is defined and preallocated in each involved node during a connection phase before any packet is transferred. The packets include a connection identifier rather than address information, and are delivered in order. See below.

Packet mode communication may be utilized with or without intermediate forwarding nodes (packet switches or routers). In all packet mode communication, network resources are managed by statistical multiplexing or dynamic bandwidth allocation in which a communication channel is effectively divided into an arbitrary number of logical variable-bit-rate channels or data streams. Statistical multiplexing, packet switching and other store-and-forward buffering introduces varying latency and throughput in the transmission. Each logical stream consists of a sequence of packets, which normally are forwarded by the multiplexers and intermediate network nodes asynchronously using first-in, first-out buffering. Alternatively, the packets may be forwarded according to some scheduling discipline for fair queuing, traffic shaping or for differentiated or guaranteed quality of service, such as weighted fair queuing or leaky bucket. In case of a shared physical medium, the packets may be delivered according to some packet-mode multiple access scheme.

Message switching

In telecommunications, **message switching** was the precursor of packet switching, where messages were routed in their entirety, one hop at a time. It was first introduced by Leonard Kleinrock in 1961. Message switching systems are nowadays mostly implemented over packet-switched or circuit-switched data networks. Each message is treated as a separate entity. Each message contains addressing information, and at each switch this information is read and the transfer path to the next switch is decided. Depending on network conditions, a conversation of several messages may not be transferred over the same path. Each message is stored (usually on hard drive due to RAM limitations) before being transmitted to the next switch. Because of this it is also known as a 'store-and-forward' network. Email is a common application for Message Switching. A delay in delivering email is allowed unlike real time data transfer between two computers.

Examples:

Hop-by-hop Telex forwarding and UUCP are examples of message switching systems.

When this form of switching is used, no physical path is established in advance in between sender and receiver. Instead, when the sender has a block of data to be sent, it is stored in the first switching office (i.e. router) then forwarded later at one hop at a time. Each block is received in its entity form, inspected for errors and then forwarded or re-transmitted.

A form of store-and-forward network. Data is transmitted into the network and stored in a switch. The network transfers the data from switch to switch when it is convenient to do so, as such the data is not transferred in real-time. Blocking can not occur, however, long delays can happen. The source and destination terminal need not be compatible, since conversions are done by the message switching networks.

A message switch is "transactional". It can store data or change its format and bit rate, then convert the data back to their original form or an entirely different form at the receive end. Message switching multiplexes data from different sources onto a common facility. A message switch is a one of the switching technology.

In message switching, when a message consists of a block of data to be sent, it is stored in the first switching office (i.e. router) and then forwarded later, one hop at a time. Each block is received in its entirety, inspected and later retransmitted. A network using this technique is referred to as a *store-and-forward* network.

Routing

Routing is the process of selecting paths in a network along which to send network traffic. Routing is performed for many kinds of networks, including the telephone network (circuit switching), electronic data networks (such as the Internet), and transportation networks. This article is concerned primarily with routing in electronic data networks using packet switching technology.

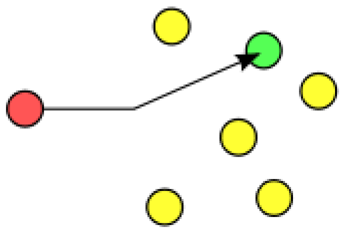
In packet switching networks, routing directs packet forwarding, the transit of logically addressed packets from their source toward their ultimate destination through intermediate nodes, typically hardware devices called routers, bridges, gateways, firewalls, or switches. General-purpose computers can also forward packets and perform routing, though they are not specialized hardware and may suffer from limited performance. The routing process usually directs forwarding on the basis of routing tables which maintain a record of the routes to various network destinations. Thus, constructing routing tables, which are held in the router's memory, is very important for efficient routing. Most routing algorithms use only one network path at a time, but multipath routing techniques enable the use of multiple alternative paths.

Routing, in a more narrow sense of the term, is often contrasted with bridging in its assumption that network addresses are structured and that similar addresses imply proximity within the network. Because structured addresses allow a single routing table entry to represent the route to a group of devices, structured addressing (routing, in the narrow sense) outperforms unstructured addressing (bridging) in large networks, and has become the dominant form of addressing on the Internet, though bridging is still widely used within localized environments.

Routing schemes differ in their delivery semantics. Some of the routing schemes are describes below:

Unicast

The term *unicast* is contrasted with the term *broadcast* which means transmitting the same data to all possible destinations. Another multi-destination distribution method, multicasting, sends data only to *interested* destinations by using special address assignments.



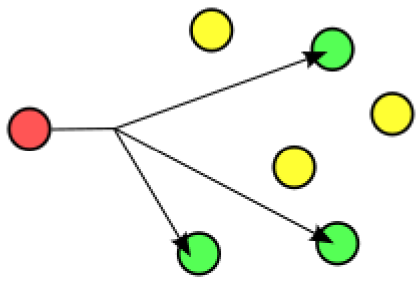
Unicast messaging is used for all network processes in which a private or unique resource is requested.

Certain network applications which are mass-distributed are too costly to be conducted with unicast transmission since each network connection consumes computing resources on the sending host and requires its own separate network bandwidth for transmission. Such applications include streaming media of many forms. Internet radio stations using unicast connections may have high bandwidth costs.

These terms are also used by streaming content providers' services. Unicast-based media servers open and provide a stream for each unique user. Multicast-based servers can support a larger audience by serving content simultaneously to multiple users.

Multicast

"Multicast" is sometimes also incorrectly used to refer to a multiplexed broadcast.



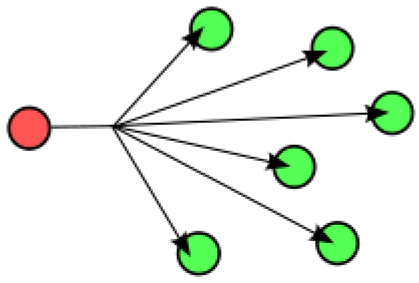
In computer networking, **multicast** is the delivery of a message or information to a group of destination computers simultaneously in a single transmission from the source. Copies are automatically created in other network elements, such as routers, but only when the topology of the network requires it.

Multicast is most commonly implemented in IP multicast, which is often employed in Internet Protocol (IP) applications of streaming media and Internet television. In IP multicast the implementation of the multicast concept occurs at the IP routing level, where routers create optimal distribution paths for datagrams sent to a multicast destination address.

At the Data Link Layer, *multicast* describes one-to-many distribution such as Ethernet multicast addressing, Asynchronous Transfer Mode (ATM) point-to-multipoint virtual circuits (P2MP) or Infiniband multicast.

Broadcast

In telecommunication and information theory, **broadcasting** refers to a method of transferring a message to all recipients simultaneously. Broadcasting can be performed as a high level operation in a program, for example broadcasting Message Passing Interface, or it may be a low level networking operation, for example broadcasting on Ethernet.



In computer networking, **broadcasting** refers to transmitting a packet that will be received by every device on the network. In practice, the scope of the broadcast is limited to a broadcast domain. Broadcast a message is in contrast to unicast addressing in which a host sends datagrams to another single host identified by a unique IP address.

Not all network technologies support broadcast addressing; for example, neither X.25 nor frame relay have broadcast capability, nor is there any form of Internet-wide broadcast. Broadcasting is largely confined to local area network (LAN) technologies, most notably

Ethernet and token ring, where the performance impact of broadcasting is not as large as it would be in a wide area network.

The successor to Internet Protocol Version 4 (IPv4), IPv6 also does not implement the broadcast method, so as to prevent disturbing all nodes in a network when only a few may be interested in a particular service. Instead it relies on multicast addressing a conceptually similar *one-to-many* routing methodology. However, multicasting limits the pool of receivers to those that join a specific multicast receiver group.

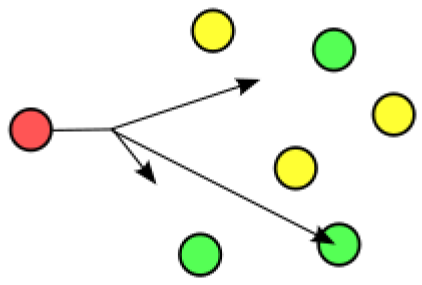
Both Ethernet and IPv4 use an all-ones broadcast address to indicate a broadcast packet. Token Ring uses a special value in the IEEE 802.2 control field.

Broadcasting may be abused to perform a DoS-attack. The attacker sends fake ping request with the source IP-address of the victim computer. The victim computer is flooded by the replies from all computers in the domain.

Anycast

From Wikipedia, the free encyclopedia

Jump to: navigation, search



Anycast is a network addressing and routing methodology in which datagrams from a single sender are routed to the topologically nearest node in a group of potential receivers all identified by the same destination address.

Anycast addressing routes datagrams to a single member of a group of potential receivers that are all identified by the same destination address. This is a *one-to-one-of-many* association.

On the Internet, anycast is usually implemented by using BGP to simultaneously announce the same destination IP address range from many different places on the Internet. This results in packets addressed to destination addresses in this range being routed to the "nearest" point on the net announcing the given destination IP address. In the past, anycast was suited to connectionless protocols (generally built on UDP), rather than connection-oriented protocols such as TCP that keep their own state. However, there are many cases where TCP anycast is now used. With TCP anycast, there are cases where the receiver selected for any given source may change from time to time as optimal routes change, silently breaking any conversations that may be in progress at the time. These conditions are typically referred to as a "pop switch". To correct for this issue, there have been proprietary advancements within custom IP stacks which allow for healing of stateful protocols where it is required. For this reason, anycast is generally used as a way to provide high availability and load balancing for stateless services such as access to replicated data; for example, DNS service is a distributed service over multiple geographically dispersed servers.

What Is The Difference Between Switching And Routing?

Routing and switching can be two terms that are difficult to differentiate, so here is a simple explanation that may help to clarify things. First of all switching and routing are not the same thing. Switching involves moving packets between devices on the same network. Conversely, routing involves moving packets between different networks.

Switches operate at layer 2 of the OSI Model. A switch, also referred to as a multi-port bridge, is able to determine where a packet should be sent by examining the MAC address within the data link header of the packet (the MAC address is the hardware address of a network adapter). A switch maintains a database of MAC addresses and what port they are connected to.

Routers, on the other hand, operate at layer 3 of the OSI Model. A router is able to determine where to send a packet using the Network ID within the Network layer header. It then uses the routing table to determine the route to the destination host.

Unit 2 : Server Administration Basic

2.1 Server and Client installation

Client

Client applications are installed on a user's computer or workstation, and interact with data and programs on a server. Client applications are not the same as desktop applications because client applications must interact with a server for full functionality. A common example of a client application is the video game World of Warcraft. Users install a client application on their computers that allows them to log into a server containing the game programming.

Businesses can use client server applications to cut down on overhead requirements for work stations. Instead of installing hundreds of copies of a particular program, users log into the application server.

Advantages and Disadvantages

Client server applications have an easier time handling complex printing requirements, on-screen updates and interface design. Client server applications have decreased security issues compared to web applications. Web applications make it easy to keep the application updated, moving between multiple computers, compatibility across different operating systems and initial deployment of the software.

The choice between client server applications and web applications usually depends on the needs of the user or business. Third-party web applications provide a ready-made solution for quick deployment. Some businesses prefer the customization available by producing client server applications.

Client vs Server Systems

Computers are needed in businesses of different sizes. Large computer setups that include networks and mainframes are used in large businesses. A computer network used in these types of businesses has a client-server architecture or two-tier architecture. The main purpose of this architecture is the division of labor which is required in large organizations.

Server

In client-server environment, the server computer acts as the “brains” of the business. A very large capacity computer is used as a server. There can be a mainframe also as it stores a wide variety of functionalities and data.

Generally, applications and data files are stored on the server computer. Employee computers or workstations access these applications and files across the network. For example, an employee can access company’s data files stored on the server, from his/her client computer.

In some cases, employees may access only specific applications from their client machine. Application server is the name given to this type of server. The client-server architecture is fully utilized in this type of environment as employees have to login from their client machine in order to access the application stored on the server. For example, these kinds of applications include graphic design programs, spreadsheets and word processors. The client- server architecture is illustrated in each case.

Apart from the storage medium, the server also acts as a processing power source. The client machines get their processing power from this server source. By doing so, no extra hardware for the client is needed and it utilizes greater processing power of the server.

Client

In client- server architecture, the client acts a smaller computer that is used by the employees of the organization in order to perform their day to day activities. The employee uses the client computer in order to access the data files or applications stored on the server machine.

The rights authorized to the client machine can be different. Some employees have the access to data files of the organization while other may only access the applications present on the server.

Apart from using the applications and data files, the client machine can also utilize the processing power of the server. In this case, the client computer is plugged-in to the server and the server machine handles all the calculations. In this way, the large processing power of the server can be utilized without any addition of hardware on the client side.

The best example of client- server architecture is WWW or World Wide Web. Here the client is the browser installed on each computer and the information about different pages is stored on the server side from which the client or the user can access it.

Difference between client and server

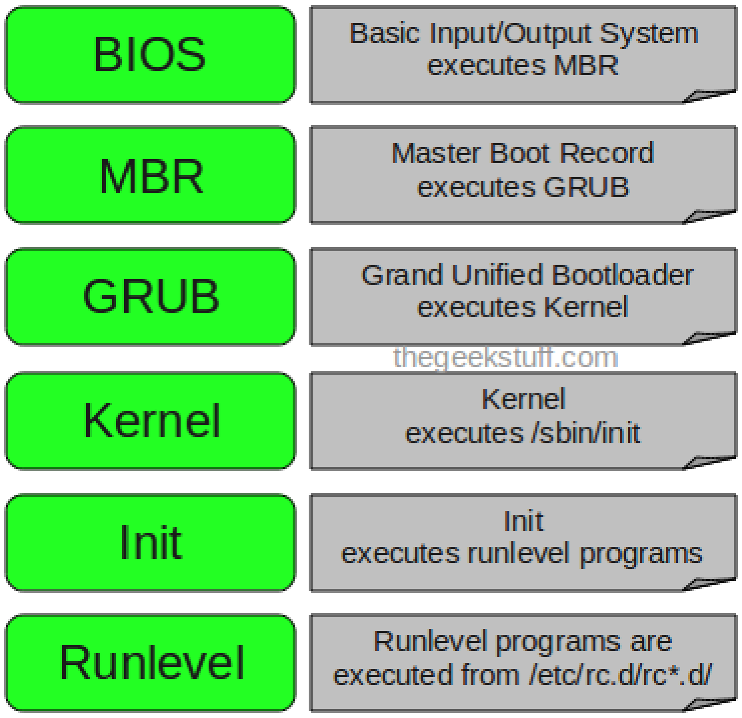
• Client is a smaller computer through which the information or application stored on the server is accessed by the user whereas server is a powerful computer that stores the data files and applications.

• In some cases, the client may utilize the greater processing power of the server machine.

• In some cases, the client side may have a better graphical user interface or GUI as compared to the server side.

2.2 Linux Boot Process (Startup Sequence)

Press the power button on your system, and after few moments you see the Linux login prompt.The following are the 6 high level stages of a typical Linux boot process.


BIOS
Basic Input/Output System executes MBR
MBR
Master Boot Record executes GRUB
GRUB
Grand Unified Bootloader executes Kernel
thegeekstuff.com
Kernel
Kernel executes /sbin/init
Init
Init executes runlevel programs
Runlevel
Runlevel programs are executed from /etc/rc.d/rc*.d/

1. BIOS

- BIOS stands for Basic Input/Output System
- Performs some system integrity checks
- Searches, loads, and executes the boot loader program.
- It looks for boot loader in floppy, cd-rom, or hard drive. You can press a key (typically F12 of F2, but it depends on your system) during the BIOS startup to change the boot sequence.
- Once the boot loader program is detected and loaded into the memory, BIOS gives the control to it.
- So, in simple terms BIOS loads and executes the MBR boot loader.

2. MBR

- MBR stands for Master Boot Record.
- It is located in the 1st sector of the bootable disk. Typically /dev/hda, or /dev/sda
- MBR is less than 512 bytes in size. This has three components 1) primary boot loader info in 1st 446 bytes 2) partition table info in next 64 bytes 3) mbr validation check in last 2 bytes.
- It contains information about GRUB (or LILO in old systems).
- So, in simple terms MBR loads and executes the GRUB boot loader.

3. GRUB

- GRUB stands for Grand Unified Bootloader.
- If you have multiple kernel images installed on your system, you can choose which one to be executed.
- GRUB displays a splash screen, waits for few seconds, if you don’t enter anything, it loads the default kernel image as specified in the grub configuration file.

- GRUB has the knowledge of the filesystem (the older Linux loader LILO didn't understand filesystem).
- Grub configuration file is /boot/grub/grub.conf (/etc/grub.conf is a link to this). The following is sample grub.conf of CentOS.

```
#boot=/dev/sda
default=0
timeout=5
splashimage=(hd0,0)/boot/grub/splash.xpm.gz
hiddenmenu
titleCentOS (2.6.18-194.el5PAE)
root (hd0,0)
kernel /boot/vmlinuz-2.6.18-194.el5PAE ro root=LABEL=/
initrd /boot/initrd-2.6.18-194.el5PAE.img
```

- As you notice from the above info, it contains kernel and initrd image.
- So, in simple terms GRUB just loads and executes Kernel and initrd images.

4. Kernel

- Mounts the root file system as specified in the "root=" in grub.conf
- Kernel executes the /sbin/init program
- Since init was the 1st program to be executed by Linux Kernel, it has the process id (PID) of 1. Do a 'ps -ef | grepinit' and check the pid.
- initrd stands for Initial RAM Disk.
- initrd is used by kernel as temporary root file system until kernel is booted and the real root file system is mounted. It also contains necessary drivers compiled inside, which helps it to access the hard drive partitions, and other hardware.

5. Init

- Looks at the /etc/inittab file to decide the Linux run level.
- Following are the available run levels

Runlevel	Target	Description
0	runlevel0.target , poweroff.target	Halt
1	runlevel1.target, rescue.target	Single-user mode
2	runlevel2.target	Not used (user-definable)
3	runlevel3.target , multi-user.target	Full multi-user mode (no GUI interface)
4	runlevel4.target	Not used (user-definable)
5	runlevel5.target, graphical.target	Full multiuser mode (with GUI interface)
6	runlevel6.target, reboot.target	Reboot

- Init identifies the default initlevel from /etc/inittab and uses that to load all appropriate program.
- Execute 'grepinitdefault /etc/inittab' on your system to identify the default run level
- If you want to get into trouble, you can set the default run level to 0 or 6. Since you know what 0 and 6 means, probably you might not do that.
- Typically you would set the default run level to either 3 or 5.

6. Runlevel programs

- When the Linux system is booting up, you might see various services getting started. For example, it might say "starting sendmail …. OK". Those are the runlevel programs, executed from the run level directory as defined by your run level.
- Depending on your default init level setting, the system will execute the programs from one of the following directories.
    - Run level 0 – /etc/rc.d/rc0.d/
    - Run level 1 – /etc/rc.d/rc1.d/
    - Run level 2 – /etc/rc.d/rc2.d/
    - Run level 3 – /etc/rc.d/rc3.d/
    - Run level 4 – /etc/rc.d/rc4.d/
    - Run level 5 – /etc/rc.d/rc5.d/
    - Run level 6 – /etc/rc.d/rc6.d/
- Please note that there are also symbolic links available for these directory under /etc directly. So, /etc/rc0.d is linked to /etc/rc.d/rc0.d.
- Under the /etc/rc.d/rc*.d/ directories, you would see programs that start with S and K.
- Programs starts with S are used during startup. S for startup.
- Programs starts with K are used during shutdown. K for kill.
- There are numbers right next to S and K in the program names. Those are the sequence number in which the programs should be started or killed.

For example, S12syslog is to start the syslog deamon, which has the sequence number of 12. S80sendmail is to start the sendmail daemon, which has the sequence number of 80. So, syslog program will be started before sendmail.

Overview of typical process

In Linux, the flow of control during a boot is from BIOS, to boot loader, to kernel. The kernel then starts the scheduler (to allow multi-tasking) and runs the first userland (i.e. outside kernel space) program Init (which is mostly responsible to run startup scripts for each runlevel), at which point the kernel goes idle unless called externally.

init (short for initialization) is a program for Unix-based computer operating systems that spawns all other processes. It runs as a daemon and typically has PID 1. The boot loader starts the kernel and the kernel starts init. If one were to delete init without a replacement, the system would encounter a kernel panic on the next reboot.

3.1. Introduction to Users and Groups

While users can be either people (meaning accounts tied to physical users) or accounts which exist for specific applications to use, groups are logical expressions of organization, tying users together for a common purpose. Users within a group can read, write, or execute files owned by that group.

Each user is associated with a unique numerical identification number called a *user ID*(UID). Likewise, each group is associated with a *group ID*(GID). A user who creates a file is also the owner and group owner of that file. The file is assigned separate read, write, and execute permissions for the owner, the group, and everyone else. The file owner can be changed only by `root`, and access permissions can be changed by both the `root` user and file owner.

Table 3.2. useradd command line options

Option	Description
`-c` '*comment*'	*comment* can be replaced with any string. This option is generally used to specify the full name of a user.
`-d` *home_directory*	Home directory to be used instead of default `/home/`*username*`/`.
`-e` *date*	Date for the account to be disabled in the format YYYY-MM-DD.
`-f` *days*	Number of days after the password expires until the account is disabled. If `0` is specified, the account is disabled immediately after the password expires. If `-1` is specified, the account is not be disabled after the password expires.
`-g` *group_name*	Group name or group number for the user's default group. The group must exist prior to being specified here.
`-G` *group_list*	List of additional (other than default) group names or group numbers, separated by commas, of which the user is a member. The groups must exist prior to being specified here.
`-m`	Create the home directory if it does not exist.
`-M`	Do not create the home directory.
`-N`	Do not create a user private group for the user.
`-p` *password*	The password encrypted with `crypt`.
`-r`	Create a system account with a UID less than 500 and without a home directory.
`-s`	User's login shell, which defaults to `/bin/bash`.
`-u` *uid*	User ID for the user, which must be unique and greater than 499.

3.3.1. Adding a New User

To add a new user to the system, typing the following at a shell prompt as `root`:

```
useradd [options] username
```

...where *options* are command line options as described in Table 3.2, "useradd command line options".

By default, the `useradd` command creates a locked user account. To unlock the account, run the following command as `root` to assign a password:

```
passwdusername
```

3.3.2. Adding a New Group

To add a new group to the system, type the following at a shell prompt as `root`:

```
groupadd [options] group_name
```

...where *options* are command line options as described in Table 3.3, "groupadd command line options"

Table 3.3. groupadd command line options

Option	Description
`-f`, `--force`	When used with `-g` *gid* and *gid* already exists, `groupadd` will choose another unique *gid* for the group.
`-g` *gid*	Group ID for the group, which must be unique and greater than 499.
`-K`, `--key` *key*=*value*	Override `/etc/login.defs` defaults.
`-o`, `--non-unique`	Allow to create groups with duplicate.
`-p`, `--password` *password*	Use this encrypted password for the new group.
`-r`	Create a system group with a GID less than 500.

INETD

inetd (**i**nter**net** service **d**aemon) is a super-serverdaemon on many Unix systems that manages Internet services. First appearing in 4.3BSD[1], it is generally located at `/usr/sbin/inetd`.

Often called a super-server, inetd listens on designated ports used by Internet services such as FTP, POP3, and telnet. When a TCP packet or UDP packet arrives with a particular destination port number, inetd launches the appropriate server program to handle the connection.

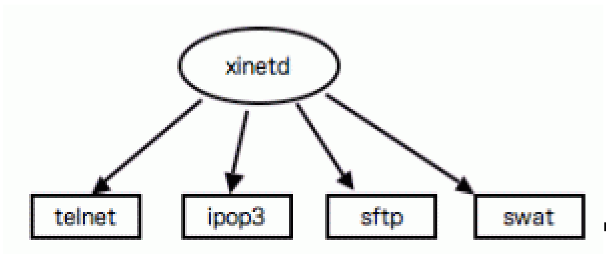
For services that are not expected to run with high loads, this method uses memory more efficiently, since the specific servers run only when needed. Furthermore, no network code is required in the application-specific daemons, as inetd hooks the sockets directly to stdin, stdout and stderr of the spawned process. For protocols that have frequent traffic, such as HTTP and POP3, a dedicated server that intercepts the traffic directly may be preferable.

inetd replacements

In recent years, because of the security limitations in the original design of inetd, it has been replaced by xinetd, rlinetd, ucspi-tcp, and others in many systems. Distributions of Linux especially have many options and Mac OS X (beginning with Mac OS X v10.2) uses xinetd. As of version Mac OS X v10.4, Apple has merged the functionality of inetd into launchd.

The services provided by inetd can be omitted entirely. This is becoming more common where machines are dedicated to a single function. For example, an HTTP server could be configured to just run httpd and have no other ports open. A dedicated firewall could have no services started.

inetd Vs xinetd in linux



The inetd daemon

The inetd daemon is best described as a super-server, created to manage many daemons or services. It listens to multiple ports, and invokes only requested services.

This reduces the load that services place on a system, because it means that network services – such as telnet, File Transfer Protocol (FTP), and Simple Mail Transfer Protocol (SMTP) – can be activated on demand rather than having to run continuously.

When a system initializes, the inetd daemon needs to determine its configuration information. To do this, the daemon accesses two files – /etc/services and /etc/inetd.conf.

The /etc/services file contains a list of network services and the ports to which they map. Network services have standard default ports on which client machines can find them. This enables clients to connect to particular services. For instance, HTTP uses port 80, FTP uses port 21, and telnet uses port 23.

Entries in the /etc/services file are divided into 4 fields,

- **servicename** – The servicename field specifies the name of a network service. Typical entries include telnet, File Transfer Protocol (FTP), and Post Office Protocol (POP).
- **port/protocol** – The port/protocol field identifies the number of the port and the protocol – or communication standard – that a service uses. The protocol is usually Transmission Control Protocol (TCP) or User Datagram Protocol (UDP). Both port and protocol entries should be included.
- **aliases** – The aliases field specifies any alternative names for a network service. For instance, a common alias for www is HTTP. This field is optional.
- **Comment** – The comment field is an optional field in which you can enter any additional information about a service.

e.g.

```
service-name port/protocol [aliases ...] [# comment]
pop2 109/tcp pop-2 postoffice # pop version
pop2 109/udp pop-2
pop3 110/tcp pop-3 postoffice # pop version 3
pop3 110/udp pop-3 postoffice
```

Another configuration file, **/etc/inetd.conf**, contains a list of network services that tell inetd which ports to listen to and which server programs to run in response to service requests. Each entry in the /etc/inetd.conf file consists of seven fields.

- **service_name** – The service_name field identifies a network service. The service name should be listed as it is defined in the /etc/services file. Any internal service should be given its official name.
- **socket_type** – The socket_type field identifies the type of port or virtual connection – usually stream or dgram (datagram) – that a network service uses. TCP services use stream and UDP services use dgram. Less common socket type values include raw, rdm, and seqpacket.
- **protocol** – The protocol field identifies the communication standard that a network service uses. Its value is usually TCP or UDP. This field should contain one of the protocols listed in the /etc/protocols file and should match the protocol field of a service's entry in the /etc/services file.
- **flags** – The flags field specifies the behavior of inetd after it invokes a server in response to a service request. Multithreaded TCP servers free the port after it starts and enable inetd to invoke a new instance of the server to handle new requests. Single-threaded UDP servers cannot free the port until the server has finished running. So you should always set dgram sockets to wait and all other socket types to nowait.
- **user** – The user field identifies the user account under which a service must run. The username should be defined in the /etc/passwd file and is often set to root. You can also specify a group name by placing a dot between the user and group names.
- **server_path** - The server_path field contains the pathname of the server that the inetd daemon must invoke when it receives a service request. In most cases, this field reads /usr/sbin/tcpd, where tcpd is responsible for controlling access to network services. If the

network service is one of inetd's local services, the entry for this field should also be local.

- **arguments** – The arguments field enables you to specify command-line arguments for the server daemon. The first argument, argv[0], is the daemon name and is always present, except in the case of internal services. If the server daemon is tcpd, an argument must specify the name of the service that tcpd must invoke.

e.g.

```
ftp       stream   tcp   nowait   root         /usr/sbin/tcpd
#telnet   stream   tcp   nowait   root         /usr/sbin/tcpd
shell     stream   tcp   nowait   root         /usr/sbin/tcpd
#login    stream   tcp   nowait   root         /usr/sbin/tcpd
talk      dgram    udp   wait     nobody.tty   /usr
ntalk     dgram    udp   wait     nobody.tty   /usr
finger stream tcp nowait nobody /usr/sbin/tcpd
```

TIP: single command to check currently enabled services from /etc/inted.conf

```
[root@easynomad1 grosetti]# grep -v "^#" /etc/inetd.conf
```

Whenever we modify the /etc/inetd.conf file, you need to refresh inetd to implement the new settings.

To do this, you can use the stop command and then the start command. Alternatively, you can use either the restart or the reload commands. One drawback of the inetd daemon is that it needs to load and initialize a new process before it can handle a new request. This slows down response times.

So if a service has a high number of connections and requires high data processing, you should run it as a standalone daemon rather than using inetd to invoke it. Suppose that you want inetd to invoke the telnet service. Before you do this, you first need to check which inetd services are currently enabled.

File System

1) In a computer, a file system (sometimes written filesystem) is the way in which files are named and where they are placed logically for storage and retrieval. The DOS, Windows, OS/2, Macintosh, and UNIX-based operating systems all have file systems in which files are placed somewhere in a hierarchical (tree) structure. A file is placed in a directory (*folder* in Windows) or subdirectory at the desired place in the tree structure.

File systems specify conventions for naming files. These conventions include the maximum number of characters in a name, which characters can be used, and, in some systems, how long the file name suffix can be. A file system also includes a format for specifying the path to a file through the structure of directories.

2) Sometimes the term refers to the part of an operating system or an added-on program that supports a file system as defined in (1). Examples of such add-on file systems include the Network File System (NFS) and the Andrew file system (AFS).

3) In the specialized lingo of storage professionals, a file system is the hardware used for nonvolatile storage , the software application that controls the hardware, and the architecture of both the hardware and software

Unix - File System Basics

A file system is a logical collection of files on a partition or disk. A partition is a container for information and can span an entire hard drive if desired.

Your hard drive can have various partitions which usually contains only one file system, such as one file system housing the / file system or another containing the /home file system.

One file system per partition allows for the logical maintenance and management of differing file systems.

Everything in Unix is considered to be a file, including physical devices such as DVD-ROMs, USB devices, floppy drives, and so forth.

Directory Structure:

Unix uses a hierarchical file system structure, much like an upside-down tree, with root (/) at the base of the file system and all other directories spreading from there.

A UNIX filesystem is a collection of files and directories that has the following properties:

- It has a root directory (/) that contains other files and directories.
- Each file or directory is uniquely identified by its name, the directory in which it resides, and a unique identifier, typically called an inode.
- By convention, the root directory has an inode number of 2 and the lost+found directory has an inode number of 3. Inode numbers 0 and 1 are not used. File inode numbers can be seen by specifying the -i option to ls command.
- It is self contained. There are no dependencies between one filesystem and any other.

The directories have specific purposes and generally hold the same types of information for easily locating files. Following are the directories that exist on the major versions of Unix:

Directory	Description
/	This is the root directory which should contain only the directories needed at the top level of the file structure.

/bin	This is where the executable files are located. They are available to all user.
/dev	These are device drivers.
/etc	Supervisor directory commands, configuration files, disk configuration files, valid user lists, groups, ethernet, hosts, where to send critical messages.
/lib	Contains shared library files and sometimes other kernel-related files.
/boot	Contains files for booting the system.
/home	Contains the home directory for users and other accounts.
/mnt	Used to mount other temporary file systems, such as cdrom and floppy for the CD-ROM drive and floppy diskette drive, respectively
/proc	Contains all processes marked as a file by process number or other information that is dynamic to the system.

Disk quota

A **disk quota** is a limit set by a system administrator that restricts certain aspects of file system usage on modern operating systems. The function of using disk quotas is to allocate limited disk space in a reasonable way.

Disk quotas are typically implemented on a per-user or per-group basis. That is, a system administrator defines a usage or file quota specific to a certain user or group.

In doing so, an administrator can prevent one user from consuming an entire file system's resources, or create a system of tiered access, whereby users can have different levels of restriction. This is used, for example, by web hosting companies to provide different levels of service based upon the needs and means of individual clients.

In most cases, quotas are also specific to individual file systems. Should an administrator want to limit the usage of a specific user on all file systems, a separate quota would have to be specified on each.

When a soft quota is violated, the system normally sends the user (and sometimes the administrator as well) some sort of message. No further action is typically taken.

Some systems prevent disk write operations that would result in hard quota violations from completing successfully, while others wait until the quota has been physically violated before denying write requests. The user is typically notified through the failed write operation error messages generated by the violating applications, while the administrator is almost always sent a notification as well.

Disk quotas are supported by most modern operating systems, including Unix-like systems, such as AIX (using JFS or JFS2 filesystem), Linux (using ext3, ext4, ext2, xfs (integrated support) among other filesystems), Solaris (using UFS or ZFS), Microsoft Windows starting with Windows 2000, Novell NetWare, VMS, and others. The method of administration for disk quotas varies between each of these operating systems. Unix-like systems typically provide a `quota` command for both administration and monitoring; graphical front-ends to the command may also be used. Unix and Unix-like operating systems frequently feature a grace period where users may violate their quota for a brief period of time. Windows 2000 and newer versions use the "Quota" tab of the disk properties dialog. Other systems provide their own quota management utilities.

Types of quotas

There are two basic types of disk quotas. The first, known as a *usage quota* or *block quota*, limits the amount of disk space that can be used. The second, known as a *file quota* or *inode quota*, limits the number of files and directories that can be created.

In addition, administrators usually define a warning level, or *soft quota*, at which users are informed they are nearing their limit, that is less than the effective limit, or *hard quota*. There may also be a small *grace interval*, which allows users to temporarily violate their quotas by certain amounts if necessary.

Common Unix Disk Quota Utilities

- quota - display a user's file system disk quota and usage;
- edquota - Edit user quotas for file system;
- repquota - Summarize quotas for a file system;
- quotacheck - File system quota consistency checker;
- quotaon - Turn file system quotas on and off;
- /etc/fstab (Linux) or /etc/vfstab (Solaris) - list of default parameters for each file system including quota status.

Job scheduler

A job scheduler is a program that enables an enterprise to schedule and, in some cases, monitor computer "batch" jobs (units of work, such as the running of a payroll program). A job scheduler can initiate and manage jobs automatically by processing prepared job control language statements or through equivalent interaction with a human operator. Today's job schedulers typically provide a graphical user interface and a single point of control for all the work in a distributed network of computers.

Some features that may be found in a job scheduler include:

- Continously automatic monitoring of jobs and completion notification
- Event-driven job scheduling
- Performance monitoring
- Report scheduling

Crontab

cron is the time-based job scheduler in Unix-like computer operating systems. cron enables users to schedule jobs (commands or shell scripts) to run periodically at certain times or dates. It is commonly used to automate system maintenance or administration, though its general-purpose nature means that it can be used for other purposes, such as connecting to the Internet and downloading email.

Cron is driven by a *crontab* (cron table) file, a configuration file that specifies shell commands to run periodically on a given schedule. The crontab files are stored where the lists of jobs and other instructions to the cron daemon are kept. Users can have their own individual crontab files and often there is a system wide crontab file (usually in `/etc` or a subdirectory of `/etc`) which only system administrators can edit.

Note: In Unix and other multitasking computer operating systems, a **daemon** (/ˈdeɪmən/ or /ˈdiːmən/)[1] is a computer program that runs as a backgroundprocess, rather than being under the direct control of an interactive user. Typically daemon names end with the letter *d*: for example, `syslogd` is the daemon that implements the system logging facility and sshd is a daemon that services incoming SSH connections. Example -

The following line causes the user program `test.pl` – ostensibly a Perl script – to be run every two hours, namely at midnight, 2am, 4am, 6am, 8am, and so on:

```
0*/2***/home/username/test.pl
```

Predefined scheduling definitions

There are several special predefined values which can be used to substitute the CRON expression.

Entry	Description	Equivalent To
`@yearly (or @annually)`	Run once a year, midnight, Jan. 1st	`0 0 1 1 *`
`@monthly`	Run once a month, midnight, first of month	`0 0 1 * *`
`@weekly`	Run once a week, midnight on Sunday	`0 0 * * 0`
`@daily`	Run once a day, midnight	`0 0 * * *`
`@hourly`	Run once an hour, beginning of hour	`0 * * * *`
`@reboot`	Run at startup	`@reboot`

1. **Cron (crond)** is the utility that runs continuously and initiates scheduled jobs defined by various parameters. In most systems there are multiple ways to run jobs on a periodic basic and that makes life more, not less, complicated. Jobs scheduled by cron assume the system is running 24 hours per day. If your job was scheduled at a time the system was not running your job does not get run. Sad but true.
2. **Anacron** is the utility that will run scheduled jobs if your system is not running 24 hours per day and these are defined using /etc/anacron. It uses a more modest time specification than cron. Most Linux distros seem to install anacron by default. BSD does not - install from usr/ports/sysutils/anacron. Use **man anacron** to get more infomation. Not discussed further.
3. **Crontab** is the method used to schedule tasks for the logged in user (more information use **man crontab**). To add a task, a file of the following format must be created:

```
SHELL=/path/to/shell
MAILTO=email-address
# if MAILTO="" (empty) mail is suppressed
HOME=/path/to/somewhere (normally /var/log)
min hour date mo dow task
# for details of time formats seecron time
# task is executed by the defined shell so may be a command or a script
```

Once the crontab has been created it must be added to the task list by running crontab as shown:

```
# for the logged in user
crontab /path/to/cron/entry

# to force a particular user
crontab -u user /path/to/cron/entry

# to list crontabs
crontab -l

# to remove crontabs
crontab -r

# to change crontabs either delete and add again or use
crontab -u user -e
# this invokes the standard editor (EDITOR). On completion of the edit
# the task is updated. Just editing the original file will not
# cause crontab to recognize the changes
```

The cron job runs under the user that issued the crontab command and thus permissions need to be carefully examined.

Howto set-up a crontab file

In Linux, Cron is a daemon/service that executes shell commands periodically on a given schedule. Cron is driven by a crontab, a configuration file that holds details of what commands are to be run along with a timetable of when to run them.

Creating a crontab file

You can create a crontab file by entering the following terminal command:

```
crontab -e
```

Entering the above command will open a terminal editor with a new blank crontab file, or it will open an existing crontab if you already have one. You can now enter the commands to be executed, see syntax below, before saving the file and exiting the editor. As long as your entries were entered correctly your commands should now be executed at the times/dates you specified. You can see a list of active crontab entries by entering the following terminal command:

```
crontab -l
```

Crontab syntax

A crontab file has six fields for specifying minute, hour, day of month, month, day of week and the command to be run at that interval. See below:

```
*     *     *     *     *  command to be executed
-     -     -     -     -
|     |     |     |     |
|     |     |     |     +----- day of week (0 - 6) (Sunday=0)
|     |     |     +------- month (1 - 12)
|     |     +--------- day of month (1 - 31)
|     +----------- hour (0 - 23)

+------------- min (0 - 59)
```

Crontab examples

Writing a crontab file can be a somewhat confusing for first time users, therefore I have listed below some crontab examples:

```
* * * * * <command> #Runs every minute

30 * * * * <command> #Runs at 30 minutes past the hour

45 6 * * * <command> #Runs at 6:45 am every day

45 18 * * * <command> #Runs at 6:45 pm every day

00 1 * * 0 <command> #Runs at 1:00 am every Sunday

00 1 * * 7 <command> #Runs at 1:00 am every Sunday

00 1 * * Sun <command> #Runs at 1:00 am every Sunday

30 8 1 * * <command> #Runs at 8:30 am on the first day of every month

00 0-23/2 02 07 * <command> #Runs every other hour on the 2nd of July
```

As well as the above there are also special strings that can be used:

```
@reboot <command> #Runs at boot

@yearly <command> #Runs once a year [0 0 1 1 *]

@annually <command> #Runs once a year [0 0 1 1 *]

@monthly <command> #Runs once a month [0 0 1 * *]

@weekly <command> #Runs once a week [0 0 * * 0]

@daily <command> #Runs once a day [0 0 * * *]

@midnight <command> #Runs once a day [0 0 * * *]

@hourly <command> #Runs once an hour [0 * * * *]
```

Multiple commands

A double-ampersand "`&&`" can be used to run multiple commands consecutively. The following example would run `command_01` and then `command_02` once a day:

```
@daily <command_01>&&<command_02>
```

Disabling email notifications

By default a cron job will send an email to the user account executing the cronjob. If this is not needed put the following command at the end of the cron job line:

```
>/dev/null 2>&1
```

Specifying a crontab file to use

As mentioned at the top of this post, you can create a new crontab file with the "`crontab -e`" command. However, you may already have a crontab file, if you do you can set it to be used with the following command:

```
crontab -u <username><crontab file>
```

Therefore the following command…

```
crontab -u tux ~/crontab
```

…would set Tux's crontab file to that of the file named "crontab" residing in Tux's home directory.

Removing a crontab file

To remove your crontab file simply enter the following terminal command:

```
crontab -r
```

Anacron

Anacron is a task scheduler similar to cron except that it does not require the system to run continuously. It can be used to run the daily, weekly, and monthly jobs usually run by cron.

To use the Anacron service, you must have the `anacron` RPM package installed. To determine if the package is installed, use the command `rpm -q anacron`.

Configuration File

Anacron tasks are listed in the configuration file `/etc/anacron`. Each line in the configuration file corresponds to a task and has the format:

```
period   delay   job-identifier  command
```

- `period` — frequency (in days) to execute the command
- `delay` — delay time in minutes
- `job-identifier` — description of the task, used in Anacron messages and as the name of the job's timestamp file, can contain any non-blank characters (except slashes).
- `command` — command to execute

For each tasks, Anacron determines if the task has been executed within the period specified in the `period` field of the configuration file. If it has not been executed within the given period, Anacron executes the command specified in the `command` field after waiting the number of minutes specified in the `delay` field.

After the task is completed, Anacron records the date in a timestamp file in the `/var/spool/anacron` directory. Only the date is used (not the time), and the value of the `job-identifier` is used as the filename for the timestamp file.

Environment variables such as `SHELL` and `PATH` can be defined at the top of `/etc/anacron` as with the cron configuration file.

The default configuration file looks similar to the following:

```
# /etc/anacrontab: configuration file for anacron

# See anacron(8) and anacrontab(5) for details.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# These entries are useful for a Red Hat Linux system.
1       5       cron.daily              run-parts /etc/cron.daily
7       10      cron.weekly             run-parts /etc/cron.weekly
30      15      cron.monthly    run-parts /etc/cron.monthly
```

Figure 22-1. Default anacrontab

As you can see in Figure 22-1, the anacrontab for Red Hat Linux is configured to make sure the daily, weekly, and monthly cron tasks are run.

Starting and Stopping the Service

To start the anacron service, use the command `/sbin/service anacron start`. To stop the service, use the command `/sbin/service anacron stop`. It is recommended that you start the service at boot time.

Log analysis

Log analysis (or *system and network log analysis*) is an art and science seeking to make sense out of computer-generated records (also called log or audit trail records). The process of creating such records is called data logging.

Typical reasons why people perform log analysis are:

- Compliance with security policies
- Compliance with audit or regulation
- System troubleshooting
- Forensics (during investigations or in response to subpoena)
- security incident response

Logs are emitted by network devices, operating systems, applications and all manner of intelligent or programmable device. A stream of messages in time-sequence often comprise a log. Logs may be directed to files, stored on disk, or directed as a network stream to a log collector.

Log messages must usually be interpreted with respect to the internal state of its source (e.g., application) and announce security-relevant or operations-relevant events (e.g., a user login, or a systems error).

Logs are often created by the software developers to aid in the debugging of the operation of the application. The syntax and semantics of data within log messages are usually application or vendor-specific. The authentication of a user to an application may be described as a login, a logon, a user connection or authentication event. Hence, log analysis must interpret messages within the context of an application, vendor, system or configuration in order to make useful comparisons to messages from different log sources.

Log message format or content may not always be fully documented. A task of the log analyst is to induce the system to emit the full range of messages in order to understand the complete domain from which the messages must be interpreted.

A log analyst may map varying terminology from different log sources into a uniform, normalized terminology so that reports and statistics can be derived from a heterogeneous environment. E.g., log messages from Windows, Unix, network firewalls, databases may be aggregated into a "normalized" report for the auditor.

Hence, log analysis practices exist on the continuum from text retrieval to reverse engineering of software.

Functions and technologies

Pattern recognition is a function of selecting incoming messages and compare with pattern book in order to filter or handle different way.

Normalization is the function of converting message parts to same format (e.g. common date format or normalized IP address).

Classification and tagging is order messages in different classes or tag them with different keywords for later usage (e.g. filtering or display).

Correlation analysis is a technology of collecting messages from different systems and finding all the messages belong to one single event (E.g. messages generated by malicious activity on different systems: network devices, firewalls, servers etc.). It is usually connected with alerting system.

Artificial Ignorance a process whereby you throw away the log entries you know aren't interesting. If there's anything left after you've thrown away the stuff you know isn't interesting, then the leftovers must be interesting. Artificial ignorance is a method to detect the anomalies in a working system. In log analysis, this means recognizing and ignoring the regular, common log messages that result from the normal operation of the system, and therefore are not too interesting. However, new messages that have not appeared in the logs before can sign important events, and should be therefore investigated.

Process management

Linux is a very dynamic system with constantly changing computing needs. The representation of the computational needs of Linux centers around the common abstraction of the *process*. Processes can be short-lived (a command executed from the command line) or long-lived (a network service). For this reason, the general management of processes and their scheduling is very important.

From user-space, processes are represented by process identifiers (PIDs). From the user's perspective, a PID is a numeric value that uniquely identifies the process. A PID doesn't change during the life of a process, but PIDs can be reused after a process dies, so it's not always ideal to cache them.

In user-space, you can create processes in any of several ways. You can execute a program (which results in the creation of a new process) or, within a program, you can invoke a `fork` or `exec` system call. The `fork` call results in the creation of a child process, while an `exec` call replaces the current process context with the new program. I discuss each of these methods to understand how they work.

For this article, I build the description of processes by first showing the kernel representation of processes and how they're managed in the kernel, then review the various means by which processes are created and scheduled on one or more processors, and finally, what happens if they die.

Linux Process Control

- accton - Turns process accounting on and off. Uses the file /var/log/pacct. To turn it on type "accton /var/log/pacct". Use the command with no arguments to turn it off.
- kill - Kill a process by number
- killall - Send a signal to a process by name
- lastcomm (1) - Display information about previous commands in reverse order. Works only if process accounting is on.
- nice - Set process priority of new processes.
- ps(1) - Used to report the status of one or more processes.
- pstree(1) - Display the tree of running processes.
- renice(8) - Can be used to change the process priority of a currently running process.
- sa(8) - Generates a summary of information about users' processes that are stored in the /var/log/pacct file.
- skill - Report process status.
- snice - Report process status.
- top - Displays the processes that are using the most CPU resources.

Checking running processes

While logged in as root, type "ps -ax |more" or "ps -aux |more". You will get a list of all processes running on your computer. You will see the process id (PID), process status (STAT) various statistics, and the command name. You can kill a process by typing "kill" and the PID number right afterwards similar to the line below.

kill 1721

You can also stop and restart processes by sending them various signals as in the below examples:

kill -STOP 1721	Stops (suspends) process 1721 by sending the STOP signal to the process. This process will still be on the task list. The process can't catch or ignore the STOP signal.
kill -CONT 1721	Continue process 1721 causing it to resume. The CONT signal is sent to the process.
kill -TERM 1721	Terminates process 1721 by sending the TERM signal to the process. This process will no longer show up on the task list if it is actually terminated. Process terminated cannot be continued. The TERM signal

	can be caught so TERM is not guaranteed to kill the process.
kill -HUP 1721	Stops, then restarts process 1721. This is usually done when a process is not working properly or the configuration files for that process have been changed. This command sends the HUP signal to the process which means hangup. This signal can be caught by the process.
killall -HUP myprint	Restarts any process with the name "myprint".
kill -TERM myprint	Terminates any process with the name "myprint".

Setting up and doing process control

The examples in this section use the "yes" command as an easy method for an example of a program that runs continually. The "yes" command outputs the string "y" until it is killed or stopped. When the output is ported to the /dev/null (null device or bit bucket), the output is basically dumped. Therefore this command is harmless, but is a good demonstration. To put the process in the background, append an "&" character to the end of the command as shown below.

```
yes> /dev/null &
```

The system will respond with a job number and process ID or PID similar to:

```
[1] 10419
```

Either number can be used to refer to the job. The "jobs" command can be used to check the job. When the command is entered the system will respond with a list of running jobs similar to the following:

```
[1]+ Running yes >/dev/null &
```

The job can be killed using the process ID or the job number. Either

```
kill %1
```

or:

```
kill 10419
```

Stopping and restarting jobs

Another way to put a job into the background is to

1. Start the job normally like:

   ```
   yes> /dev/null
   ```

   The prompt does not come back.

2. Use the <Ctrl-Z> key to stop the job.
3. Use the command "bg" or "bg %1" where 1 is the job number to put the process in the background. The system reports the job number when you stop the job.
   Before the last step, the job was suspended. The "fg" command could have been used to bring the job into the foreground rather than using the "bg" command to put it in the background. If the job is running in the foreground, you can type &@60Ctrl-C> to terminate the process.

Killing or Reconfiguring a Daemon without Restarting

killall -1 inetd	Restarts inetd by sending signal number 1 which is the hangup signal.
killall -HUP inetd	Causes the daemon to reload its config file by sending the hangup signal. The difference between this example and the previous one is the signal is called by name here rather than number.

To make changes to inetd:

1. Reconfigure /etc/inetd.conf
2. Restart inetd by sending it the hangup signal

The easy way to reset a service that was started via the rc script files during system startup:

1. Find the file for the service, you want to start. For example find the file for the print daemon "lpd". These files should typically be in the directory "/etc/rc.d/init.d". The file name in this case is "lpd". (Note this is a script file, that starts the daemon, not the actual binary daemon file).
2. Go to that subdirectory "cd /etc/rc.d/init.d" and type "./lpd restart".
3. You should get output to the screen that indicates this service has been shut down and then started.

Setting process priority

In Linux, processes have a priority number between -20 and 19. The value of -20 is the highest, and 19 is the lowest priority. Process priority can be set with the nice(1) command and changed using the renice(8) command. To set a process to have the highest priority find the process ID number using the ps command. If your process name is "myprog" type:

```
ps -ax |grepmyprog
```

You should get something like:

```
756 tty1 S 0:00 myprog
```

The first number on the line is your process ID. Enter the command:

renice -20 756

This will set your process (PID=756) to priority of -20. Modify the process ID number for that of your program running on your system. You can use the nice command to determine the default priority of new processes by typing "nice" on the command line. If you want to start a process with a specific priority, use the nice(1) command when you invoke the process.

Setting limits on the number of processes that can run

The command "ulimit" is used to limit the number of processes users can run along with available system resources. All processes which will be started from the shell (bash in many cases), will have the same resource limits. See the bash manual page for more information. To set the limits for daemons which are running at boot time add ulimit command to boot scripts.

The command "ulimit -a" reports the current limits.

Process management

Maximum processes

Although processes are dynamically allocated within Linux, certain maximums are observed. The maximum is represented in the kernel by a symbol called `max_threads`, which can be found in ./linux/kernel/fork.c). You can change this value from user-space through the proc file system at /proc/sys/kernel/threads-max.

Now, let's explore how you manage processes within Linux. In most cases, processes are dynamically created and represented by a dynamically allocated `task_struct`. One exception is the `init` process itself, which always exists and is represented by a statically allocated `task_struct`. You can see an example of this in ./linux/arch/i386/kernel/init_task.c.

All processes in Linux are collected in two different ways. The first is a hash table, which is hashed by the PID value; the second is a circular doubly linked list. The circular list is ideal for iterating through the task list. As the list is circular, there's no head or tail; but as the `init_task` always exists, you can use it as an anchor point to iterate further. Let's look at an example of this to walk through the current set of tasks.

The task list is not accessible from user-space, but you can easily solve that problem by inserting code into the kernel in the form of a module. A very simple program is shown in Listing 2 that iterates the task list and provides a small amount of information about each task (`name`, `pid`, and `parent` name). Note here that the module uses `printk` to emit the output. To view the output, you need to view the /var/log/messages file with the `cat` utility (or `tail -f /var/log/messages` in real time). The `next_task` function is a macro in sched.h that simplifies the iteration of the task list (returns a `task_struct` reference of the next task).

What is Internet Protocol -- IP?

IP (short for Internet Protocol) specifies the technical format of packets and the addressing scheme for computers to communicate over a network. Most networks combine IP with a higher-level protocol called Transmission Control Protocol (TCP), which establishes a virtual connection between a destination and a source.

IP by itself can be compared to something like the postal system. It allows you to address a package and drop it in the system, but there's no direct link between you and the recipient. TCP/IP, on the other hand, establishes a connection between two hosts so that they can send messages back and forth for a period of time.

There are currently two version of Internet Protocol (IP): ***IPv4*** and a new version called ***IPv6.*** IPv6 is an evolutionary upgrade to the Internet Protocol. IPv6 will coexist with the older IPv4 for some time.

What is IPv4 -- Internet Protocol Version 4?

IPv4 (***I****nternet* ***P****rotocol* ***V****ersion* ***4***) is the fourth revision of the Internet Protocol (IP) used to to identify devices on a network through an addressing system. The Internet Protocol is designed for use in interconnected systems of packet-switched computer communication networks (*see RFC:791*).

IPv4 is the most widely deployed Internet protocol used to connect devices to the Internet. IPv4 uses a 32-bit address scheme allowing for a total of 2^32 addresses (just over 4 billion addresses). With the growth of the Internet it is expected that the number of unused IPv4 addresses will eventually run out because every device -- including computers, smartphones and game consoles -- that connects to the Internet requires an address.

A new Internet addressing system Internet Protocol version 6 (IPv6) is being deployed to fulfill the need for more Internet addresses.

What is IPv6 -- Internet Protocol Version 6?

IPv6 (***I****nternet* ***P****rotocol* ***V****ersion* ***6***) is also called IPng (***I****nternet* ***P****rotocol* ***n****ext* ***g****eneration*) and it is the newest version of the Internet Protocol (IP) reviewed in the IETF standards committees to replace the current version of IPv4 (Internet Protocol Version 4).

IPv6 is the successor to Internet Protocol Version 4 (IPv4). It was designed as an evolutionary upgrade to the Internet Protocol and will, in fact, coexist with the older IPv4 for some time. IPv6 is designed to allow the Internet to grow steadily, both in terms of the number of hosts connected and the total amount of data traffic transmitted.

IPv6 is often referred to as the "next generation" Internet standard and has been under development now since the mid-1990s. IPv6 was born out of concern that the demand for IP addresses would exceed the available supply.

While increasing the pool of addresses is one of the most often-talked about benefit of IPv6, there are other important technological changes in IPv6 that will improve the IP protocol:

- No more NAT (Network Address Translation)
- Auto-configuration
- No more private address collisions
- Better multicast routing
- Simpler header format
- Simplified, more efficient routing
- True quality of service (QoS), also called "flow labeling"
- Built-in authentication and privacy support
- Flexible options and extensions
- Easier administration (say good-bye to DHCP)

The Difference Between IPv6 and IPv4 IP Addresses

1. An IPv6 address consists of 128 bits, while an IPv4 address consists of only 32.

2. IPv6 has a lot more usable addresses compared to IPv4.

3. IPv6 makes the router's task more simple compared to IPv4.

4. IPv6 is better suited to mobile networks than IPv4.

5. IPv6 addresses are represented in a hexadecimal, colon-separated notation, while IPv4 address use the dot-decimal notation.

6. IPv6 allows for bigger payloads than what is allowed in IPv4.

7. IPv6 is used by less than 1% of the networks, while IPv4 is still in use by the remaining 99%.

IPv4	**IPv6**
IPv4 addresses are 32 bit length.	IPv6 addresses are 128 bit length.
IPSec support is only optional. Inbuilt	IPSec support.
Fragmentation is done by sender and forwarding routers.	Fragmentation is done only by sender.
No packet flow identification.	Packet flow identification is available within the IPv6 header using the Flow Label field.
Checksum field is available in header. .	No checksum field in header
Options fields are available in header.	No option fields, but Extension headers are available.
Address Resolution Protocol (ARP) is available to map IPv4 addresses to MAC addresses.	Address Resolution Protocol (ARP) is replaced with Neighbor Discovery Protocol.
Internet Group Management Protocol (IGMP) is used to manage multicast group membership.	IGMP is replaced with Multicast Listener Discovery (MLD) messages.
Broadcast messages are available.	Broadcast messages are not available. Instead a link-local scope all-nodes multicast address is used for broadcast.

Manual configuration (Static) of IP addresses or DHCP (Dynamic configuration) is required to configure IP addresses.	Auto-configuration of addresses is available.

How to fix Linux boot problems

Booting, or "bootstrapping" for us older folk, is that deeply mysterious sequence of operations performed by your computer between the moment when you switch it on and the moment it's ready for you to log in. During this time, all kinds of incomprehensible messages scroll up the screen, but they're not something you usually take much notice of, and most linuxdistros cover them up with a pretty splash screen and a nice encouraging progress bar. This is all fine, of course, until it stops working. In this tutorial we'll examine the boot process in more detail, looking in particular at what can go wrong, and how to diagnose and fix the problem.

Grokking the problem

When I'm teaching Linux on one of my courses, many attendees tell me they are interested in troubleshooting of one form or another. Some of them are looking for a cookbook approach - "If you see the error message X, run command Y", but troubleshooting rarely works that way. My initial advice to anyone who needs to troubleshoot is always the same: "The most important thing in troubleshooting is to understand how the system is supposed to work in the first place. The second most important thing is figuring out exactly what the system was trying to do when it went wrong."

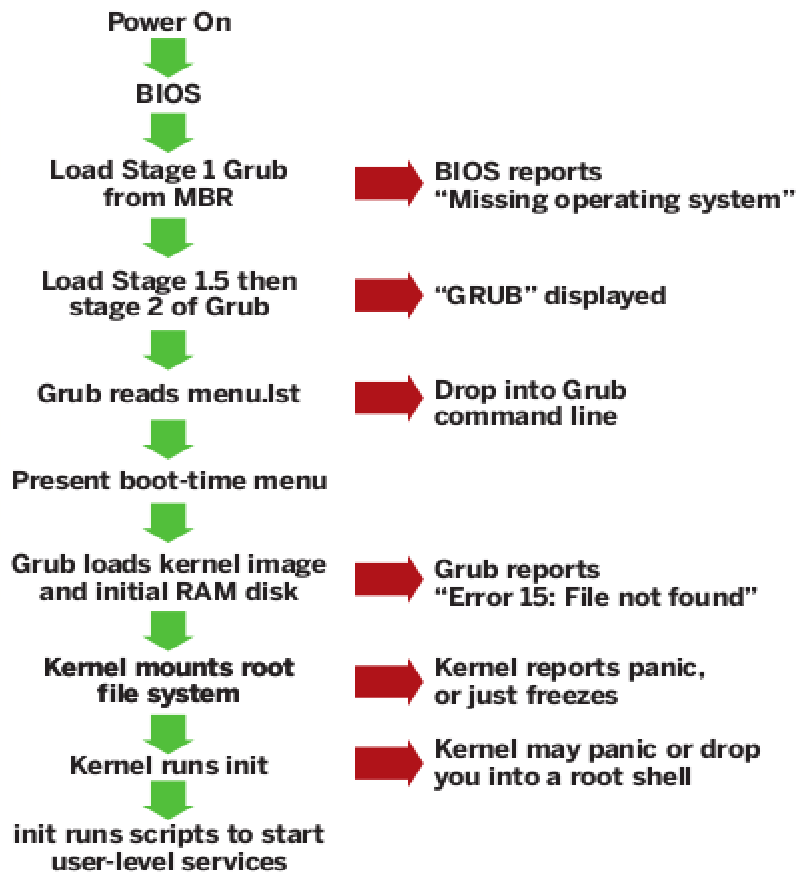

Power On
BIOS
Load Stage 1 Grub from MBR
BIOS reports "Missing operating system"
Load Stage 1.5 then stage 2 of Grub
"GRUB" displayed
Grub reads menu.lst
Drop into Grub command line
Present boot-time menu
Grub loads kernel image and initial RAM disk
Grub reports "Error 15: File not found"
Kernel mounts root file system
Kernel reports panic, or just freezes
Kernel runs init
Kernel may panic or drop you into a root shell
init runs scripts to start user-level services

Figure 1: the normal sequence of events when booting Linux.

With this in mind, let's take a look at how Linux boots. Knowing the normal sequence of events, and determining how far it got before it ran into trouble, are key to diagnosing and fixing boot-time problems. Figure 1 above, right shows the normal sequence of events (green arrows) and indicates some of the possible failure paths (red arrows).

Picking yourself up by your bootstraps

Booting is a multi-stage affair. When a PC is powered up, control initially passes to a program (called the BIOS) stored in read-only memory on the motherboard. The BIOS performs a self-test of the hardware and scouts around looking for a device to boot from. The BIOS provides configuration screens that allow you to assign the order in which it searches for a bootable device, and modern BIOSes support a wide range of boot devices, including PXE booting from a network server. The only case we consider here is booting from the hard drive.

The BIOS loads the Master Boot Record (MBR) of the selected boot device and executes it. (If this fails, the BIOS will report something like "Missing Operating System", and come to a screaming halt.) The MBR occupies the very first sector of the drive. It holds the drive's partition table (64 bytes) and a very short program (446 bytes) which is 'stage 1' of the bootstrap loader.

This stage 1 loader is pretty dumb - all it does is to display the word GRUB on the screen then load a second stage boot loader using a 'block map' that is embedded into the MBR. (The block map specifies the disk block numbers where the second stage loader resides). I'm assuming here that we're using the Grub boot loader. There's an earlier boot loader called Lilo, but Grub is more recent, smarter, and used in most modern Linux distros. The second stage of the Grub boot loader is actually called stage one-and-a-half, and if you list the directory /boot/grub you can see the files that contain the various versions of this; they have names like e2fs_stage_1_5 and reiserfs_stage_1_5.

Each of these programs is able to access files by name using one particular filesystem format. e2fs_stage_1_5 can read ext2 and ext3 filesystems, and reiserfs_stage_1_5 can read reiserfilesystems, and so on. Grub's ability to access files by name at boot time (before linux is running) is the thing that really sets it apart from Lilo. The stage one-and-a-half program loads Grub stage 2 which is considerably larger. This stage reads the Grub configuration file (usually /boot/grub/menu.lst or /boot/grub/grub.conf) and, based on the entries it finds there, it presents a menu of choices of the operating system you want to boot. If Grub can't find its config file it will drop down to an interactive command-line prompt to allow you to enter Grub commands manually.

A typical entry in menu.lst looks like this:

```
titleopenSUSE 10.2
root (hd0,0)
kernel /boot/vmlinuz-2.6.18.2-34-default root=/dev/hda1 vga=0x317 showopts
initrd /boot/initrd-2.6.18.2-34-default
```

The title line simply specifies the text that will appear in Grub's boot-time menu. The lines that follow specify the commands that Grub will execute if you select that item from the menu. The root line sets Grub's idea of where the root filesystem resides. Grub has its own way of naming disk partitions which is confusingly different from the naming scheme used by Linux.

In Grub-speak, hd0 refers to the first drive - on a typical PC with IDE drives this corresponds to the Linux device name /dev/hda, or, in some of the more recent distros, /dev/sda. In Grub-speak, (hd0,0) refers to the first partition on that drive. Linux would call this /dev/hda1 or /dev/sda1. The kernel line specifies the file that Grub will load as the Linux kernel; at the end of this line you may see some additional boot parameters that are passed to the kernel. More about these later.

The initrd line specifies the file that contains the 'initial RAM Disk' - a file system image that will be used by the kernel as it boots. Grub is also responsible for loading this file into memory. If Grub fails to find the kernel or the ramdisk images it will report Error 15: File not found, and halt.

Once the kernel starts running, it mounts the root file system from the hard drive. The name of the partition that holds this file system is passed as a parameter to the kernel, as you can see from the menu.lst entry above. Mounting the root file system is a key point in the boot process and if you're trying to pin down a boot-time problem it's vital to figure out if the kernel was able to get this far. Failure to mount the root file system will generally result in a kernel panic, though some systems just appear to halt.

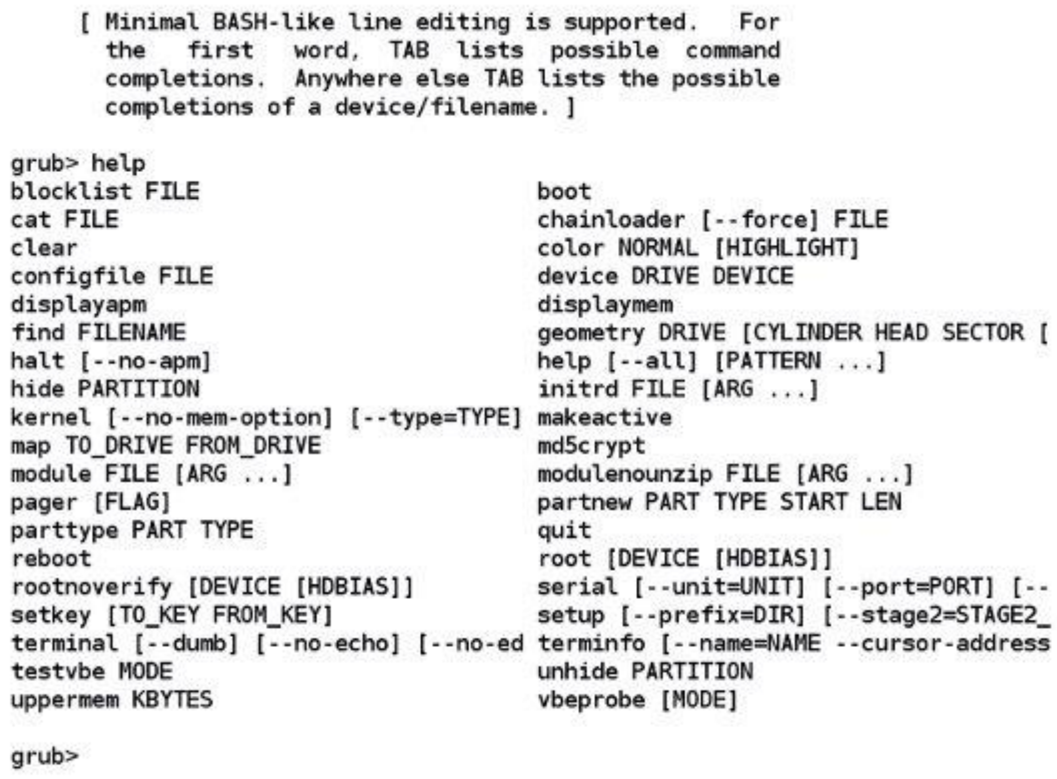
If the kernel succeeds in mounting the root file system, it creates a single process (with process ID 1) which runs the program /sbin/init. If the kernel can't find init, it will either panic and halt or (depending on the distro) drop you into a root shell. Oh, by the way, just to add a little confusion, Ubuntu doesn't use init any more, it uses a replacement called upstart.

The program init is responsible for running the scripts that will start all the other services in the system. There is one important and rather low-level script run by init early in the process. On Red Hat-style systems it's /etc/rc.d/rc.sysinit, on SUSE it's /etc/init.d/boot. Among other things, these early scripts consistency-check and mount any other disk partitions, as specified in /etc/fstab. Although there is certainly plenty of scope for things going wrong at this stage, we need to leave the story at that point, at least for this month.

Getting to grips with Grub

A key skill in fixing boot-time problems is knowing how to manually intervene in the Grub boot sequence. Most distros configure Grub to boot a default choice from the menu, but allow a short time window (a few seconds) in which you can press Esc to interrupt this and gain direct control of Grub. Typically this will exit from the Grub splash screen and drop to a character-based menu.

From here, follow the on-screen instructions to select an item from the menu, and edit the commands associated with that menu selection before booting. It's even possible to drop down to a Grub command prompt where you can enter Grub commands directly; for example at this point you could, in theory, manually enter the root, kernel and initrd lines from the menu.lst file we looked at earlier. Figure 2 (below) shows the result of typing help at the Grub command prompt.



```
      [ Minimal BASH-like line editing is supported.   For
        the   first   word,  TAB  lists  possible  command
        completions.  Anywhere else TAB lists the possible
        completions of a device/filename. ]

grub> help
blocklist FILE                        boot
cat FILE                              chainloader [--force] FILE
clear                                 color NORMAL [HIGHLIGHT]
configfile FILE                       device DRIVE DEVICE
displayapm                            displaymem
find FILENAME                         geometry DRIVE [CYLINDER HEAD SECTOR [
halt [--no-apm]                       help [--all] [PATTERN ...]
hide PARTITION                        initrd FILE [ARG ...]
kernel [--no-mem-option] [--type=TYPE] makeactive
map TO_DRIVE FROM_DRIVE               md5crypt
module FILE [ARG ...]                 modulenounzip FILE [ARG ...]
pager [FLAG]                          partnew PART TYPE START LEN
parttype PART TYPE                    quit
reboot                                root [DEVICE [HDBIAS]]
rootnoverify [DEVICE [HDBIAS]]        serial [--unit=UNIT] [--port=PORT] [--
setkey [TO_KEY FROM_KEY]              setup [--prefix=DIR] [--stage2=STAGE2_
terminal [--dumb] [--no-echo] [--no-ed terminfo [--name=NAME --cursor-address
testvbe MODE                          unhide PARTITION
uppermem KBYTES                       vbeprobe [MODE]

grub>
```

Figure 2: the result that is output when 'help' is typed at the Grub command prompt.

Rescue Booting

If no amount of tweaking with the Grub boot commands will allow your system to boot, it may be time to perform a 'rescue boot', which means that you'll boot Linux from an installation CD or other 'rescue media'. The kernel and its modules are loaded off the CD, along with a small file system that is built in memory. This results in a small but working Linux system that isn't dependent on any file systems on the hard drive.

You can then mount the hard drive's partitions into the file system and access them to repair the damage. The installation DVD or CD of almost any modern Linux distribution can be used for this purpose - there is no requirement that the rescue media is from the same distribution as the one you're trying to rescue.

It's important to keep a clear head when using a rescue system because the files on your hard drive won't be in the same place when they are mounted into the rescue system as when they're viewed by the 'real' installation. For example, if in the rescue system I were to mount my system's root partition onto /mnt, then the file I would normally see as /etc/fstab will be seen as /mnt/etc/fstab.

Network Interfaces

Under Red Hat Enterprise Linux, all network communications occur between configured software *interfaces* and *physical networking devices* connected to the system.

The configuration files for network interfaces are located in the `/etc/sysconfig/network-scripts/` directory. The scripts used to activate and deactivate these network interfaces are also located here. Although the number and type of interface files can differ from system to system, there are three categories of files that exist in this directory:

1. *Interface configuration files*
2. *Interface control scripts*
3. *Network function files*

The files in each of these categories work together to enable various network devices.

This chapter explores the relationship between these files and how they are used.

Network Configuration Files

Before delving into the interface configuration files, let us first itemize the primary configuration files used in network configuration. Understanding the role these files play in setting up the network stack can be helpful when customizing a Red Hat Enterprise Linux system.

The primary network configuration files are as follows:

`/etc/hosts`

> The main purpose of this file is to resolve hostnames that cannot be resolved any other way. It can also be used to resolve hostnames on small networks with no DNS server. Regardless of the type of network the computer is on, this file should contain a line specifying the IP address of the loopback device (`127.0.0.1`) as `localhost.localdomain`. For more information, refer to the `hosts` man page.

`/etc/resolv.conf`

> This file specifies the IP addresses of DNS servers and the search domain. Unless configured to do otherwise, the network initialization scripts populate this file. For more information about this file, refer to the `resolv.conf` man page.

`/etc/sysconfig/network`

> This file specifies routing and host information for all network interfaces. For more information about this file and the directives it accepts, refer to Section 30.1.21, “`/etc/sysconfig/network`”.

`/etc/sysconfig/network-scripts/ifcfg-`*`<interface-name>`*

> For each network interface, there is a corresponding interface configuration script. Each of these files provide information specific to a particular network interface. Refer to Section 15.2, “Interface Configuration Files” for more information on this type of file and the directives it accepts.

Interface Configuration Files

Interface configuration files control the software interfaces for individual network devices. As the system boots, it uses these files to determine what interfaces to bring up and how to configure them. These files are usually named `ifcfg-<name>`, where *<name>* refers to the name of the device that the configuration file controls.

Ethernet Interfaces

One of the most common interface files is `ifcfg-eth0`, which controls the first Ethernet *network interface card* or *NIC* in the system. In a system with multiple NICs, there are multiple `ifcfg-eth<X>`files (where *<X>* is a unique number corresponding to a specific interface). Because each device has its own configuration file, an administrator can control how each interface functions individually.

The following is a sample `ifcfg-eth0` file for a system using a fixed IP address:

```
DEVICE=eth0
BOOTPROTO=none
ONBOOT=yes
NETWORK=10.0.1.0
NETMASK=255.255.255.0
IPADDR=10.0.1.27
USERCTL=no
```

The values required in an interface configuration file can change based on other values. For example, the `ifcfg-eth0` file for an interface using DHCP looks different because IP information is provided by the DHCP server:

```
DEVICE=eth0
BOOTPROTO=dhcp
ONBOOT=yes
```

The **Network Administration Tool**(`system-config-network`) is an easy way to make changes to the various network interface configuration files.

However, it is also possible to manually edit the configuration files for a given network interface.

IPsec Interfaces

The following example shows the `ifcfg` file for a network-to-network IPsec connection for LAN A. The unique name to identify the connection in this example is `ipsec1`, so the resulting file is named `/etc/sysconfig/network-scripts/ifcfg-ipsec1`.

```
TYPE=IPsec
ONBOOT=yes
IKE_METHOD=PSK
SRCNET=192.168.1.0/24
DSTNET=192.168.2.0/24
DST=X.X.X.X
```

In the example above, *`X.X.X.X`* is the publicly routable IP address of the destination IPsec router.

Channel Bonding Interfaces

Red Hat Enterprise Linux allows administrators to bind multiple network interfaces together into a single channel using the `bonding` kernel module and a special network interface called a *channel* *bonding interface*. Channel bonding enables two or more network interfaces to act as one, simultaneously increasing the bandwidth and providing redundancy.

To create a channel bonding interface, create a file in the `/etc/sysconfig/network-scripts/` directory called `ifcfg-bond<N>`, replacing `<N>` with the number for the interface, such as `0`.

The contents of the file can be identical to whatever type of interface is getting bonded, such as an Ethernet interface. The only difference is that the `DEVICE=` directive must be `bond<N>`, replacing `<N>` with the number for the interface.

The following is a sample channel bonding configuration file, `ifcfg-bond0`:

```
DEVICE=bond0
IPADDR=192.168.1.1
NETMASK=255.255.255.0
ONBOOT=yes
BOOTPROTO=none
USERCTL=no
BONDING_OPTS="<bonding parameters separated by spaces>"
```

After the channel bonding interface is created, the network interfaces to be bound together must be configured by adding the `MASTER=` and `SLAVE=` directives to their configuration files. The configuration files for each of the channel-bonded interfaces can be nearly identical.

For example, if two Ethernet interfaces are being channel bonded, both `eth0` and `eth1` may look like the following example:

```
DEVICE=eth<N>
BOOTPROTO=none
ONBOOT=yes
MASTER=bond0
SLAVE=yes
USERCTL=no
```

In this example, replace `<N>` with the numerical value for the interface.

For a channel bonding interface to be valid, the kernel module must be loaded. To ensure that the module is loaded when the channel bonding interface is brought up, add the following line to `/etc/modprobe.conf`:

```
alias bond<N> bonding
```

Replace `<N>` with the number of the interface, such as `0`.

Alias and Clone Files

Two lesser-used types of interface configuration files are *alias* and *clone* files.

Alias interface configuration files, which are used to bind multiple addresses to a single interface, use the `ifcfg-<if-name>:<alias-value>`naming scheme.

For example, an `ifcfg-eth0:0` file could be configured to specify `DEVICE=eth0:0` and a static IP address of 10.0.0.2, serving as an alias of an Ethernet interface already configured to receive its IP information via DHCP in `ifcfg-eth0`. Under this configuration, `eth0` is bound to a dynamic IP address, but the same physical network card can receive requests via the fixed, 10.0.0.2 IP address.

Dialup Interfaces

If you are connecting to the Internet via a dialup connection, a configuration file is necessary for the interface.

PPP interface files are named using the following format:

`ifcfg-ppp<X>`

where *<X>* is a unique number corresponding to a specific interface.

The PPP interface configuration file is created automatically when `wvdial`, the **Network Administration Tool** or **Kppp** is used to create a dialup account. It is also possible to create and edit this file manually.

The following is a typical `ifcfg-ppp0` file:

```
DEVICE=ppp0
NAME=test
WVDIALSECT=test
MODEMPORT=/dev/modem
LINESPEED=115200
PAPNAME=test
USERCTL=true
ONBOOT=no
PERSIST=no
DEFROUTE=yes
PEERDNS=yes
DEMAND=no
IDLETIMEOUT=600
```

Serial Line Internet Protocol (SLIP) is another dialup interface, although it is used less frequently. SLIP files have interface configuration file names such as `ifcfg-sl0`.

Other Interfaces

Other common interface configuration files include the following:

`ifcfg-lo`

A local *loopback interface* is often used in testing, as well as being used in a variety of applications that require an IP address pointing back to the same system. Any data sent to the loopback device is immediately returned to the host's network layer.

Firewall setup for Windows XP Firewall

Windows XP includes a firewall built-in, which is enabled by default if Service Pack 2 is installed. If you don't wish to get a separate firewall such as Kerio Personal Firewall or ZoneAlarm, then you should make sure it is in use.

For instructions on how to enable ping, read the relevant section below.

Original and Service Pack 1

In these versions of XP, the firewall is disabled by default. Therefore it must be enabled, and then set to allow ping.

Click "Start" -> "Control Panel".

Click the "Network and Internet Connections" option.

Then click the "Network Connections" option.

Double click the icon labelled "Local Area Connection". The icon may have a number after it, for example "Local Area Connection 5".

Click the "Properties" button.

On the "Advanced" tab, select "Protect my computer and network by limiting or preventing access to this computer from the Internet"

Click Settings, then on the ICMP tab select "Allow Incoming echo request".

Click OK, OK, Close.

Your computer is now set up to use the firewall.

Service Pack 2

Load the Control Panel as above, then select the Security Center icon. The Firewall status should be On.

Click on "Windows Firewall" under "Manage security settings for:"

In the ICMP section, click Settings.

Select "Allow Incoming echo request".

Click OK, OK, and close the window.

Your computer is now set up correctly.

Linux Firewalls Using iptables

CentOS / RedhatIptables Firewall Configuration Tutorial

How do I configure a host-based firewall called Netfilter (iptables) under CentOS / RHEL / Fedora / Redhat Enterprise Linux?

Netfilter is a host-based firewall for Linux operating systems. It is included as part of the Linux distribution and it is activated by default. This firewall is controlled by the program called iptables. Netfilter filtering take place at the kernel level, before a program can even process the data from the network packet.

IptablesConfig File

The default config files for RHEL / CentOS / Fedora Linux are:

- /etc/sysconfig/iptables - The system scripts that activate the firewall by reading this file.

Task: Display Default Rules

Type the following command:

```
iptables --line-numbers -n -L
```

Sample outputs:

```
Chain INPUT (policy ACCEPT)
num  targetprot opt source               destination
1    RH-Firewall-1-INPUT  all  --  0.0.0.0/0            0.0.0.0/0
Chain FORWARD (policy ACCEPT)
num  targetprot opt source               destination
1    RH-Firewall-1-INPUT  all  --  0.0.0.0/0            0.0.0.0/0
Chain OUTPUT (policy ACCEPT)
num  targetprot opt source               destination
Chain RH-Firewall-1-INPUT (2 references)
num  targetprot opt source               destination
1    ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0
2    ACCEPT     icmp--  0.0.0.0/0            0.0.0.0/0           icmp type
255
3    ACCEPT     udp  --  0.0.0.0/0            224.0.0.251         udp
dpt:5353
4    ACCEPT     udp  --  0.0.0.0/0            0.0.0.0/0           udp
dpt:53
5    ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0           state
RELATED,ESTABLISHED
6    ACCEPT     tcp  --  0.0.0.0/0            0.0.0.0/0           state NEW
tcp dpt:22
7    ACCEPT     tcp  --  0.0.0.0/0            0.0.0.0/0           state NEW
tcp dpt:53
8    REJECT     all  --  0.0.0.0/0            0.0.0.0/0           reject-
with icmp-host-prohibited
```

Task: Turn On Firewall

Type the following two commands to turn on firewall:

```
chkconfigiptables on
serviceiptables start
# restart the firewall
serviceiptables restart
# stop the firewall
serviceiptables stop
```

Understanding Firewall

There are total 4 chains:

1. **INPUT** - The default chain is used for packets addressed to the system. Use this to open or close incoming ports (such as 80,25, and 110 etc) and ip addresses / subnet (such as 202.54.1.20/29).
2. **OUTPUT** - The default chain is used when packets are generating from the system. Use this open or close outgoing ports and ip addresses / subnets.
3. **FORWARD** - The default chains is used when packets send through another interface. Usually used when you setup Linux as router. For example, eth0 connected to ADSL/Cable modem and eth1 is connected to local LAN. Use FORWARD chain to send and receive traffic from LAN to the Internet.
4. **RH-Firewall-1-INPUT** - This is a user-defined custom chain. It is used by the INPUT, OUTPUT and FORWARD chains.

Packet Matching Rules

1. Each packet starts at the first rule in the chain .
2. A packet proceeds until it matches a rule.
3. If a match found, then control will jump to the specified target (such as REJECT, ACCEPT, DROP).

Target Meanings

1. The target **ACCEPT**means allow packet.
2. The target **REJECT**means to drop the packet and send an error message to remote host.
3. The target **DROP**means drop the packet and do not send an error message to remote host or sending host.

/etc/sysconfig/iptables

Edit /etc/sysconfig/iptables, enter:

```
# vi /etc/sysconfig/iptables
```

You will see default rules as follows:

```
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
:RH-Firewall-1-INPUT - [0:0]
-A INPUT -j RH-Firewall-1-INPUT
-A FORWARD -j RH-Firewall-1-INPUT
-A RH-Firewall-1-INPUT -i lo -j ACCEPT
-A RH-Firewall-1-INPUT -p icmp --icmp-type any -j ACCEPT
-A RH-Firewall-1-INPUT -p udp --dport 5353 -d 224.0.0.251 -j ACCEPT
-A RH-Firewall-1-INPUT -p udp -m udp --dport 53 -j ACCEPT
-A RH-Firewall-1-INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A RH-Firewall-1-INPUT -m state --state NEW -m tcp -p tcp --dport 22 -j
ACCEPT
-A RH-Firewall-1-INPUT -m state --state NEW -m tcp -p tcp --dport 53 -j
ACCEPT
-A RH-Firewall-1-INPUT -j REJECT --reject-with icmp-host-prohibited
COMMIT
```

Drop All Traffic

Find lines:

```
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
```

Update as follows to change the default policy to DROP from ACCEPT for the INPUT and FORWARD built-in chains:

```
:INPUT DROP [0:0]
:FORWARD DROP [0:0]
```

Log and Drop Spoofing Source Addresses

Append the following lines before final COMMIT line:

```
-A INPUT -i eth0 -s 10.0.0.0/8 -j LOG --log-prefix "IP DROP SPOOF "
-A INPUT -i eth0 -s 172.16.0.0/12 -j LOG --log-prefix "IP DROP SPOOF "
-A INPUT -i eth0 -s 192.168.0.0/16 -j LOG --log-prefix "IP DROP SPOOF "
-A INPUT -i eth0 -s 224.0.0.0/4 -j LOG --log-prefix "IP DROP MULTICAST "
-A INPUT -i eth0 -s 240.0.0.0/5 -j LOG --log-prefix "IP DROP SPOOF "
-A INPUT -i eth0 -d 127.0.0.0/8 -j LOG --log-prefix "IP DROP LOOPBACK "
-A INPUT -i eth0 -s 169.254.0.0/16  -j LOG --log-prefix "IP DROP MULTICAST
"
-A INPUT -i eth0 -s 0.0.0.0/8  -j LOG --log-prefix "IP DROP "
-A INPUT -i eth0 -s  240.0.0.0/4  -j LOG --log-prefix "IP DROP "
-A INPUT -i eth0 -s  255.255.255.255/32  -j LOG --log-prefix "IP DROP  "
-A INPUT -i eth0 -s 168.254.0.0/16  -j LOG --log-prefix "IP DROP "
-A INPUT -i eth0 -s 248.0.0.0/5  -j LOG --log-prefix "IP DROP "
```

Log And Drop All Traffic

Find the lines:

```
-A RH-Firewall-1-INPUT -j REJECT --reject-with icmp-host-prohibited
COMMIT
```

Update it as follows:

```
-A RH-Firewall-1-INPUT -j LOG
-A RH-Firewall-1-INPUT -j DROP
COMMIT
```

Open Port

To open port 80 (Http server) add the following before COMMIT line:

```
-A RH-Firewall-1-INPUT -m tcp -p tcp --dport 80 -j ACCEPT
```

To open port 53 (DNS Server) add the following before COMMIT line:

```
-A RH-Firewall-1-INPUT -m tcp -p tcp --dport 53 -j ACCEPT
-A RH-Firewall-1-INPUT -m udp -p tcp --dport 53 -j ACCEPT
```

To open port 443 (Https server) add the following before COMMIT line:

```
-A RH-Firewall-1-INPUT -m tcp -p tcp --dport 443 -j ACCEPT
```

To open port 25 (smtp server) add the following before COMMIT line:

```
-A RH-Firewall-1-INPUT -m tcp -p tcp --dport 25 -j ACCEPT
```

Only allow SSH traffic From 192.168.1.0/24

```
-A RH-Firewall-1-INPUT -s 192.168.1.0/24 -m state --state NEW -p tcp --
dport22 -j ACCEPT
```

Enable Printing Access For 192.168.1.0/24

```
-A RH-Firewall-1-INPUT -s 192.168.1.0/24 -p udp -m udp --dport631 -j ACCEPT
-A RH-Firewall-1-INPUT -s 192.168.1.0/24 -p tcp -m tcp --dport631 -j ACCEPT
```

Allow Legitimate NTP Clients to Access the Server

```
-A RH-Firewall-1-INPUT -s 192.168.1.0/24 -m state --state NEW -p udp --
dport123 -j ACCEPT
```

Open FTP Port 21 (FTP)

```
-A RH-Firewall-1-INPUT -m state --state NEW -p tcp --dport21 -j ACCEPT
```

Save and close the file. Edit /etc/sysconfig/iptables-config, enter:

```
# vi /etc/sysconfig/iptables-config
```

Make sure ftp module is loaded with the space-separated list of modules:

```
IPTABLES_MODULES="ip_conntrack_ftp"
```

To restart firewall, type the following commands:

```
# service iptables restart
# iptables -vnL --line-numbers
```

Edit /etc/sysctl.conf For DoS and Syn Protection

Edit /etc/sysctl.conf to defend against certain types of attacks and append / update as follows:

```
net.ipv4.conf.all.log_martians = 1
net.ipv4.conf.default.accept_source_route = 0
net.ipv4.conf.default.accept_redirects = 0
net.ipv4.conf.default.secure_redirects = 0
net.ipv4.icmp_echo_ignore_broadcasts = 1
#net.ipv4.icmp_ignore_bogus_error_messages = 1
net.ipv4.tcp_syncookies = 1
net.ipv4.conf.all.rp_filter = 1
net.ipv4.conf.default.rp_filter = 1
```

Troubleshooting iptables

A number of tools are at your disposal for troubleshooting iptables firewall scripts. One of the best methods is to log all dropped packets to the /var/log/messages file.

Checking The Firewall Logs

You track packets passing through the iptables list of rules using the LOG target. You should be aware that the LOG target:

- Logs all traffic that matches the iptables rule in which it is located.
- Automatically writes an entry to the /var/log/messages file and then executes the next rule.

If you want to log only unwanted traffic, therefore, you have to add a matching rule with a DROP target immediately after the LOG rule. If you don't, you'll find yourself logging both desired and unwanted traffic with no way of discerning between the two, because by default iptables doesn't state why the packet was logged in its log message.

This example logs a summary of failed packets to the file /var/log/messages. You can use the contents of this file to determine which TCP/UDP ports you need to open to provide access to specific traffic that is currently stopped.

```
#---------------------------------------------------------------
# Log and drop all other packets to file /var/log/messages
# Without this we could be crawling around in the dark
#---------------------------------------------------------------

iptables -A OUTPUT -j LOG
iptables -A INPUT -j LOG
iptables -A FORWARD -j LOG

iptables -A OUTPUT -j DROP
iptables -A INPUT -j DROP
iptables -A FORWARD -j DROP
```

Here are some examples of the output of this file:

- Firewall denies replies to DNS queries (UDP port 53) destined to server 192.168.1.102 on the home network.

```
Feb 23 20:33:50 bigboy kernel: IN=wlan0 OUT=
MAC=00:06:25:09:69:80:00:a0:c5:e1:3e:88:08:00 SRC=192.42.93.30
DST=192.168.1.102 LEN=220 TOS=0x00 PREC=0x00 TTL=54 ID=30485 PROTO=UDP
SPT=53 DPT=32820 LEN=200
```

- Firewall denies Windows NetBIOS traffic (UDP port 138)

```
Feb 23 20:43:08 bigboy kernel: IN=wlan0 OUT=
MAC=ff:ff:ff:ff:ff:ff:00:06:25:09:6a:b5:08:00 SRC=192.168.1.100
DST=192.168.1.255 LEN=241 TOS=0x00 PREC=0x00 TTL=64 ID=0 DF PROTO=UDP
SPT=138 DPT=138 LEN=221
```

- Firewall denies Network Time Protocol (NTP UDP port 123)

```
Feb 23 20:58:48 bigboy kernel: IN= OUT=wlan0 SRC=192.168.1.102
DST=207.200.81.113 LEN=76 TOS=0x10 PREC=0x00 TTL=64 ID=0 DF PROTO=UDP
SPT=123 DPT=123 LEN=56
```

The traffic in all these examples isn't destined for the firewall; Therefore, you should check your INPUT, OUTPUT, FORWARD, and NAT related statements. If the firewall's IP address is involved, then you should focus on the INPUT and OUTPUT statements

If nothing shows up in the logs, then follow the steps in "Simple Network Troubleshooting", to determine whether the data is reaching your firewall at all and, if it is not, the location on your network that could be causing the problem.

As a general rule, you won't be able to access the public NAT IP addresses from servers on your home network. Basic NAT testing requires you to ask a friend to try to connect to your home network from the Internet.

You can then use the logging output in /var/log/messages to make sure that the translations are occurring correctly and iptables isn't dropping the packets after translation occurs.

iptables Won't Start

The iptables startup script expects to find the /etc/sysconfig/iptables before it starts. If none exists, then symptoms include the firewall status always being "inactive". An easy way fix this is to simply create the file using the touch command and then using chmod to give the file the correct permissions. Starting iptables should be successful after this.

```
[root@bigboytmp]# touch /etc/sysconfig/iptables
[root@bigboytmp]# chmod 600 /etc/sysconfig/iptables
[root@bigboytmp]# systemctl start iptables.service
[root@bigboytmp]# systemctl status iptables.service
iptables.service - IPv4 firewall with iptables
  Loaded: loaded (/usr/lib/systemd/system/iptables.service; enabled)
  Active: active (exited) since Thu, 09 Aug 2012 22:25:50 -0700; 4s ago
rocess: 19177 ExecStop=/usr/libexec/iptables.init stop (code=exited,
status=0/SUCCESS)
rocess: 19231 ExecStart=/usr/libexec/iptables.init start (code=exited,
status=0/SUCCESS)
CGroup: name=systemd:/system/iptables.service

Aug 09 22:25:50 web-003 iptables.init[19231]: iptables: Applying firewall
rules: [  OK  ]
[root@bigboytmp]#
```

Simple Network Troubleshooting

Introduction

You will eventually find yourself trying to fix a network related problem which usually appears in one of two forms. The first is slow response times from the remote server, and the second is a complete lack of connectivity. These symptoms can be caused by:

Sources of Network Slowness

- NIC duplex and speed incompatibilities
- Network congestion

- Poor routing
- Bad cabling
- Electrical interference
- An overloaded server at the remote end of the connection
- Misconfigured DNS

Sources of a Lack of Connectivity

All sources of slowness can become so severe that connectivity is lost. Additional sources of disconnections are:

- Power failures
- The remote server or an application on the remote server being shut down.

We discuss how to isolate these problems and more in the following sections.

Doing Basic Cable and Link Tests

Your server won't be able to communicate with any other device on your network unless the NIC's "link" light is on. This indicates that the connection between your server and the switch/router is functioning correctly.

Other sources of link failure include:

- The cables are bad.
- The switch or router to which the server is connected is powered down.
- The cables aren't plugged in properly.

Testing Your NIC

It is always a good practice in troubleshooting to be versed in monitoring the status of your NIC card from the command line. The following sections introduce a few commands that will be useful.

Some Networking Troubleshooting Commands

pingdestination_address

ifcfg

traceroute

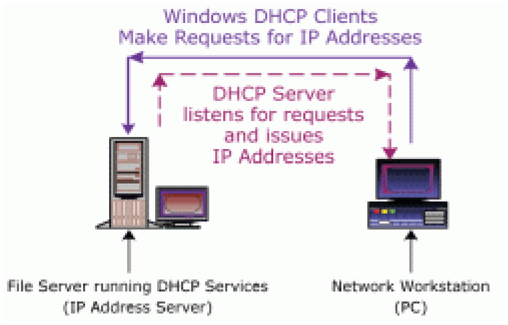
tcpdump

nmap

nslookupdestination_address

digdestination_address

DHCP



DHCP stands for Dynamic Host Configuration Protocol. It handles the automatic assignment of IP addresses and other configuration settings for devices on your network.

Isn't there a way to configure IP addresses manually? Yes, there is; if you've explored the Network and Sharing Center in your control panel, you've probably seen this. But DHCP automates it. This is especially good for people who have laptops, aren't hooked up to the Internet all the time and often move from place to place with their portable devices. They can simply get a new IP address as needed without having to do it manually.

DHCP Basics

DHCP is designed to make the assignment of IP addresses and other network configuration information faster and easier. Rather than going around to every device on your network and setting up its network configuration manually, you can use your DHCP server to set up pools of addresses, called **scopes,** from which clients can request a temporary IP address.

DHCP is a protocol that uses Level 4 on the OSI model. It communicates using **User Datagram Protocol (UDP)** datagrams through UDP Port 68. DHCP works with most current and past Windows clients, and also Linux, Macintosh, and many network-capable printers.

Benefits of running DHCP

- *Flexible configuration*. DHCP makes it easy to implement changes in IP address configuration. Rather than manually configure each device's network connection every time a new DNS server is added, you can go into the DHCP server and make the necessary changes.
- *Scalable design.* Any size network can benefit from DHCP.
- *Centralized administration.* You can make needed configuration changes in a single place. This saves time and effort over going around to every device on your network.
- *Automatic host configuration.* DHCP automates the assignment of IP addresses.

DHCP components

These components work together to automate IP addressing.

- DHCP leases
- DHCP scopes
- DHCP reservations
- DHCP options
- DHCP relay agents

DHCP Leases

DHCP leases define the **lease duration**, or the amount of time that a client can keep an IP address before releasing it. In Windows Server 2008, the default lease duration is 8 days for wired clients and 6 days for wireless clients. On the DHCP server, leases act as placeholders in its DHCP database. When the lease is issued, the IP address is removed from the list of available addresses to prevent the issuing of the same address to more than one computer.
To determine the duration of a lease, one must consider the client type, the connection time, and the available range of IP addresses.
Client type has to do with the type of devices on your network, such as desktop computers, mobile notebooks, and servers. If you have more than the usual number of laptops on your network, the length of lease duration should decrease.
You should estimate the *connection time*, or the average length of time your clients spend on the network. If this time is relatively low, such as 2 to 4 hours, your lease duration doesn't have to be as long.
An important factor is the *number of IP addresses* you have available, as well as the number of clients who need DHCP's range of addresses. If you have a lot of clients in relation to your number of available addresses, a shorter lease duration is advisable to avoid running out of IP addresses.
For most networks, the default settings should be sufficient. Shorter lease times would be necessary for networks with many clients that connect for less than a day. A college campus is a good example, since campuses usually offer free wireless access for their students, who are typically connected for a few hours a day.

DHCP Scope

A **DHCP Scope** is a range of IP addresses and related configuration information available by request from a DHCP client. These scopes usually represent a single subnet, or segment of a network. Each scope is a continuous range of IP addresses defined by a beginning IP address and an ending IP address. If you need to exclude IP addresses, you must create **exclusions** for those addresses. One reason for creating these addresses might be hardware with static IP addresses, like printers.

DHCP Reservations

When would you reserve an IP address? Well, in some cases, a network device needs to have a static IP address. An example would be a server, a router, or a network printer. In the DHCP role console, you reserve these addresses using the list below.

Common DHCP Options

Use these option codes to reserve IP addresses in DHCP.

- 3: Router
- 6: DNS server
- 15: DNS domain name
- 42: NTP server
- 44: WINS server (NetBIOS name server)
- 45: NetBIOS datagram disribution server (NBDD)
- 46: WINS/NetBIOS node type
- 47: NetBIOS scope ID
- 51: Lease Time
- 53: DHCP message type
- 55: Special option type used to communicate a parameter request list to the DHCP server
- 58: Renewal time value (T1)
- 59: Rebind time value (T2)

DHCP Options

DHCP options are basic settings that a client needs for proper network communication. These options include an IP address, a subnet mask, a default gatewar, primary and secondary DNS servers, primary and secondary Windows Internet Name Service (WINS) if applicable, and DHCP lease expiration. You can define these options when creating the scope or change them later.

Server options are settings defined on each server that apply to all scopes on a specific DHCP server. **Scope options** are settings defined on each scope that apply only to the scope to which they are added. Router options are typically defined using scope options, which override server options. Server options are usually used for network resources whose IP addresses are the same for all scopes, such as DNS and WINS.

DHCP Relay Agent

DHCP requests are broadcast messages that cannot be routed, so they are limited to the subnet of the client requesting an IP address. You can choose one of two options to get around this. You can have a DHCP server on each subnet, which may be expensive and, therefore, implausible. Or, you can use a **DHCP relay agent** to forward DHCP requests. This agent accepts the broadcast packets and converts them to unicast packets that can traverse a routed network and vice versa when the DHCP server replies to the client requesting an IP address. Most modern routers support the passing of DHCP requests.

12.2. Configuring a DHCP Server

The `dhcp` package contains an ISC DHCP server. First, install the package as the superuser:

```
~]#yum install dhcp
```

Installing the `dhcp` package creates a file, `/etc/dhcp/dhcpd.conf`, which is merely an empty configuration file:

```
~]#cat /etc/dhcp/dhcpd.conf
#
# DHCP Server Configuration file.
#   see /usr/share/doc/dhcp*/dhcpd.conf.sample
```

The sample configuration file can be found at `/usr/share/doc/dhcp-<version>/dhcpd.conf.sample`. You should use this file to help you configure `/etc/dhcp/dhcpd.conf`, which is explained in detail below.

DHCP also uses the file `/var/lib/dhcpd/dhcpd.leases` to store the client lease database.

12.2.1. Configuration File

The first step in configuring a DHCP server is to create the configuration file that stores the network information for the clients. Use this file to declare options and global options for client systems.

The configuration file can contain extra tabs or blank lines for easier formatting. Keywords are case-insensitive and lines beginning with a hash sign (#) are considered comments.

There are two types of statements in the configuration file:

- Parameters — State how to perform a task, whether to perform a task, or what network configuration options to send to the client.
- Declarations — Describe the topology of the network, describe the clients, provide addresses for the clients, or apply a group of parameters to a group of declarations.

The parameters that start with the keyword option are referred to as *options*. These options control DHCP options; whereas, parameters configure values that are not optional or control how the DHCP server behaves.

Parameters (including options) declared before a section enclosed in curly brackets ({ }) are considered global parameters. Global parameters apply to all the sections below it.

Restart the DHCP daemon for the changes to take effect

If the configuration file is changed, the changes do not take effect until the DHCP daemon is restarted with the command `service dhcpd restart`.

DHCP Servers

Dynamic Host Configuration Protocol (DHCP) is a network protocol that automatically assigns TCP/IP information to client machines. Each DHCP client connects to the centrally located DHCP server, which returns that client's network configuration (including the IP address, gateway, and DNS servers).

Why Use DHCP?

DHCP is useful for automatic configuration of client network interfaces. When configuring the client system, the administrator chooses DHCP instead of specifying an IP address, netmask, gateway, or DNS servers. The client retrieves this information from the DHCP server. DHCP is also useful if an administrator wants to change the IP addresses of a large number of systems. Instead of reconfiguring all the systems, he can just edit one DHCP configuration file on the server for the new set of IP addresses. If the DNS servers for an organization changes, the changes are made on the DHCP server, not on the DHCP clients. When the administrator restarts the network or reboots the clients, the changes will go into effect.

If an organization has a functional DHCP server properly connected to a network, laptops and other mobile computer users can move these devices from office to office.

DHCP Troubleshooting

The most common problems with DHCP usually aren't related to the server; after the server is configured correctly there is no need to change any settings and it therefore runs reliably. The problems usually occur at the DHCP client's end for a variety of reasons. The following sections present simple troubleshooting steps that you can go through to ensure that DHCP is working correctly on your network.

DHCP Clients Obtaining 169.254.0.0 Addresses

Whenever Microsoft DHCP clients are unable to contact their DHCP server they default to selecting their own IP address from the 169.254.0.0 network until the DHCP server becomes available again. This is frequently referred to as Automatic Private IP Addressing (APIPA). Here are some steps you can go through to resolve the problem:

- Ensure that your DHCP server is configured correctly and use the pgrep command discussed earlier to make sure the DHCP process is running. Pay special attention to your 255.255.255.255 route, especially if your DHCP server has multiple interfaces.
- Give your DHCP client a static IP address from the same range that the DHCP server is supposed to provide. See whether you can ping the DHCP server. If you cannot, double-check your cabling and your NIC cards.

- DHCP uses the BOOTP protocol for its communication between the client and server. Make sure there are no firewalls blocking this traffic. DHCP servers expect requests on UDP port 67 and the DHCP clients expect responses on UDP port 68. Use `tcpdump` on the server's NIC to verify the correct traffic flows.

Other DHCP Failures

If the DHCP server fails to start then use your regular troubleshooting techniques outlined in Chapter 4, "Simple Network Troubleshooting", to help rectify your problems. Most problems with an initial setup are often due to:

- Incorrect settings in the /etc/dhcpd.conf file such as not defining the networks for which the DHCP server is responsible;
- Firewall rules that block the DHCP bootp protocol on UDP ports 67 and 68;
- Routers failing to forward the bootp packets to the DHCP server when the clients reside on a separate network.

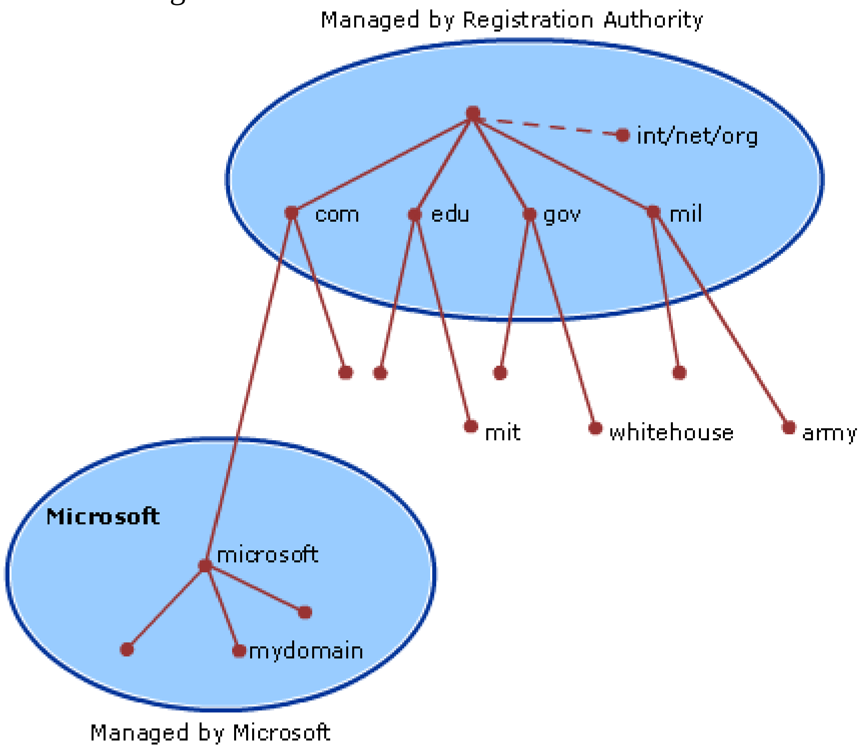
Always check your /var/logs/messages file for dhcpd errors and remember that mandatory keywords in your configuration file may change when you upgrade your operating system. Always read the release notes to be sure.

1 Name Server and Configuration **6Hrs**

1.1 DNS principles and Operations
1.2 Basic Name Server and Client Configuration
1.3 Caching Only name server -*DNSCACHE*
1.4 Primary and Slave Name Server
1.5 DNS Zone Transfers
1.6 DNS Dynamic Updates
1.7 DNS Delegation
1.8 Troubleshooting

What Is DNS?

Domain Name System (DNS) provides **name resolution** for all Internet communications and many private networks, including all Windows Active Directory domains. This means that, whenever you type a website's address into your Web browser, DNS will go out and fetch the IP address associated with that address. This beats having to remember the IP address 69.147.114.224 every time you want to go to Yahoo.com. DNS also assists the delivery of e-mail by providing mail exchanger records that tell SMTP servers where to send e-mail messages.



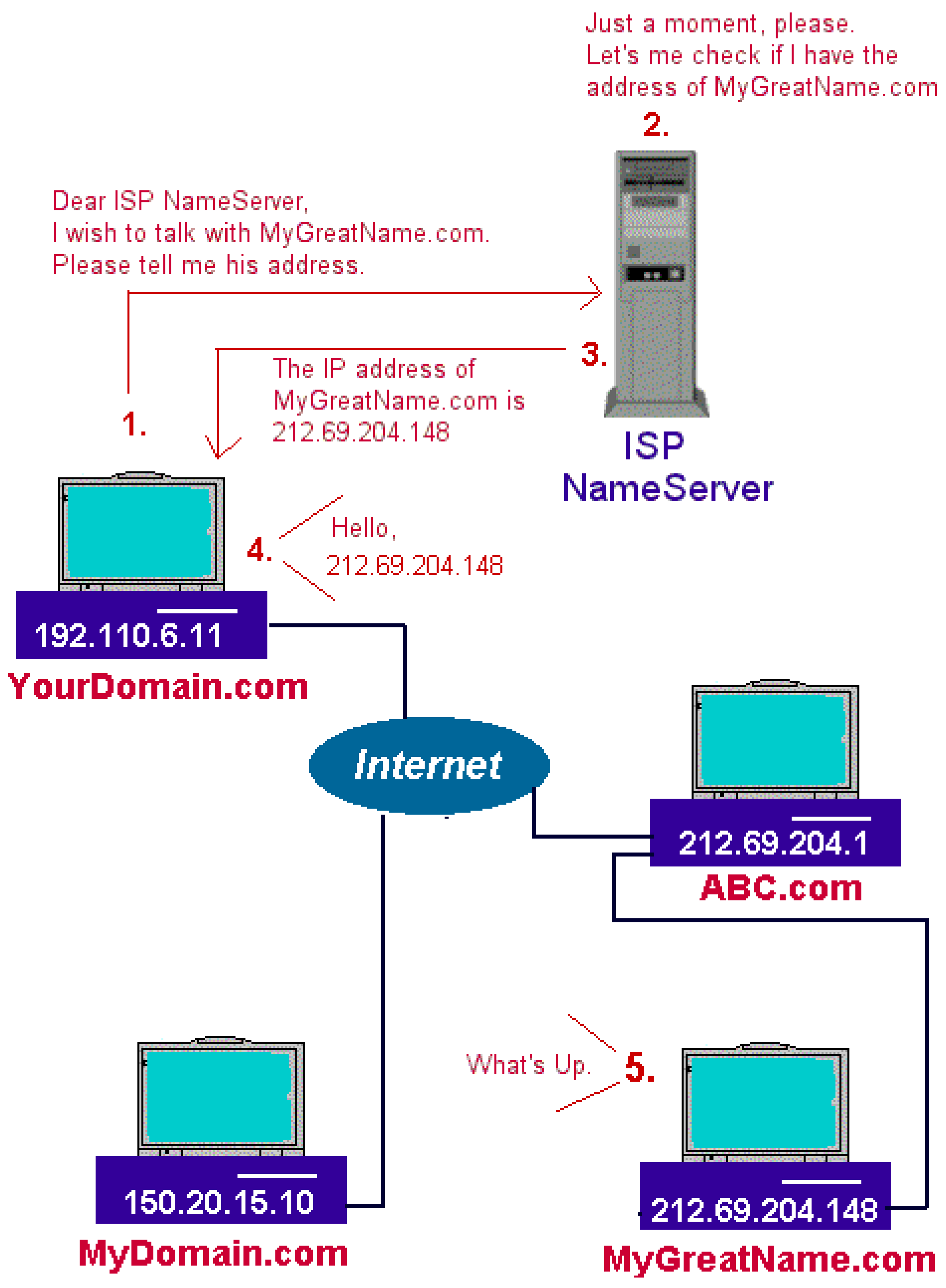
On the Internet, there are thousands of **DNS servers** that maintain DNS records for name resolution.

Basic Concept of Domain Name System (DNS)

When the computers (hosts) on the Internet is getting more and more, and thus the growing list of HOSTS.txt. The solutions used in the early stage of Internet was not more suitable.

In view of this, Paul Mockapertris designed a system to manage the domain names on the Internet. The system is called Domain Name System, or DNS in short, in 1984

The working principle of DNS is not difficult to understand, as shown below:



Working Procedures of DNS:

1. The information (mainly domain names and their corresponding IP Addresses) of hosts (computers) on the Internet is saved in the Domain Name Servers. The Domain Name Servers are distributed widely on the Internet.

2. When your computer need to connect with a host on the Internet (e.g. MyGreatName.com), you only need to enter the Domain Name (e.g.

MyGreatName.com) in the URL of browser. Your computer will then contact the configured or default Name Servers (usually your ISP Name Server), asking for the IP Address of the host (e.g. MyGreatName.com).

3. The Name Server will then tell your computer the IP Address of the query host.

4. Once your computer get the IP Address of the host, your computer can then communicate with the host.

   From the above working procedures of DNS, you should notice that there are a lot of disadvantages. For example:

   - Each Name Server has to save the information of ALL hosts on the Internet.

   - If a Name Server forgets to update the information, many new domain names will not be found!

   - To guarantee that all new domains are activated on the Internet, the information of all Name Servers must be updated. This may take 2 - 3 months!

   - How to guarantee that all Name Servers are updated on schedule?

The above procedures only show the basic concept of DNS.

DNS Operation

DNS Servers

On the client side, a DNS resolver is used to send queries to DNS servers. The resolver is normally part of a library routine or it is built into the application. DNS uses zone files to keep name and IP address database information for the internet domain or hierarchical set of domains. Zones are storage of information in a file for a DNS domain or DNS subdomains (DNS domains are not the same as Windows domains). DNS does not yet support dynamic configuration but has been modified for Windows systems to do so. Different aliases may be created by the administrator for the same host. Three types of name servers as defined by how it relates to the zone information:

- **Primary** - Locally stored files exist on the name server data base. The master zone file copy is stored here.
- **Secondary** - Gets data called a zone transfer from another server that is the zone authority.
- **Caching Only** - Caches name server information and does not contain its own files.

A primary and secondary name server should be used on a network. **When a zone is defined, some server must be configured to be a master name server for the zone**. There can be different master name servers for different zones. The master server provides copies of the zone information to the secondary DNS server. Name servers can be configured to get information from other name servers when the information is not found in the local database. These types are forwarders and slaves. Name servers as categorized by function:

- **Master** - The zone authority that contains the master zone files.
- **Forwarders** - A name server that passes name resolution requests to other name servers. This configuration is done on a per server basis.
- **Slaves** - Slave name servers are configured to use forwarders.

Windows introduces additional terminology:

- **Standard primary** - The same as a primary DNS server listed above. This is a master server by function.
- **Active Directory Integrated** (primary) - DNS entries are stored with Active Directory data rather than a normal zone file. More than one of these Active Directory primary servers may exist due to Active directory replication. This term is used to refer to both the Active Directory Integrated zones and files that support the zone.
- **Standard secondary** - The same as a secondary DNS server listed above. This is a slave server by function.
- **Root server** - The server that has the DNS data for the root zone. The root zone is the organization internal network root zone or internet root zone. It is used when a private network is not directly on the internet (no connection or via proxy server).

If the DNS server is connected to the internet, the DNS Server Wizard will not allow the DNS server to be configured as a root server.

Queries

Query types are:

- **Inverse** - Getting the name from the IP address. These are used by servers as a security check.
- **Iterative** - Server gives its best answer. This type of inquiry is sent from one server to another.
- **Recursive** - Cannot refer the query to another name server.

Zone Transfers

The DNS zone file serial number is used to track DNS changes. The notify function is used to initiate zone transfers. Zone transfer types are:

- **Full** - AXFR Query - Secondary server refresh interval expires and it sends an AXFR query.
- **Incremental** - IXFR query - Only new or updated entries are copied.

DNS Zones

Possible zones include:

- **Forward lookup zone** - Name to IP address map.
- **Reverse lookup zone** - IP address to name map.
- **Standard primary zone (primary zone)** - A master copy of a forward or reverse lookup zone.

- **Active Directory integrated zone** - A copy of a standard primary or Active Directory integrated zone. The IP address and computer name is stored in Active Directory and replicated to all local domain controllers. DNS information is not replicated to domain controllers outside the domain.
- **Standard secondary zone (secondary zone)**

Name Server

A name server translates domain names into IP addresses. This makes it possible for a user to access a website by typing in the domain name instead of the website's actual IP address. For example, when you type in "www.microsoft.com," the request gets sent to Microsoft's name server which returns the IP address of the Microsoft website.

Each domain name must have at least two name servers listed when the domain is registered. These name servers are commonly named ns1.servername.com and ns2.servername.com, where "servername" is the name of the server. The first server listed is the primary server, while the second is used as a backup server if the first server is not responding.

Name servers are a fundamental part of the Domain Name System (DNS). They allow websites to use domain names instead of IP addresses, which would be much harder to remember. In order to find out what a certain domain name's name servers are, you can use a WHOIS lookup tool.

What is a name server?

A name server is similar to a telephone switchboard. It holds the information that tells the Internet where to find your web site and where to deliver your email. Name servers look something like this: *yns1.yahoo.com*.

Yahoo!'s name servers must be listed as the primary and secondary name servers in order for Yahoo! to host your domain services properly. If you replace Yahoo!'s default name servers, Yahoo! will no longer be able to provide any services associated with your domain.

To add new name servers or replace Yahoo!'s name servers with your own, head to your Domain Control Panel, click the "Advanced DNS settings" link, then choose the "Change Name Servers" button. On the following page, you'll be able to modify your name servers.

Tip: Name server host names you submit for use with your service must be registered. If your name server host names are not registered, you may encounter errors when trying to modify your name servers.

Please note: Any changes that you make to your advanced DNS settings can interrupt your service. If you are not an advanced user, we strongly recommend that you not change these settings.

DNS Name Server Type:

Primary

The primary nameserver is the authoritative source for all information about a specific domain. It loads the domain information from a locally maintained disk file that is built by the domain administrator. This file (the zone file) contains the most accurate information about a piece of the domain hierarchy over which this server has authority. The primary server is a master server, because it can answer any query about its domain with full authority.

The terms *master server* and *authoritative server* are used interchangeably.

Configuring a primary server requires creating a complete set of configuration files: zone files for the regular domain and the reverse domain, the boot file, the cache file, and the loopback file. No other configuration requires creating this complete set of files.

Secondary(replicating)
A secondary server transfers a complete set of domain information from the primary server. The zone file is transferred from the primary server and stored on the secondary server as a local disk file. This transfer is called a *zone file transfer*. A secondary server keeps a complete copy of all domain information, and can answer queries about that domain with authority. Therefore, a secondary server is also considered a master server.

Configuring a secondary server does not require creating local zone files, because the zone files are downloaded from the primary server. However, the other files (a boot file, a cache file, and a loopback file) are required.

Caching-only
A caching-only server runs the nameserver software, but keeps no nameserver database files. It learns the answer to every nameserver query from some remote server. Once it learns an answer, the server caches the answer and uses it to answer future queries for the same information. All nameservers use cached information in this manner, but a caching-only server depends on this technique for all of its nameserver information. It is not considered an authoritative (or master) server, because all of the information it provides is secondhand. Only a boot file and a cache file are required for a caching-only configuration. But the most common configuration also includes a loopback file. This is probably the most common nameserver configuration, and apart from the resolver-only configuration, it is the easiest to configure.

DNS Cache

DNS is a caching protocol. When a client queries its local DNS server, and the local DNS server is not authoritative for the query, then this server will go looking for an authoritative name server in the DNS tree. The local name server will first query a root server, then a TLD server and then a domain server. When the local name server resolves the query, then it will relay this information to the client that submitted the query, and it will also keep a copy of these queries in its cache. So when a(nother) client submits the same query to this name server, then it will retrieve this information form its cache.
For example, a client queries for the A record on www.linux-training.be to its local server. This is the first query ever received by this local server. The local server checks that it is not authoritative for the linux-training.be domain, nor for the .be TLD, and it is also not a root

server. So the local server will use the root hints to send an iterative query to a root server. The root server will reply with a reference to the server that is authoritative for the .be domain (root DNS servers do not resolve fqdn's, and root servers do not respond to recursive queries). The local server will then sent an iterative query to the authoritative server for the .be TLD. This server will respond with a reference to the name server that is authoritative for the linux-training.be domain. The local server will then sent the query for www.linux-training.be to the authoritative server (or one of its slave servers) for the linux-training.be domain. When the local server receives the ip-address for www.linux-training.be, then it will provide this information to the client that submitted this query. Besides caching the A record for www.linux-training.be, the local server will also cache the NS and A record for the linux-training.be name server and the .be name server.

Caching nameserver

Fortunately, setting up a caching nameserver is easy using RHEL (Red Hat Enterprise Linux) or Fedora RPMs. The following RPMs need to be installed on the machine acting as the nameserver (use `rpm -q` to determine if these packages are installed):

- `bind` (includes DNS server, named)
- `bind-utils` (utilities for querying DNS servers about host information)
- `bind-libs` (libraries used by the bind server and utils package)
- `bind-chroot` (tree of files which can be used as a chroot jail for bind)
- `caching-nameserver` (config files for a simple caching nameserver)

A caching nameserver forwards queries to an upstream nameserver and caches the results. Open the file `/var/named/chroot/etc/named.conf` and add the following lines to the global options section:

```
     forwarders { xxx.xxx.xxx.xxx; xxx.xxx.xxx.xxx; }; #IP of upstream ISP
nameserver(s)
     forward only; #rely completely on our upstream nameservers
```

The block above will cause the caching name server to forward DNS requests it can't resolve to your ISP nameserver. Save the `named.conf` file and then assign 644 permissions:

```
chmod 644 named.conf
```

Check the syntax using the `named-checkconf` utility provided by the bind RPM:

```
named-checkconf named.conf
```

Correct any syntax errors (watch those semicolons) `named-checkconf` reports. Monitoring the `/var/log/messages` file may also be helpful in debugging any errors.
We now need to set the local resolver to point to itself for DNS resolution. Modify the `/etc/resolv.conf` file to the following:

```
nameserver 127.0.0.1
```

If you are running a DHCP server on your router make sure your `/etc/resolv.conf` file does not get overwritten whenever your DHCP lease is renewed. To prevent this from happening, modify `/etc/sysconfig/network-scripts/ifcfg-eth0` (replace eth0 with your network interface if different) and make sure the following settings are set:

```
BOOTPROTO=dhcp
PEERDNS=no
```

```
TYPE=Ethernet
```

Go ahead and start the nameserver as root and configure to start in runlevels 2-5:

```
service named start
chkconfig named on
```

Linux: Setup as DNS Client / Name Server IP Address

Many new Linux user finds it difficult to setup / modify new name server address (NS1 / NS2).

Local name resolution is done via /etc/hosts file. If you have small network, use /etc/hosts file. DNS (domain name service is accountable for associating domain names with ip address, for example domain yahoo.com is easy to remember than IP address 202.66.66.12) provides better name resolution. To configure Linux as DNS client you need to edit or modify /etc/resolv.conf file. This file defines which name servers to use. You want to setup Linux to browse net or run network services like www or smtp; then you need to point out to correct ISP DNS servers:

/etc/resolv.conf file

In Linux and Unix like computer operating systems, the /etc/resolv.conf configuration file contains information that allows a computer connected to the Internet to convert alpha-numeric names into the numeric IP addresses that are required for access to external network resources on the Internet. The process of converting domain names to IP addresses is called "resolving."

The resolv.conf file typically contains the IP addresses of nameservers (DNS name resolvers) that attempt to translate names into addresses for any node available on the network.

Setup DNS Name resolution

Steps to configure Linux as DNS client, first login as a root user (use su command):

Step # 1: Open /etc/resolv.conf file:

```
# vi /etc/resolv.conf
```

Step #2: Add your ISP nameserver as follows:

```
search isp.com
nameserver 202.54.1.110
nameserver 202.54.1.112
nameserver 202.54.1.115
```

Note Max. three nameserver can be used/defined at a time.

Step # 3: Test setup nslookup or dig command:

```
$ dig www.nixcraft.com
$ nslookup www.nixcraft.com
```

Setting Up a Nameserver

A nameserver is a server that resolves hostnames to IP addresses. Instead of having to type in "209.81.10.250", I can just type in "www.penguincomputing.com" to get to a site. BIND is nameserver software that runs on many types of machines, originally written by Paul Vixie and now maintained by the Internet Software Consortium (ISC).

This was one of the trickiest things I've had to try to figure out so far. I didn't talk to anybody as to how to go about this, unlike PPP setup. But I finally found the Linux DNS HOWTO and used that. Hopefully this will save you some trouble. Here I have instructions for BIND 8, which is newer and more advanced than BIND 4 (which some distributions still use).

1. Removing Old Nameserver (BIND 4)
2. Installing New Nameserver (BIND 8)
3. Configuration: /etc/named.conf
4. Caching Nameserver
5. Zone Files in /var/named
6. Examples of Zone Files
    - Domains
    - Localhost
    - Reverse Mapping

Removing Old Nameserver (BIND 4)

In the event that your distribution already comes with BIND 8, then all you need to do is find out how the configuration works, and/or put in entries for machines that the Linux box will be the nameserver for. I think Slackware comes with BIND 8. I don't know about Debian. Most new distributions (including Red Hat 5.2) will already have BIND 8 so you'll probably have to look for the configuration file and how it has been pre-configured.

I had to remove the old nameserver first, so it wouldn't get in the way of the new one. It's fine to remove it and replace it with a new one; there aren't any other packages that *really* depend on it.

Using Red Hat 5.1, I typed **`rpm -e bind bind-utils caching-nameserver`**. That removed all the old packages. Be careful about this, especially about bind-utils, because bind-utils contains the tools such as **dnsquery** and **nslookup** that you might be using fairly often. Red Hat 5.2 already has BIND 8.

Installing New Nameserver (BIND 8)

First, download BIND from ftp.isc.org. If you've got it on your system already, then you don't need to get BIND 8 unless you want to make a minor upgrade. The filename is something like bind-8.1.2-src.tar.gz, which says that it's BIND version 8.1.2 in source format (which you have to compile on your system). I'll work with the source version since that's what I usually do anyway.

After you have it on your system, type **`tar -zxvf bind-8.1.2.tar.gz`**. It will extract a directory called `src/` and that's where you go into. Simply type "make" and have it compiled for you. If you need anything to be tweaked then read the INSTALL file (**`less INSTALL`**) and find what you need. After it finishes compiling, type **`make install`**.

Configuration: /etc/named.conf

Configuring the nameserver was probably the hardest part of the process that I had to go through. Hopefully, what I've written up about what I learned will save the reader some trouble.

The main BIND 8 configuration file is in `/etc/named.conf`. The configuration syntax for the file is documented at http://www.isc.org/bind8/config.html. Here's a sample of a configuration file (as /etc/named.conf), with an explanation below.

```
/*
 * A simple BIND 8 configuration
 */

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

zone "penguincomputing.com" in {
        type master;
        file "master/penguincomputing.com";
};

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

zone "." in {
        type hint;
        file "named.cache";
};
```

In "options" I only had one option: where the directory for **zone files** were. Zone files are where information about domains is stored, and each file has information about a zone. It's a section to cover, I guess, so that's why they're called zones. I have `/var/named/` as my named data directory.

The "penguincomputing.com" section is pretty straightforward. It just indicates the location of the penguincomputing.com zone files and tells **named** that this server is a master nameserver for the penguincomputing.com zone.

The "0.0.127.in-addr.arpa" zone is for mapping localhost to 127.0.0.1, basically. It has its own zone file.

The "." zone indicates a caching nameserver; that is, someone can actually use your machine to resolve hostnames (including you). I've heard that is is efficient especially when using PPP connections, but I don't know for sure. Read the "Caching Nameserver" section to read up on how to create one.

Caching Nameserver

First you need to get a "named.cache" file. I'm not sure if you can name it anything else, but let's just use that filename. In /var/named/ (or wherever you put your nameserver's data files), type **`dig @a.root-servers.net > named.cache`**. This will ask for the addresses of the main DNS servers of the Internet and direct them to a file. I'm *guessing* that the purpose of this is to give your machine an idea of which machines on the Internet to ask about hosts.

Periodically, like once a month, update the named.cache file by running that command once in a while. You can use a cron job for that. If you don't know what I'm talking about here, don't worry about it. Just be sure to update it using **dig** once in a while, that's all you have to do.

You have `/etc/named.conf` point to wherever your named.cache file is under the "." zone.

Zone Files in /var/named/

In `/var/named/`, I created directories for every type of zone I had. The directories I have in there are: `master`, `slave/`, and `zone`. With the domain name system, there is a server for each domain that is the main server (the master). I suppose that the slave server is there in case the main (master) server is down. For each domain there should be *at least* 2 servers, one master and one slave. That's just the way it goes.

While interning at Penguin Computing I set up both the master and slave DNS servers. The master's information should go in the `master` directory. You should be able to figure out where the slave's information goes. The information they store is the same, but since one machine is the main one that keeps the information (master) and the other simply follows the master's information (slave), you need to stay organized and make sure you're configuring the right machine for its right place in the nameserver system.

Note that the slave nameserver for one domain can also be the master nameserver for another domain. There just can't be two masters for a single domain, though I think there can be several slaves.

Examples of Zone Files

To figure something like this out, I was looking hard for examples. And examples really help, so hopefully you won't be too confused by my examples. Hey, I try.

Domains

The information for each domain is put in a single file. This file contains valuable information for each domain, such as machines that are under that domain (like, for the penguincomputing.com domain, the nameservers would have the information for what IP address pasta.penguincomputing.com gets and the one that antarctica.penguincomputing.com gets). Here's an example of a domain's records:

```
@       IN      SOA     penguincomputing.com.  root.penguincomputing.com.
(
                        1998082403      ;       serial
```

```
                               4H              ;       refresh, seconds
                               2H              ;       retry, seconds
                               1W              ;       expire, seconds
                               1D      )       ;       minimum, seconds
                        NS      pasta.penguincomputing.com.
                        NS      rice.penguincomputing.com.
                        MX      10 penguincomputing.com. ;  Primary Mail Exchanger

localhost       A       127.0.0.1
router          A       140.174.204.2

penguincomputing.com.  A       209.81.10.250
ns              A       209.81.10.250
www             A       209.81.10.250
ftp             CNAME   penguincomputing.com.
mail            CNAME   penguincomputing.com.
news            CNAME   penguincomputing.com.
pasta           CNAME   penguincomputing.com.
slashdot        CNAME   penguincomputing.com.
rice            CNAME   antarctica.penguincomputing.com.
antarctica      A       209.81.10.252
antarctic       CNAME   antarctica.penguincomputing.com.
www.antarctic   CNAME   antarctica.penguincomputing.com.
www.antarctica  CNAME   antarctica.penguincomputing.com.
zork            A       209.81.10.253
tux             A       209.81.10.146
xfce            A       209.81.10.252

@               TXT     "Penguin Computing"
@               HINFO   Linux 2.0.34
```

There's a pretty weird syntax to be used for these zone files. I never would have figured it out on my own had I not read the Linux DNS HOWTO document. Basically, it specifies information about all the machines in the domain, and it contains information about the domain itself, such as the type of machine the server is running on.

I'll start explaining what all the stuff does. In the first line, it's saying that this file specifies the zones for the penguincomputing.com domain, and to send anything about the domain to *root@penguincomputing.com*. Since the "@" character has special significance in these **zone files**, the username (root) and machine name (penguincomputing.com) have to be separated by a dot. I guess BIND just knows how to split it up. That's how you fill in stuff for your domain as well.

The line with the comment "serial" shows the serial number of that domain. The syntax is YYYYMMDDRR; that is, a four digit year, two digit month in numerical form, two digit day format, and a two digit revision number. In this example (1998082403), it shows that the zone file was last modified on August 24, 1998. It's the third revision for that day. When you're changing anything in the file, make sure to increase the revision number by one if the previous change was on the same day. If I were to change the IP of one of the hosts, I would make the last two numbers, currently 03, to 04.

The next few lines show times for certain functions such as refreshing, retrying, and expiring the information. I'm not *absolutely* sure, but my best guess is that H stands for hour, D stands for day, and W stands for week.

The "NS" line indicates all the nameservers for that particular domain, including the one this information is on. This information has to match what has been registered with InterNIC. For the hostnames of the nameservers, remember to add a dot at the end. If you don't, it will add the hostname to the current domain. For example, if you forgot the dot at the end of pasta.penguincomputing.com, you would end up with the nameserver being pasta.penguincomputing.com.penguincomputing.com, which is obviously not what it's supposed to be. Watch out for this.

The MX file is the *M*ail e*X*change record, so that mail can get through to the domain. There should also be an entry in /etc/sendmail.cw to allow messages coming in from that domain (assuming you're using Sendmail, the default on many Linux systems, for mail transfer).

The next couple of lines point to the local loopback (127.0.0.1), which all Linux systems should have even if they aren't connected to a network. The "router" line points to the IP address of where the machine's Internet connection is. I'm not sure if it's really necessary but I was playing it safe back then and trying to copy the example from the DNS HOWTO as closely as possible.

The rest of the entries use either A (address) or CNAME (Canonical Name) to point hostnames to IP addresses. Note that hostnames can be mapped to other hostnames, or they can be mapped to IP addresses. Use A to map a name to an IP address, and CNAME to map a hostname to another hostname (which must be mapped to another IP address).

Localhost

The file for mapping localhost is pretty simple. Not much explanation needed. Of course, if you want to copy and paste, be sure you make the proper changes.

```
@       IN      SOA     penguincomputing.com root.penguincomputing.com (
                        1998072401      ;       Serial number
                        3H              ;       Refresh
                        1H              ;       Retry
                        604800          ;       Expire
                        86400)          ;       Minimum TTL

                NS      pasta.penguincomputing.com.
                NS      rice.penguincomputing.com.

1               PTR     localhost.
```

Reverse Mapping

This file looks similar to the zone file for the domains, but it provides the opposite function. It points IP addresses to hostnames (as opposed to vice versa), because many servers on the Internet do this thing called reverse lookup on the IP address of your hostname to make sure that you're not doing anything sneaky.

This is for the zone "209.81.10" specified in the sample configuration file. Note that my example is not complete, nor does it work in reality, because Penguin Computing doesn't own the whole block of "209.81.10.*". But this is how you'd fill in a file to resolve your IP addresses to hostnames *if* you owned the entire block of IP addresses.

```
@                IN      SOA     penguincomputing.com.
        root.penguincomputing.com. (
                                 1998072002       ; Serial
                                 4H     ; Refresh
                                 2H     ; Retry
                                 604800  ; Expire
                                 86400)  ; Minimum TTL
                         NS      pasta.penguincomputing.com.
                         NS      rice.penguincomputing.com.
;
;       Servers
;
250     PTR     pasta.penguincomputing.com.
250     PTR     penguincomputing.com.
250     PTR     ftp.penguincomputing.com.
250     PTR     www.penguincomputing.com.
250     PTR     mail.penguincomputing.com.
251     PTR     rice.penguincomputing.com.

;
;       Workstations
;
252     PTR     antarctica.penguincomputing.com.
252     PTR     antarctic.penguincomputing.com.
```

If you were to fill in an actual zone file like this, it's necessary to fill in *all* the entries in your block of IP addresses, from 1 to 255. For something like that you may want to assign the task to anyone who looks bored.

So what should you do if you only own a domain but not the block of IP addresses that it's part of? Ask the people who are in charge of that block of IP addresses to map your IP addresses to their respective hostnames for you.

DNS zone transfer

DNS zone transfer, also sometimes known by the inducing DNS query type **AXFR**, is a type of DNS transaction. It is one of the many mechanisms available for administrators to replicate DNS databases across a set of DNS servers. Zone transfer comes in two flavors, full (AXFR[RFC 1035]) and incremental (IXFR[RFC 1995]). Nearly universal at one time, it is now becoming less popular in favor of the use of other database replication mechanisms that modern DNS server packages provide.

Operation

Zone transfer operates on top of the Transmission Control Protocol (TCP), and takes the form of a client–server transaction. The parties involved in a zone transfer are a client (the "slave" requesting the data from a portion of the database to be transferred to it) and a server (the "master" supplying those data from its database). Some sources refer to the slave as a "secondary" server and the master as a "primary" server. The portion of the database that is replicated is a "zone".

Zone transfer comprises a preamble followed by the actual data transfer. The preamble comprises a lookup of the SOA (Start of Authority) resource record for the "zone apex", the node of the DNS namespace that is at the top of the "zone". The fields of this SOA resource record, in particular the "serial number", determine whether the actual data transfer need occur at all. The client compares the serial number of the SOA resource record with the serial number in the last copy of that resource record that it has. If the serial number of the record being transferred is greater, the data in the zone are deemed to have "changed" (in some fashion) and the slave proceeds to request the actual zone data transfer. If the serial numbers are identical, the data in the zone are deemed not to have "changed", and the client may continue to use the copy of the database that it already has, if it has one.

The actual data transfer proper begins by the client sending a query (opcode 0) with the special QTYPE (query type) AXFR (value 252) over the TCP connection to the server. The server responds with a series of response messages, comprising all of the resource records for every domain name in the "zone". The first response comprises the SOA resource record for the zone apex. The other data follow in no specified order. The end of the data is signalled by the server repeating the response containing the SOA resource record for the zone apex.

Some zone transfer clients perform the SOA lookup of the preamble using their system's normal DNS query resolution mechanism. These clients do not open a TCP connection to the server until they have determined that they need to perform the actual data transfer. However, since TCP can be used for normal DNS transactions, as well as for zone transfer, other zone transfer clients perform the SOA lookup preamble over the same TCP connection as they then (may) perform the actual data transfer. These clients open the TCP connection to the server before they even perform the preamble.

The preceding describes full zone transfer. Incremental zone transfer differs from full zone transfer in the following respects:

- The client uses the special QTYPE IXFR (value 251) instead of the AXFR QTYPE.
- The client sends the SOA resource record for the zone apex that it currently has, if any, in the IXFR message, letting the server know which version of the "zone" it believes to be current.
- Though the server may respond in the normal AXFR manner with the full data for the zone, it may also instead respond with an "incremental" data transfer. This latter comprises the list of changes to the zone data, in zone serial number order, between the version of the zone that the client reported to the server as having and the version of the zone that is current at the server. The changes comprise two lists, one of resource records that are deleted and one of resource records that are inserted. (A modification to a resource record is represented as a deletion followed by an insertion.)

Zone transfer is entirely client-initiated. Though servers can send a NOTIFY message to clients (that they have been informed about) whenever a change to the zone data has been made, the scheduling of zone transfers is entirely under the control of the clients. Clients schedule zone transfers initially, when their databases are empty, and thereafter at regular intervals, in a pattern controlled by the values in the "refresh", "retry", and "expire" fields in the SOA resource record of the zone apex.

Limitations

Though it is standardized, full-zone transfer being described as one of the possible database replication mechanisms in RFC 1034 (incremental zone transfer described in RFC 1995), zone transfer is the most limited of those database replication mechanisms. Zone transfer operates in terms of "wire format" resource records, *i.e.* resource records as they are transferred using the DNS protocol. However, the schema of wire format resource records may not match the database schema used by the back ends of the DNS servers themselves.

Dynamic update

Applies To: Windows Server 2003, Windows Server 2003 R2, Windows Server 2003 with SP1, Windows Server 2003 with SP2

Dynamic update

Dynamic update enables DNS client computers to register and dynamically update their resource records with a DNS server whenever changes occur. This reduces the need for manual administration of zone records, especially for clients that frequently move or change locations and use DHCP to obtain an IP address.

The DNS Client and Server services support the use of dynamic updates, as described in Request for Comments (RFC) 2136, "Dynamic Updates in the Domain Name System." The DNS Server service allows dynamic update to be enabled or disabled on a per-zone basis at each server configured to load either a standard primary or directory-integrated zone. By default, the DNS Client service will dynamically update host (A) resource records (RRs) in DNS when configured for TCP/IP. For more information about RFCs, see DNS RFCs.

How client and server computers update their DNS names

By default, computers that are statically configured for TCP/IP attempt to dynamically register host (A) and pointer (PTR) resource records (RRs) for IP addresses configured and used by their installed network connections. By default, all computers register records based on their fully qualified domain name (FQDN).

The primary full computer name, a FQDN, is based on the **primary DNS suffix of a computer** appended to its **Computer name**.

Both of these settings are displayed or configured from the **Computer Name** tab in **System properties**. For more information, see View system properties.

Notes

- By default, the DNS client on Windows XP does not attempt dynamic update over a Remote Access Service (RAS) or virtual private network (VPN) connection. To modify this configuration, you can modify the advanced TCP/IP settings of the particular network connection or modify the registry. For more information, see Configure TCP/IP to use DNS and the Microsoft Windows Resource Kits Web site.
- By default, the DNS client does not attempt dynamic update of top-level domain (TLD) zones. Any zone named with a single-label name is considered a TLD zone, for example, com, edu, blank, my-company. To configure the DNS client to allow the dynamic update of TLD zones, you can use the **Update Top Level Domain Zones** policy setting or modify the registry.

- By default, the primary DNS suffix portion of a computer's FQDN is the same as the name of the Active Directory domain to which the computer is joined. To allow different primary DNS suffixes, a domain administrator may create a restricted list of allowed suffixes by modifying the **msDS-AllowedDNSSuffixes** attribute in the domain object container. This attribute is managed by the domain administrator using Active Directory Service Interfaces (ADSI) or the Lightweight Directory Access Protocol (LDAP).

  For more information, see Programming interfaces and Directory access protocol.

Dynamic updates can be sent for any of the following reasons or events:

- An IP address is added, removed, or modified in the TCP/IP properties configuration for any one of the installed network connections.
- An IP address lease changes or renews with the DHCP server any one of the installed network connections. For example, when the computer is started or if the **ipconfig /renew** command is used.
- The **ipconfig /registerdns** command is used to manually force a refresh of the client name registration in DNS.
- At startup time, when the computer is turned on.
- A member server is promoted to a domain controller.

When one of the previous events triggers a dynamic update, the DHCP Client service (not the DNS Client service) sends updates. This is designed so that if a change to the IP address information occurs because of DHCP, corresponding updates in DNS are performed to synchronize name-to-address mappings for the computer. The DHCP Client service performs this function for all network connections used on the system, including connections not configured to use DHCP.

Notes

- The process of how dynamic updates are performed for computers running Windows 2000, Windows XP, or Windows Server 2003 operating systems that use DHCP to obtain their IP address is different than is described in this section. For more information, see Using DNS servers with DHCP.
- The update process described in this section assumes that installation defaults are in effect for computers running Windows 2000, Windows XP, or servers running Windows Server 2003. Specific names and update behavior is tunable where advanced TCP/IP properties are configured to use non-default DNS settings.

  In addition to the full computer name (or primary name) of the computer, additional connection-specific DNS names can be configured and optionally registered or updated in DNS. For more information, see Configuring multiple names or Configure TCP/IP to use DNS.

Example: How dynamic update works

Dynamic updates are typically requested when either a DNS name or IP address changes on the computer. For example, suppose a client named "oldhost" is first configured in **System properties** with the following names:

Computer name	oldhost
DNS domain name of computer	example.microsoft.com
Full computer name	oldhost.example.microsoft.com

In this example, no connection-specific DNS domain names are configured for the computer. Later, the computer is renamed from "oldhost" to "newhost", resulting in the following name changes on the system:

Computer name	newhost
DNS domain name of computer	example.microsoft.com
Full computer name	newhost.example.microsoft.com

Once the name change is applied in **System properties**, you are prompted to restart the computer. When the computer restarts Windows, the DHCP Client service performs the following sequence to update DNS:

1. **The DHCP Client service sends a start of authority (SOA) type query using the DNS domain name of the computer.**

   The client computer uses the currently configured FQDN of the computer (such as "newhost.example.microsoft.com") as the name specified in this query.

2. **The authoritative DNS server for the zone containing the client FQDN responds to the SOA-type query.**

   For standard primary zones, the primary server (owner) returned in the SOA query response is fixed and static. It always matches the exact DNS name as it appears in the SOA RR stored with the zone. If, however, the zone being updated is directory-integrated, any DNS server loading the zone can respond and dynamically insert its own name as the primary server (owner) of the zone in the SOA query response.

3. **The DHCP Client service then attempts to contact the primary DNS server.**

   The client processes the SOA query response for its name to determine the IP address of the DNS server authorized as the primary server for accepting its name. It then proceeds to perform the following sequence of steps as needed to contact and dynamically update its primary server:

   1. It sends a dynamic update request to the primary server determined in the SOA query response.

If the update succeeds, no further action is taken.

2. If this update fails, the client next sends an NS-type query for the zone name specified in the SOA record.
3. When it receives a response to this query, it sends an SOA query to the first DNS server listed in the response.
4. After the SOA query is resolved, the client sends a dynamic update to the server specified in the returned SOA record.

   If the update succeeds, no further action is taken.
5. If this update fails, then the client repeats the SOA query process by sending to the next DNS server listed in the response.

4. **Once the primary server is contacted that can perform the update, the client sends the update request and the server processes it.**

   The contents of the update request include instructions to add A (and possibly PTR) RRs for "newhost.example.microsoft.com" and remove these same record types for "oldhost.example.microsoft.com", the name that was previously registered.

   The server also checks to ensure that updates are permitted for the client request. For standard primary zones, dynamic updates are not secured, so any client attempt to update succeeds. For Active Directory-integrated zones, updates are secured and performed using directory-based security settings.

Dynamic updates are sent or refreshed periodically. By default, computers send a refresh once every 7 days. If the update results in no changes to zone data, the zone remains at its current version and no changes are written. Updates result in actual zone changes or increased zone transfer only if names or addresses actually change.

Note that names are not removed from DNS zones if they become inactive or are not updated within the refresh interval (7 days). DNS does not use a mechanism to release or tombstone names, although DNS clients do attempt to delete or update old name records when a new name or address change is applied.

When the DHCP Client service registers A and PTR resource records for a computer, it uses a default caching Time to Live (TTL) of 15 minutes for host records. This determines how long other DNS servers and clients cache a computer's records when they are included in a query response.

Delegating Domain Names

Domain delegation means placement of DNS servers list with the domain technical records (zone file) to DNS servers enabling top level domains functioning. Delegation is a prerequisite for site and mail operation on the domain.

DNS Servers List for RU-CENTER Services Functioning

- For domain delegation to Hosting
- For domain and mail redirection services
- For DNS server Primary-Standard, Primary-Auto and Secondary services

Delegation Periods, Suspending/Unsuspending

Within the whole domain registration period the Registrant may suspend delegation in "Manage your account" → My domains" (for gTLDs and ccTLDs the domain should be delegated with an empty DNS servers list).

Earlier suspended domain will be unsuspended subject to domains delegation periods.

Domains delegation periods are subject to the moment the modifications are made:

- for .RU domain - up to 9 hours;
- for .РФ and .SU domains - up to 2 hours;
- for third-level domains - up to 1 hour;
- delegation of gTLDs and ccTLDs takes several minutes.

DNS Servers Testing

During domain delegation the specified DNS servers may be checked for operation. If testing proves successful the domain will be delegated with the new DNS servers.

How to Complete DNS Servers List

- specify at least two DNS servers;
- if you specify DNS servers, containing the name of the IDN, the name of this IDN should be entered with "XN--" prefix, but not in national script;

Irregular Cases

For DNS servers, which name contains the domain name to be delegated (child DNS server) IP address should be specified; .PRO domain supports only IPv4; For .ORG domains delegation with child DNS servers different IP addresses need to be specified.

What is dns delegation?

When the authoritative name server for a domain receives a request for a subdomain's records and responds with NS records for other name servers, that is DNS delegation. Essentially it is saying "I am passing on authority for this subdomain to another collection of name servers, go ask them for the details."

For example, "com" is a domain that delegates all (or perhaps almost all) of its subdomains to other name servers. When a request is received at a "com" name server for "google.com", the "com" name server responds with NS records pointing to Google's name servers.

The dig command with the +trace option will demonstrate. Try `dig +trace google.com` and part of it will look like this:

```
google.com. 172800 IN NS ns2.google.com.
google.com. 172800 IN NS ns1.google.com.
google.com. 172800 IN NS ns3.google.com.
google.com. 172800 IN NS ns4.google.com.
;; Received 164 bytes from 192.31.80.30#53(d.gtld-servers.net) in 150 ms
```

Those NS records indicate that those 4 Google name servers are authoritative for the google.com domain.

Unit 6

Introduction to Apache Server

Apache is probably the most popular Linux-based Web server application in use. Once you have DNS correctly setup and your server has access to the Internet, you'll need to configure Apache to accept surfers wanting to access your Web site.

Download and Install The Apache Package

Most RedHat and Fedora Linux software products are available in the RPM format. When searching for the file, remember that the Apache RPM's filename usually starts with the word httpd followed by a version number, as in httpd-2.0.48-1.2.rpm. It is best to use the latest version of Apache. When searching for the file, remember that the Redhat / Fedora Apache RPM package's filename usually starts with the word `httpd` followed by a version number, as in `httpd-2.0.48-1.2.rpm`. With Ubuntu / Debian the package name will have the `apache` prefix instead.

Managing the Apache Server

Managing Apache's httpd daemon is easy to do, but the procedure differs between Linux distributions. Here are some things to keep in mind.

1. Firstly, different Linux distributions use different daemon management systems. Each system has its own set of commands to do similar operations. The most commonly used daemon management systems are SysV and Systemd.
2. Secondly, the daemon name needs to be known. In this case the name of the daemon is **httpd**.

Armed with this information you can know how to:

1. Start your daemons automatically on booting
2. Stop, start and restart them later on during troubleshooting or when a configuration file change needs to be applied.

Configuring DNS For Apache

Remember that you will never receive the correct traffic unless you configure DNS for your domain to make your new Linux box Web server the target of the DNS domain's www entry. To do this, refer to Chapter 18, "Configuring DNS", or Chapter 19, "Dynamic DNS".

DHCP and Apache

As you remember, if your Internet connection uses DHCP to get its IP address, then you need to use dynamic DNS to get the correct Internet DNS entry for your Web server. If your Web server and firewall are different machines, then you probably also need to set up port forwarding for your Web traffic to reach the Web server correctly. (Chapter 19, "Dynamic DNS", explains port forwarding, as well.).

DHCP on your protected home network is different. In the book's sample topology, the web server lives on the 192.168.1.0 home network protected by a firewall. The firewall uses NAT and port forwarding to pass Internet traffic on to the web server. Remember that the IP address of your web server can change if it gets its IP address using DHCP. This could cause your firewall port forwarding, not Dynamic DNS, to break.

In this case I recommend that your web server on the 192.168.1.0 network uses a fixed, or static IP address that is outside of the range of the DHCP server to prevent you from having this problem.

General Configuration Steps

The configuration file used by Apache is `/etc/httpd/conf/httpd.conf` in Redhat / Fedora distributions and `/etc/apache*/httpd.conf` in Debian / Ubuntu distributions. As for most Linux applications, you must restart Apache before changes to this configuration file take effect.

Where To Put Your Web Pages

All the statements that define the features of each web site are grouped together inside their own <VirtualHost> section, or container, in the httpd.conf file. The most commonly used statements, or directives, inside a <VirtualHost> container are:

- **servername**: Defines the name of the website managed by the <VirtualHost> container. This is needed in named virtual hosting only, as I'll explain soon.
- **DocumentRoot**: Defines the directory in which the web pages for the site can be found.

By default, Apache searches the DocumentRoot directory for an index, or home, page named index.html. So for example, if you have a servername of www.my-site.com with a DocumentRoot directory of /home/www/site1/, Apache displays the contents of the file /home/www/site1/index.html when you enter http://www.my-site.com in your browser.

Some editors, such as Microsoft FrontPage, create files with an .htm extension, not .html. This isn't usually a problem if all your HTML files have hyperlinks pointing to files ending in .htm as FrontPage does. The problem occurs with Apache not recognizing the topmost index.htm page. The easiest solution is to create a symbolic link (known as a shortcut to Windows users) called index.html pointing to the file index.htm. This then enables you to edit or copy the file index.htm with index.html being updated automatically. You'll almost never have to worry about index.html and Apache again!

This example creates a symbolic link to index.html in the /home/www/site1 directory.

```
[root@bigboy tmp]# cd /home/www/site1
[root@bigboy site1]# ln -s index.htm index.html
[root@bigboy site1]# ll index.*
-rw-rw-r--    1 root     root        48590 Jun 18 23:43 index.htm
lrwxrwxrwx    1 root     root            9 Jun 21 18:05 index.html ->
index.htm
[root@bigboy site1]#
```

The l at the very beginning of the index.html entry signifies a link and the -> the link target.

The Default File Location

By default, Apache expects to find all its web page files in the /var/www/html/ directory with a generic DocumentRoot statement at the beginning of httpd.conf. The examples in this chapter use the /home/www directory to illustrate how you can place them in other locations successfully.

File Permissions And Apache

Apache will display Web page files as long as they are world readable. You have to make sure you make all the files and subdirectories in your DocumentRoot have the correct permissions.

It is a good idea to have the files owned by a nonprivileged user so that Web developers can update the files using FTP or SCP without requiring the root password.

To do this:

1. Create a user with a home directory of /home/www.
2. Recursively change the file ownership permissions of the /home/www directory and all its subdirectories.
3. Change the permissions on the /home/www directory to 755, which allows all users, including the Apache's httpd daemon, to read the files inside.

```
[root@bigboy tmp]# useradd -g users www
[root@bigboy tmp]# chown -R www:users /home/www
[root@bigboy tmp]# chmod 755 /home/www
```

Now we test for the new ownership with the ll command.

```
[root@bigboy tmp]# ll /home/www/site1/index.*
-rw-rw-r--    1 www     users       48590 Jun 25 23:43 index.htm
lrwxrwxrwx    1 www     users           9 Jun 25 18:05 index.html ->
index.htm
[root@bigboy tmp]#
```

Note: Be sure to FTP or SCP new files to your web server as this new user. This will make all the transferred files automatically have the correct ownership.

If you browse your Web site after configuring Apache and get a "403 Forbidden" permissions-related error on your screen, then your files or directories under your DocumentRoot most likely have incorrect permissions. Appendix II, "Codes, Scripts, and Configurations," has a short script that you can use to recursively set the file permissions in a directory to match those expected by Apache. You may also have to use the Directory directive to make Apache serve the pages once the file permissions have been correctly set. If you have your files in the default /home/www directory then this second step becomes unnecessary.

Virtual hosting

Virtual hosting is a method for hosting multiple domain names (with separate handling of each name) on a single server (or pool of servers). This allows one server to share its resources, such as memory and processor cycles, without requiring all services provided to use the same host name. The term virtual hosting is usually used in reference to web servers but the principles carry over to other internet services.

One widely used application is shared web hosting. Shared web hosting prices are lower than a dedicated web server because many customers can be hosted on a single server. It is also very common for a single entity to want to use multiple names on the same machine so that the names can reflect services offered rather than where those services happen to be hosted.

There are two main types of virtual hosting, name based and IP based. Name based virtual hosting uses the host name presented by the client. This saves IP addresses and the associated administrative overhead but the protocol being served must supply the host name at an appropriate point. In particular there are significant difficulties using name based virtual hosting with SSL/TLS. IP based virtual hosting uses a separate IP address for each host name, and it can be performed with any protocol but requires a dedicated IP address per domain name served. Port based virtual hosting is also possible in principle but is rarely used in practice because it is unfriendly to users.

Name based and IP based virtual hosting can be combined, a server may have multiple IP address and serve multiple names on some or all of those IP addresses. This technique can be useful when using SSL/TLS with wildcard certificates. For example if a server operator had two certificates one for *.example.com and one for *.example.net he could serve foo.example.com and bar.example.com off the same IP address but would need a separate IP address for baz.example.net.

Name-based

Name-based virtual hosts use multiple host names for the same IP address.

A technical prerequisite needed for name-based virtual hosts is a web browser with HTTP/1.1 support (commonplace today) to include the target hostname in the request. This allows a server hosting multiple sites behind one IP address to deliver the correct site's content. More specifically it means setting the `Host`HTTP header.

For instance, a server could be receiving requests for two domains, `www.example.com` and `www.example.net`, both of which resolve to the same IP address. For `www.example.com`, the server would send the HTML file from the directory `/var/www/user/Joe/site/`, while requests for `www.example.net` would make the server serve pages from `/var/www/user/Mary/site/`. Equally two subdomains of the same domain may be hosted together for example a blog server may host blog1.example.com and blog2.example.com

The biggest issue with name based virtual hosting is that it is difficult to host multiple secure websites running SSL/TLS. Because the SSL/TLS handshake takes place before the expected hostname is sent to the server, the server doesn't know which certificate to present in the handshake. It is possible for a single certificate to cover multiple names either through the "subjectaltname" field or through wildcards but the practical application of this approach is limited by administrative considerations and by the matching rules for wildcards. There is an extension to TLS called Server Name Indication which presents the name at the start of the handshake but browser support for this extension is not yet wide enough for public sites to rely on it (in particular it is not supported by Internet explorer on Windows XP).

Furthermore if the Domain Name System (DNS) is not properly functioning, it is difficult to access a virtually-hosted website even if the IP address is known. If the user tries to fall back to using the IP address to contact the system, as in `http://10.23.45.67/`, the web browser will send the IP address as the host name. Since the web server relies on the web browser client telling it what server name (vhost) to use, the server will respond with a default website—often not the site the user expects.

A workaround in this case is to add the IP address and host name to the client system's hosts file. Accessing the server with the domain name should work again. Users should be careful when doing this, however, as any changes to the true mapping between host name and IP address will be overridden by the local setting. This workaround is not really useful for an average web user, but may be of some use to a site administrator while fixing DNS records.

IP-based

When IP-based virtual hosting is used, each site (either a DNS host name or a group of DNS host names that act the same) points to a unique IP address. The webserver is configured with multiple physical network interfaces, virtual network interfaces on the same physical interface or multiple IP addresses on one interface.

The web server can either open separate listening sockets for each IP address or it can listen on all interfaces with a single socket and obtain the address the TCP connection was received on after accepting the connections. Either way it can use the IP address to determine which website to serve. The client is not involved in this process and therefore (unlike with name based virtual hosting) there are no compatibility issues.

The downside of this approach is the server needs a different IP address for every web site. This increases administrative overhead (both assigning addresses to servers and justifying the use of those addresses to internet registries) and contributes to IPv4 address exhaustion.

Port-based

The default port number for HTTP is 80. However, most webservers can be configured to operate on almost any port number, provided the port number is not in use by any other program on the server.

For example, a server may host the website `www.example.com`. However, if the owner wishes to operate a second site, and does not have access to the domain name configuration for their domain name, and/or owns no other IP addresses which could be used to serve the site from, they could instead use another port number, for example, `www.example.com:81` for port 81, `www.example.com:8000` for port 8000, or `www.example.com:8080` for port 8080.

However this is not a user unfriendly approach. Users cannot reasonably be expected to know the port numbers for their websites and moving a site between servers may require changing the port number. Using non-standard port numbers may also be seen as unprofessional and unattractive to users. In addition, some firewalls block all but the most common ports, causing a site hosted on a non-standard port to appear unavailable to some users.

Uses

Virtual web hosting is often used on large scale in companies whose business model is to provide low cost website hosting for customers. The vast majority of web hosting service customer websites worldwide are hosted on shared servers, using virtual hosting technology.

Many businesses utilize virtual servers for internal purposes, where there is a technological or administrative reason to operate several separate websites, such as a customer extranet website, employee extranet, internal intranet, and intranets for different departments. If there are not security concerns in the website architectures, they can be merged into a single server using virtual hosting technology, which reduces management and administrative overhead and the number of separate servers required to support the business.

HTTP Caching Features and Issues

The explosive growth of the World Wide Web was a marvel for its users, but a nightmare for networking engineers. The biggest problem that the burgeoning Web created was an overloading of the internetworks over which it ran. Many of the features that were added to HTTP/1.1 were designed specifically to improve the efficiency of the protocol and reduce unnecessary bandwidth consumed by HTTP requests and responses. Arguably the most important of these is a set of features designed to support *caching*.

The subject of caching comes up again and again in discussions of computers and networking, because of a phenomenon that is widely observed in these technologies:

whenever a user, hardware device or software process requests a particular piece of data, there is a good chance it will ask for it again in the near future. Thus, by storing recently-retrieved items in a cache, we can eliminate duplicated effort. This is why caching plays an important role in the efficiency of protocols such as ARP and DNS.

The Significance of Caching to HTTP

Caching is important to HTTP because Web users tend to request the same documents over and over again. For example, in writing this section on HTTP, I made reference to RFC 2616 many, many times. Each time, I loaded it from a particular Web server. Since the document never changes, it would be more efficient to just load it from a local cache rather than having to retrieve it from the distant Web server each time.

However, caching is even more essential to HTTP than to most other protocols or technologies where it used. The reason is that Web documents tend to be structured so that a request for one resource leads to a request for many others. Even if I load a number of ***different*** documents, they may each refer to common elements that do not change between user requests. Thus, caching can be of benefit in HTTP even if a user never asks for the same document twice, or if a single document changes over time so that caching the document itself would be of little value.

For example, suppose that each morning I load up CNN's Web site to see what is going on in the world. Obviously, the headlines will be different every day, so caching of the main CCN.com Web home page won't be of much value. However, many of the graphical elements on the page (CNN's logo, dividing bars, perhaps a "breaking news" graphic) will be the same, every day, and these can be cached. Another example would be a set of discussion forums on a Web site. As I load up different topics to read, each one is different, but they have common elements (such as icons and other images) that would be wasteful to have to retrieve over and over again.

Caching in HTTP yields two main benefits. The first is reduced bandwidth use, by eliminating unneeded transfers of requests and responses. The second, equally important, is faster response time for the user loading a resource. Consider that on many Web pages today, the image files are much larger than the HTML page that references them. Caching these graphics will allow the entire page to load far more quickly. Figure 319 illustrates how caching reduces bandwidth and speeds up resource retrieval by "short-circuiting" the request/response chain.

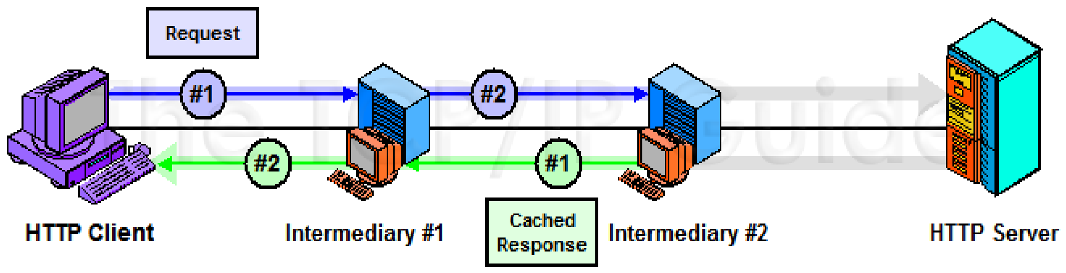


Figure 319: Impact of Caching on HTTP Request/Response Chain

This diagram illustrates the impact of caching on the request/response chain of Figure 316. In this example, intermediary #2 is able to satisfy the client's request from its cache. This "short-circuits" the communication chain after two transfers, which means the client gets its resource more quickly, and the HTTP server is spared the need to process the client's request.

The obvious advantages of caching have made it a part of the Web since pretty much the beginning. However, it was not until HTTP/1.1 that the importance of caching was really recognized in the protocol itself, and many features added to support it. Where the HTTP/1.0 standard makes passing mention of caching and some of the issues related to it, HTTP/1.1 devotes 26 full pages to caching (over 20% of the main body of the document!) I obviously cannot go into that level of detail here, but I will discuss HTTP caching in general terms to give you a feel for the subject.

squid - proxy caching server

Synopsis

squid [-dhisrvzCFNRSVYX] [-lfacility] [-f config-file] [-[au] port] [-k " signal"] [-n service-name] [-O cmd-line]

Description

squid is a high-performance proxy caching server for web clients, supporting FTP, gopher, ICAP, ICP, HTCP and HTTP data objects. Unlike traditional caching software, squid handles all requests in a single, non-blocking process.

squid keeps meta data and especially hot objects cached in RAM, caches DNS lookups, supports non-blocking DNS lookups, and implements negative caching of failed requests.

squid supports SSL, extensive access controls, and full request logging. By using the lightweight Internet Cache Protocols ICP, HTCP or CARP, squid caches can be arranged in a hierarchy or mesh for additional bandwidth savings.

squid consists of a main server program squid, some optional programs for custom processing and authentication, and some management and client tools. When squid starts up, it spawns a configurable number of helper processes, each of which can perform parallel lookups. This reduces the amount of time the cache waits for results. squid is derived from the ARPA-funded Harvest Project http://harvest.cs.colorado.edu/ This manual page only lists the command line arguments. For details on how to configure squid see the file @DEFAULT_CONFIG_FILE@.documented, the Squid wiki FAQ and examples at http://wiki.squid-cache.org/ , or the configuration manual on the squid home page http://www.squid-cache.org/Doc/config/

Options

-a port Specify HTTP port number where Squid should listen for requests, in addition to any http_port specifications in squid.conf.

-d level Write debugging to stderr also.

-f file Use the given config-file instead of @DEFAULT_CONFIG_FILE@. If the file name starts with a ! or | then it is assumed to be an external command or command line. Can for example be used to pre-process the configuration before it is being read by Squid. To facilitate this Squid also understands the common #line notion to indicate the real source file.

-h Print help message.

-i Install as a Windows Service (see -n option).

-k reconfigure | rotate | shutdown | interrupt | kill | debug | check | parse -

Parse configuration file, then send signal to running copy (except -k parse) and exit.

-n name Specify Windows Service name to use for service operations, default is: Squid

-r Remove a Windows Service (see -n option).

-s Enable logging to syslog. Also configurable in @DEFAULT_CONFIG_FILE@

-l facility Use specified syslog facility. implies -s

-u port Specify ICP port number (default: 3130), disable with 0.

-v Print version and build details.

-z Create swap directories

-C Do not catch fatal signals.

-F	**Don't serve any requests until store is rebuilt.**
-N	**No daemon mode.**
-O options	**Set Windows Service Command line options in Registry.**
-R	**Do not set REUSEADDR on port.**
-S	**Double-check swap during rebuild.**
-X	**Force full debugging.**
-Y	**Only return UDP_HIT or UDP_MISS_NOFETCH during fast reload.**

Files

@DEFAULT_CONFIG_FILE@

The main configuration file. You must initially make changes to this file for squid to work. For example, the default configuration does not allow access from any browser.

***.default files**

Reference copy of the configuration file. Always kept up to date with the version of Squid you are using. Use this to look up the default configuration settings and syntax after upgrading.

@DEFAULT_CONFIG_FILE@.documented

Reference copy of the configuration file. Always kept up to date with the version of Squid you are using. Use this to read the documentation for configuration options available in your build of Squid. The online configuration manual is also available for a full reference of options. seehttp://www.squid-cache.org/Doc/config/

cachemgr.conf **The main configuration file for the web cachemgr.cgi tools.**

msntauth.conf **The main configuration file for the Sample MSNT authenticator.**

errorpage.css

CSS Stylesheet to control the display of generated error pages. Use this to set any company branding you need, it will apply to every Language Squid provides error pages for.

@DEFAULT_MIME_TABLE@ (mime_table) **MIME type mappings for FTP gatewaying**

@DEFAULT_ERROR_DIR@ **Location of squid error pages and templates.**

Proxy Authentication

We implement proxy servers to improve internal network security. Mainly we're attempting to keep bad traffic from communicating with the outside world. If malware lands on an internal box, it may for example attempt a call back to its command and control server. Typically this occurs over port 443 and is lost in the sea of legitimate traffic because your typical firewall is rather dumb. Compared to a proxy it does not inspect all 7 layers of the OSI model. It doesn't understand there is a covert channel pretending to be SSL encrypted traffic.

By funneling all outbound traffic through a proxy we can stop most of these bad packets. Often malware is not proxy aware and it will attempt to escape your network through the default gateway (typically your firewall). When implementing a proxy server, you should be adding a DENY ALL rule blocking ALL hosts from leaving out your front door. While this sounds like a great solution, there is a small loophole. Modern malware is slowly becoming proxy aware. Hence some bad traffic may evade even your smart proxy server trough a covert channel especially when configured in transparent mode.

Enter, Proxy Authentication to the rescue. It's a simple concept, before a proxy accepts your request you have to present your credentials and be authenticated, otherwise your packets become lost in space, hence no reverse shell connection to a command and control server. I've talked about Squid, and its many features and benefits. And you guessed it this week I'll discuss enabling Squid Proxy Authentication.

Squid supports the following authentication mechanisms;

- Basic – Simplest to implement but most insecure (clear text username and password)
- Digest – Easy to implement and more secure using MD5 hashed username and password
- NTLM – Harder to implement using Microsoft's proprietary SMB protocol
- Negotiate – Hardest to implement but most secure with Kerberos and AD

Depending on your environment and requirements I would suggest NTLM or Negotiate for most Windows centric networks. Negotiate with Kerberos would be my first choice but there are a few hoops to jump through. NTLM is a good second choice but because Microsoft hasn't been forth coming with its proprietary protocol implementation, it's been reverse engineered by some very smart people.

Over the next several weeks I'll take you through each authentication mechanism in more detail. If you're currently not using Proxy authentication with your proxy I would strongly suggest you begin to think about it and plan to implement it in the near future. As malware evolves, so should we and our defenses.

Till next week.

WHAT IS 'NTLM Authorization Proxy Server'?

'NTLM Authorization Proxy Server' (APS) is a proxy software that allows you to authenticate via an MS Proxy Server using the proprietary NTLM protocol. Since version 0.9.5 APS has an ability to behave as a standalone proxy server and authenticate http clients at web servers using NTLM method. It can change arbitrary values in your client's request header so that those requests will look like they were created by MS IE. It is written in Python v1.5.2 language.

Main features:

- supports NTLM authentication via parent proxy server (Error 407 Proxy Authentication Required);
- supports NTLM authentication at web servers (Error 401 Access Denied/Unauthorized);
- supports translation of NTLM scheme to standard "Basic" authentication scheme;
- supports the HTTPS 'CONNECT' method for transparent tunnelling through parent proxy server;
- has ability to change arbitrary values in client's request headers;
- supports unlimited number of client connections;
- supports connections from external hosts;
- supports HTTP 1.1 persistent connections;
- stores user's credentials in config file or requests password from a console during the start time;
- supports intelligent failure detection and failover between multiple upstream proxies;

The server had been written for wget that could not pass through MS Proxy set up in our LAN. But then it turned out that even browsers can use it, so I spend some time to get it more RFC friendly and now it looks like it works with most software that can use http/https proxies.

Even distributed Intel-United Devices Cancer Research Project can be used with APS. Just use HTTPS proxy in "proxy settings" of the United Devices' software and point to your local NTLMAPS, like server - "localhost" and port - "8080" or something that you set in *server.cfg*.

Licensing and Pricing:

'NTLM Authorization Proxy Server' is distributed under the GNUGeneral Public License which is included in this archive (see file COPYING).
The above mean that 'NTLM Authorization Proxy Server' is pretty much free. You have to pay nothing for it.

System requirements:

Python language interpreter version 1.5.2 or higher.

There are no binary files in the distribution. Thus you can use the software on any system that has Python, with minimal modifications.

NTLMAPS uses only standard modules from a Python distribution.

Troubleshooting:

There are two options in *server.cfg***DEBUG** and **BIN_DEBUG**, if you have toubles with the server so set these options to **DEBUG:1** and **BIN_DEBUG:1** just before requesting a problem page (or resource). You have to restart proxy server to reread *server.cfg*. This will give you 3 log files per http request (per connection to be exact), like *127.0.0.1-1048*, *127.0.0.1-1048.bin.client* and *127.0.0.1-1048.bin.rserver*. In the first one there is an info on what APS did, two others contain raw traffic from client and from proxy.

Configuring a Squid Server to authenticate from MySQL database

Install squid using your distro package management system or using source.
Make sure squid is compiled with *--enable-basic-auth-helpers=DB* option.

Creating MySQL db/table to hold user credentials

```
mysql> create database squid;
mysql> grant select on squid.* to someuser@localhost identified by 'your_passwd';
```

Create table 'passwd' in 'squid' db.

```
mysql> CREATE TABLE `passwd` (
 `user` varchar(32) NOT NULL default '',
 `password` varchar(35) NOT NULL default '',
 `enabled` tinyint(1) NOT NULL default '1',
 `fullname` varchar(60) default NULL,
 `comment` varchar(60) default NULL,
 PRIMARY KEY  (`user`)
);
```

Populate the table with some test data, eg

```
mysql> insert into passwd values('Nikesh','test',1,'Test User','for testing purpose');
```

Squid Configuration File
Edit squid.conf so that authentication against MySQL db works

```
auth_param basic program /usr/local/squid/libexec/squid_db_auth --user someuser --password
your_passwd --plaintext --persist
auth_param basic children 5
auth_param basic realm Web-Proxy
auth_param basic credentialsttl 1 minute
auth_param basic casesensitive off
```

```
acl db-auth proxy_auth REQUIRED
http_access allow db-auth
http_access allow localhost
http_access deny all
```

Proxy RADIUS Authentication

Figure 46 illustrates how RADIUS authentication messages are forwarded by proxy:

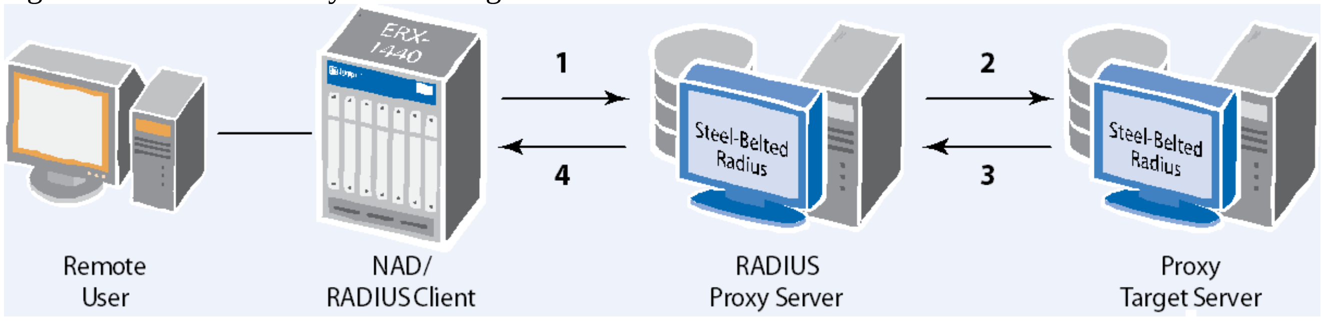
1. A network access device (RADIUS client) sends an authentication request to a RADIUS proxy server.

2. The proxy RADIUS server forwards the message to a RADIUS target server.

3. The target RADIUS server performs the authentication services indicated by the message, then returns a response message to the proxy RADIUS server.

4. The proxy RADIUS server relays the response message to the RADIUS client.

Figure 46: RADIUS Proxy Forwarding



Proxy RADIUS Accounting

RADIUS accounting messages are forwarded by proxy as follows:

1. A RADIUS server receives an accounting request.

2. Depending on its configuration, the RADIUS server forwards the accounting message to a target server, records accounting attributes locally on the proxy server, or records the information in both places.

3. If the proxy server does not receive an acknowledgement of the forwarded packet, it periodically re-sends the packet according to its retry policy.

4. When the target server acknowledges the request, the proxy server forwards an acknowledgement to the RADIUS client.

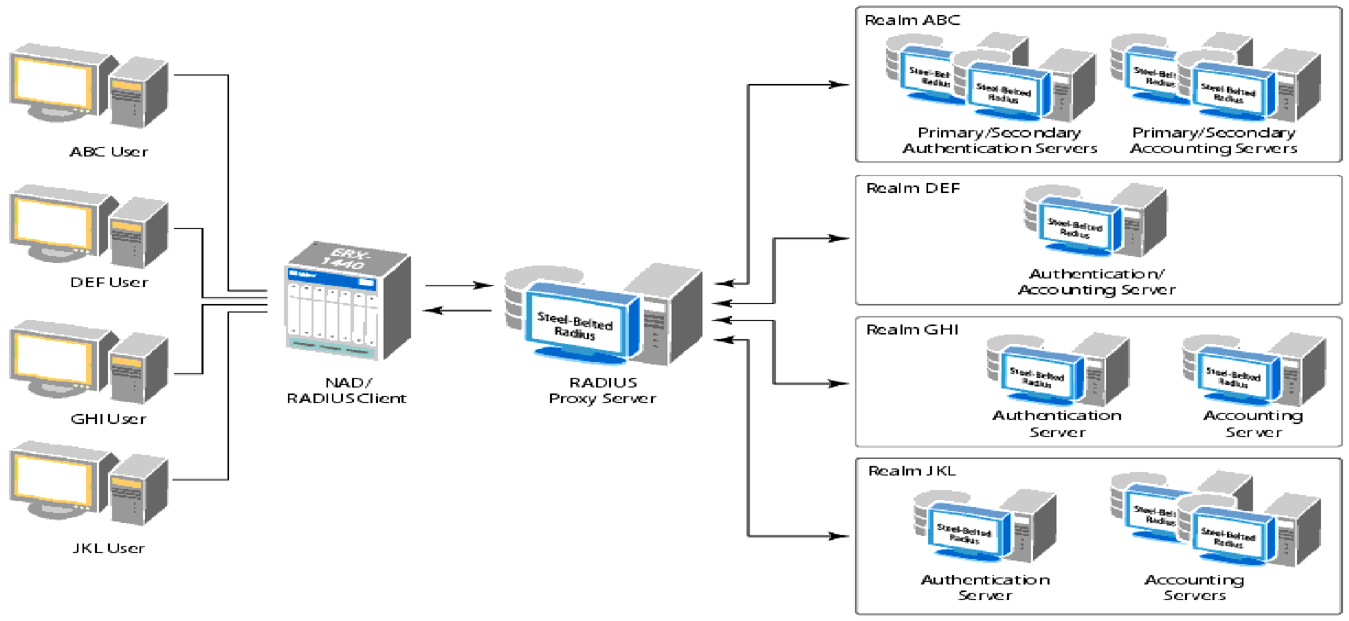
Proxy RADIUS Realms

Proxy RADIUS realms are pools of RADIUS servers to which Steel-Belted Radius Carrier can forward RADIUS requests. Proxy RADIUS realms can be configured to support workload distribution, redundancy, fault tolerance, retry policies, primary-secondary server roles, and separation of authentication and accounting responsibilities by server. A carrier with Steel-Belted Radius Carrier installed on its LAN might create a realm that consists of all the RADIUS servers owned by a particular service provider. In this case, the RADIUS servers are already owned and maintained by the service provider; the realm simply organizes the routing of RADIUS packets from Steel-Belted Radius Carrier to these servers.

The carrier might define several such realms, one for each service provider that employs its services. If a service provider's network is extremely large, a carrier might decide to use several realms to represent a single service provider. For each of these realms, it is possible to define an independent set of conventions for storing, forwarding, routing, and filtering the RADIUS requests that enter the Steel-Belted Radius Carrier server.

For more information, see Figure 47 and Configuring a Proxy RADIUS Realm163.

Figure 47: RADIUS Server and Proxy Realms



Target Selection Within a Realm

For proxy RADIUS realms, after the destination realm is identified, Steel-Belted Radius Carrier must select a target within the realm. Target selection depends upon a number of factors, all of which you can set up in advance by editing the realm configuration files on the Steel-Belted Radius Carrier server: proxy.ini, radius.ini, filter.ini, and one `RealmName`.pro file per realm.

After the target is selected, Steel-Belted Radius Carrier matches the target name with a proxy entry in its database. Using the data in this entry (IP address, UDP port, shared secret) Steel-Belted Radius Carrier establishes a connection between itself and the target, and proxy-forwards the RADIUS request. You can configure the realm so that all realm routing information and delimiters are stripped from the Username before forwarding.

The target processes the request as it normally would for RADIUS authentication or accounting. In the case of authentication, Steel-Belted Radius Carrier waits for a response from the target, then relays this response to its RADIUS client.

Unit 7

General Samba Configuration

Samba is a suite of utilities that allows your Linux box to share files and other resources, such as printers, with Windows boxes. This chapter describes how you can make your Linux box into a Windows Primary Domain Controller (PDC) or a server for a Windows Workgroup. Either configuration will allow everyone at home to have:

- their own logins on all the home windows boxes while having their files on the Linux box appear to be located on a new Windows drive
- shared access to printers on the Linux box
- shared files accessible only to members of their Linux user group.

What's the difference between a PDC and Windows Workgroup member? A detailed description is beyond the scope of this chapter, but this simple explanation should be enough:

- A PDC stores the login information in a central database on its hard drive. This allows each user to have a universal username and password when logging in from all PCs on the network.
- In a Windows Workgroup, each PC stores the usernames and passwords locally so that they are unique for each PC.

This chapter will only cover the much more popular PDC methodology used at home. By default, Samba mimics a Windows PDC in almost every way needed for simple file sharing. Linux functionality doesn't disappear when you do this. Samba Domains and Linux share the same usernames so you can log into the Samba based Windows domain using your Linux password and immediately gain access to files in your Linux user's home directory. For added security you can make your Samba and Linux passwords different.

When it starts up, and with every client request, the Samba daemon reads the configuration file /etc/samba/smb.conf to determine its various modes of operation. You can create your own smb.conf using a text editor or the Web-based SWAT utility which is easier. Keep in mind, however, that if you create /etc/samba/smb.conf with a text editor then subsequently use SWAT to edit the file, you will lose all the comments you inserted with the text editor. I'll explain how to use both SWAT and a text editor to configure Samba later in this chapter.

Samba is comprised of a suite of RPMs that come on the Fedora CDs. The files are named:

- samba
- samba-common
- samba-client
- samba-swat

When searching for the file, remember that the RPM's filename usually starts with the RPM name followed by a version number as in samba-client-3.0.0-15.i386.

Managing the Samba Daemon

Managing the samba daemons is easy to do, but the procedure differs between Linux distributions. Here are some things to keep in mind.

- Firstly, different Linux distributions use different daemon management systems. Each system has its own set of commands to do similar operations. The most commonly used daemon management systems are SysV and Systemd.
- Secondly, the daemon name needs to be known. In this case the names of the daemons are **smb** and **nmb**.

Armed with this information you can know how to:

- Start your daemons automatically on booting
- Stop, start and restart them later on during troubleshooting or when a configuration file change needs to be applied.

The Samba Configuration File

The /etc/samba/smb.conf file is the main configuration file you'll need to edit. It is split into five major sections, which Table 10-1 outlines:

Table 10-1 : File Format - smb.conf

Section	Description
[global]	General Samba configuration parameters
[printers]	Used for configuring printersUsed for configuring printers
[homes]	Defines treatment of user logins
[netlogon]	A share for storing logon scripts. (Not created by default.)
[profile]	A share for storing domain logon information such as "favorites" and desktop icons.

	(Not created by default.)

You can edit this file by hand, or more simply through Samba's SWAT web interface.

SAMBA SWAT

You must always remember that SWAT edits the smb.conf file but also strips out any comments you may have manually entered into it beforehand. The original Samba smb.conf file has many worthwhile comments in it, you should save a copy as a reference before proceeding with SWAT. For example, you could save the original file with the name /etc/samba/smb.conf.original as in:

```
[root@bigboy tmp]# cp /etc/samba/smb.conf /etc/samba/smb.conf.original
```

As you can see, using SWAT requires some understanding of the smb.conf file parameters because it eliminates these comments

SWAT doesn't encrypt your login password. Because this could be a security concern in a corporate environment you might want to create a Samba administrator user that has no root privileges or only enable SWAT access from the GUI console or localhost.

The enabling and disabling, starting and stopping of SWAT is controlled by xinetd, which is covered ,via a configuration file named /etc/xinetd.d/swat. Here is a sample:

```
service swat
{

port            = 901
   socket_type     = stream
protocol        = tcp
wait            = no
user            = root
server          = /usr/sbin/swat
   log_on_failure  += USERID
disable         = no
   only_from       = localhost

}
```

The file's formatting is fairly easy to understand, especially as there are only two entries of interest.

- The disable parameter must be set to no to accept connections. This can automatically be switched between yes and no as we will see later.

- The default configuration only allows SWAT web access from the VGA console only as user root on port 901 with the Linux root password. This means you'll have to enter "http://127.0.0.1:901" in your browser to get the login screen.

You can make SWAT accessible from other servers by adding IP address entries to the only_from parameter of the SWAT configuration file. Here's an example of an entry to allow connections only from 192.168.1.3 and localhost. Notice that there are no commas between the entries.

```
only_from = localhost 192.168.1.3
```

Therefore in this case you can also configure Samba on your Linux server bigboy IP with address 192.168.1.100 from PC 192.168.1.3 using the URL http://192.168.1.100:901.

Remember that most firewalls don't allow TCP port 901 through their filters. You may have to adjust your rules for this traffic to pass.

Network File System

Network File System (**NFS**) is a distributed file system protocol originally developed by Sun Microsystems in 1984,[1] allowing a user on a client computer to access files over a network in a manner similar to how local storage is accessed. NFS, like many other protocols, builds on the Open Network Computing Remote Procedure Call (ONC RPC) system. The Network File System is an open standard defined in RFCs, allowing anyone to implement the protocol.

NFS is often used with Unix operating systems (such as Solaris, AIX and HP-UX) and Unix-like operating systems (such as Linux and FreeBSD). It is also available to operating systems such as the classic Mac OS, OpenVMS, Microsoft Windows, Novell NetWare, and IBM AS/400. Alternative remote file access protocols include the Server Message Block (SMB, also known as CIFS), Apple Filing Protocol (AFP), NetWare Core Protocol (NCP), and OS/400 File Server file system (QFileSvr.400). SMB and NetWare Core Protocol (NCP) occur more commonly than NFS on systems running Microsoft Windows; AFP occurs more commonly than NFS in Macintosh systems; and QFileSvr.400 occurs more commonly in AS/400 systems.

Assuming a Unix-style scenario in which one machine (the client) requires access to data stored on another machine (the NFS server):

1. The server implements NFS daemon processes (running by default as `nfsd`) in order to make its data generically available to clients.
2. The server administrator determines what to make available, exporting the names and parameters of directories (typically using the `/etc/exports` configuration file and the `exportfs` command).
3. The server security-administration ensures that it can recognize and approve validated clients.

4. The server network configuration ensures that appropriate clients can negotiate with it through any firewall system.
5. The client machine requests access to exported data, typically by issuing a `mount` command. (The client asks the server (rpcbind) which port the NFS server is using, the client connects to the NFS server (nfsd), nfsd passes the request to mountd)
6. If all goes well, users on the client machine can then view and interact with mounted filesystems on the server within the parameters permitted.

Note that automation of the NFS mounting process may take place — perhaps using `/etc/fstab` and/or automounting facilities.

NFS Client Configuration

The `mount` command mounts NFS shares on the client side. Its format is as follows:

```
# mount -t nfs -o optionshost:/remote/export/local/directory
```

This command uses the following variables:

`options`

A comma-delimited list of mount options; refer to Section 12.5, "Common NFS Mount Options" for details on valid NFS mount options.

`server`

The hostname, IP address, or fully qualified domain name of the server exporting the file system you wish to mount

`/remote/export`

The file system / directory being exported from *`server`*, i.e. the directory you wish to mount

`/local/directory`

The client location where *`/remote/export`* should be mounted

The NFS protocol version used in Red Hat Enterprise Linux 6 is identified by the `mount` options `nfsvers` or `vers`. By default, `mount` will use NFSv4 with `mount -t nfs`. If the server does not support NFSv4, the client will automatically step down to a version supported by the server. If you use the `nfsvers`/ `vers` option to pass a particular version not supported by the server, the mount will fail. The file system type nfs4 is also available for legacy reasons; this is equivalent to running `mount -t nfs -o nfsvers=4 host:/remote/export/local/directory`.

If an NFS share was mounted manually, the share will not be automatically mounted upon reboot. Red Hat Enterprise Linux offers two methods for mounting remote file systems automatically at boot time: the `/etc/fstab` file and the `autofs` service.

Mounting NFS File Systems using `/etc/fstab`

An alternate way to mount an NFS share from another machine is to add a line to the `/etc/fstab` file. The line must state the hostname of the NFS server, the directory on the server being exported, and the directory on the local machine where the NFS share is to be mounted. You must be root to modify the `/etc/fstab` file.

Example 12.1. Syntax example

The general syntax for the line in `/etc/fstab` is as follows:

```
server:/usr/local/pub    /pub   nfs    defaults 0 0
```

The mount point `/pub` must exist on the client machine before this command can be executed. After adding this line to `/etc/fstab` on the client system, use the command `mount /pub`, and the mount point `/pub` is mounted from the server.

The `/etc/fstab` file is referenced by the `netfs` service at boot time, so lines referencing NFS shares have the same effect as manually typing the `mount` command during the boot process.

A valid `/etc/fstab` entry to mount an NFS export should contain the following information:

```
server:/remote/export/local/directory nfs options 0 0
```

The variables *`server`*, *`/remote/export`*, *`/local/directory`*, and *`options`* are the same ones used when manually mounting an NFS share. Refer to Section 12.3, “NFS Client Configuration” for a definition of each variable.

Note

The mount point *`/local/directory`* must exist on the client before `/etc/fstab` is read. Otherwise, the mount will fail.

Configuring CUPS

One of the latest Linux printing solutions is the **Common UNIX Printing System (CUPS)**, it supports the **Internet Printing Protocol (IPP)** and provides a complete platform independent printing solution for most networking environments. The CUPS server is able to be administered remotely using a web browser, which makes it ideal for a ‘headless server’.

Configuring CUPS

One of the major CUPS advantages is that it can be completely controlled remotely using a standard web browser, so really all we need to do is get it configured and then access it remotely. However this guide will provide all the steps necessary to configure from the command line (print drivers need to be updated manually).

[bash]# vi /etc/cups/cupsd.conf

```
ServerName          server.example.com
ServerAdmin         admin@example.com
AccessLog            /var/log/cups/access_log
DataDir             /usr/share/cups
DefaultCharset    utf-8
DefaultLanguage en
ErrorLog            /var/log/cups/error_log
MaxLogSize        10485760
LogLevel           info
Printcap           /etc/printcap
RequestRoot     /var/spool/cups
ServerBin         /usr/lib/cups
ServerRoot       /etc/cups
User                lp
Group              sys
Listen             127.0.0.1:631
Listen            172.31.81.3:631
```

Additional Configuration For Remote Access

To access the resources remotely via a web browser, we need to specify the access controls (deny/allow) which will be applied to each resource. The “/” (root) resource may be provided to all users without any authentication so they may view which printers are available and the status of the queues.

The “admin” resource has been configured for authentication so that all the administration tasks are only accessible to authorised personnel. It is important to note that no encryption has been established for CUPS in our configuration (it is optional), so if you are administering the system via an untrusted network, be aware that your user details may be captured. This should not be a problem for home networks.

```
<Location />
Require valid-user
Order Deny,Allow
Deny From All
Allow From 127.0.0.1
Allow From 192.168.1.0/24
</Location>

<Location /admin>
Require group printer-admins
```

```
Order Deny,Allow
Deny From All
Allow From 127.0.0.1
Allow From 192.168.1.0/24
</Location>
```

Additional Network Configuration On Printing Solution

CUPS is a full networking solution and it is capable of browsing the network and discovering other CUPS servers. Depending on the configuration, this allows full administration of all your network printers so they can be centrally managed from the one location. The default for browsing is on, however for a small home network you would probably prefer to turn this feature off.

```
Browsing Off
BrowseProtocols cups
BrowseOrder Deny,Allow
BrowseAllow from @LOCAL
BrowseAllow from @IF(eth1)
```

After making the necessary changes to the configuration file, we need to make the system aware of the changes by restarting the printing services. The command to do so is

```
[bash]# /etc/init.d/cups restart
```

The CUPS daemon can now be controlled through a standard web browser if you are connecting from either the localhost, or from the internal network. The service is running on port 631 (default) and can be accessed here: http://localhost:631/.

The message log can be read to decode for any changes and any problems in the CUPS configuration

```
[bash]# tail -f /var/log/cups/error_log
```

Refer to : <a href=”../?p=101″ target=”_parent”>http://blog.supportgurukul.com/?p=101</a> for basic printing configuration (CUPS Configuration).

Refer to : <a href=”../?p=111″ target=”_parent”>http://blog.supportgurukul.com/?p=111</a> for Installing Printer Device Driver (Printer Device Driver Configuration).

Setup CUPS (Common UNIX Printing System) Server and Client in Debian

The Common UNIX Printing SystemTM, or CUPS, is the software you use to print from applications like the web browser you are using to read this page. It converts the page descriptions produced by your application (put a paragraph here, draw a line there, and so forth) into something your printer can understand and then sends the information to the printer for printing.

Now, since every printer manufacturer does things differently, printing can be very complicated. CUPS does its best to hide this from you and your application so that you can concentrate on printing and less on how to print. Generally, the only time you need to know anything about your printer is when you use it for the first time, and even then CUPS can often figure things out on its own.

Install CUPS printer server in Debian

```
#apt-get install cupsys cupsys-driver-gutenprint foomatic-db-gutenprint foomatic-filters
fontconfig libtiff4 libfreetype6
```

NOTE:- If your network use DHCP it's a good idea to set up your server to use static IP. I will use as 172.20.22.74 for the server and 172.20.22.150 for administration workstation.

Configure CUPS

Default configuration file located at /etc/cups/cupsd.conf in this file you need to configure the following sections to make it work.

First, check the encryption setting and change

```
# Default authentication type, when authentication is required...
DefaultAuthType Basic
```

to

```
# Default authentication type, when authentication is required...
DefaultAuthType Basic
DefaultEncryption IfRequested
```

Then we need to tell it to listen for the server change

```
# Only listen for connections from the local machine.
Listen localhost:631
Listen /var/run/cups/cups.sock
```

to

```
# Only listen for connections from the local machine.
Listen localhost:631
Listen 172.20.22.74
Listen /var/run/cups/cups.sock
```

We need it to be visible to the entire network

```
# Show shared printers on the local network.
Browsing On
BrowseOrder allow,deny
BrowseAllow @LOCAL
```

what machines that may access the server change

```
# Restrict access to the server...
<Location/>
Order allow,deny
Allow localhost
</Location>
```

to

```
# Restrict access to the server...
<Location/>
Order allow,deny
Allow localhost
Allow 172.20.22.*
</Location>
```

And the same for the configuration files change

```
# Restrict access to configuration files...
<Location /admin/conf>
AuthType Basic
Require user @SYSTEM
Order allow,deny
Allow localhost
</Location>
```

to

```
# Restrict access to configuration files...
<Location /admin/conf>
AuthType Basic
Require user @SYSTEM
Order allow,deny
Allow localhost
Allow 172.20.22.150
</Location>
```

Other configuration i left default one so need to change anything.

Now you need to restart CUPS using the following command

```
#/etc/init.d/cupsys restart
```

Setting up the CUPS clients

The CUPS clients are easy to set up and the config is identical on all machines.You need to install the following packages for client

```
#apt-get install cupsys cupsys-client
```

Configuring CUPS Client

You need to create /etc/cups/client.conf as root

```
#touch /etc/cups/client.conf
```

Now you need to edit the /etc/cups/client.conf file

```
#vi /etc/cups/client.conf
```

Enter the following information the server IP and the encryption requirement

```
# Servername
ServerName 172.20.22.74

# Encryption
Encryption IfRequested
```

Save the file, then restart the client

```
#/etc/init.d/cupsys restart
```

FTP Server

Introduction

The FTP protocol dates back to the early 1970s, and was designed for a relatively simple network, with few computers on it, mostly directly connected, rather than servers within complex private networks (behind routers and firewalls) being accessed from the internet.

It has to a large extent been superseded by Hypertext Transfer Protocol (HTTP) which has expanded its abilities way beyond those originally planned in the late 1980s, including the transport of large binary files.

FTP excels where there are complex hierarchies of directories and files, which a suitable graphical client can show the local and distant machines alongside each other, allowing easy comparison of the two machines. Curiously its most useful application is in conjunction with web servers - it can be configured to quickly and transparently compare file structures on client and server, updating the one or the other with newer or non-existent files.

Which FTP Client ?

There is a bewildering array of clients available: Absolute FTP, Cute FTP, FileZilla, NcFTP, Smart FTP, WS_FTP to name but a few. A search on Google for ***ftp client*** produced 24 million hits !

Most of these use a graphical interface, but if you are a diehard command-line user, the FTP client built into even the latest Windows operating system will perform the task perfectly well.

You can even use a browser using an address like:

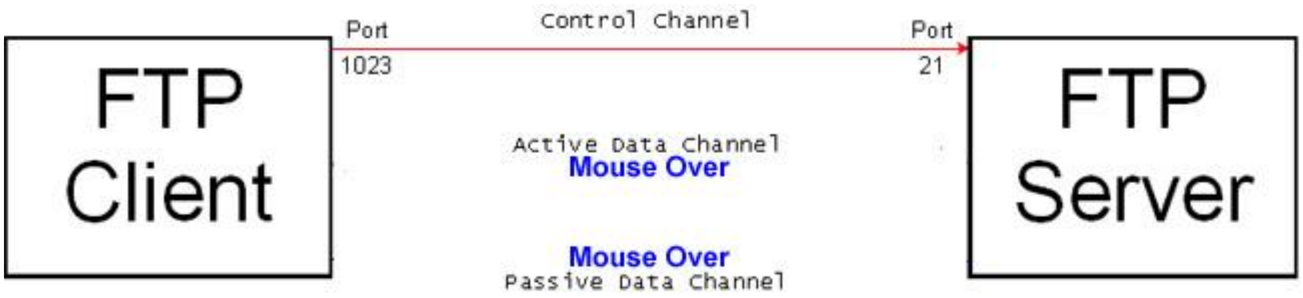
ftp://myftpserver.mydomain.com

In fact, files are often downloaded using FTP by browsers without the user ever knowing - the link for the final download may actually access a FTP server.

How does FTP work ?

The client will initiate a connection with the server, usually via port 21 - this connection does not pass any data, only control messages, hence it is called the ***Control Channel***. This allows the connection to be established, and the login to take place (either anonymous or using a special username and password).

The transport of data (in either direction, and this includes file/directory listings) requires the setting up of a separate data channel (initiated by the FTP client), which may be of two possible types, as shown in the diagram below.



As the image above shows, there are two methods of establishing the data connection, Active or Passive modes, and these are both usually supported by FTP clients. **Mouse over** the diagram above, to show how these work. Active mode is the default: the reason for the different methods will be explained under Problems with Routers & Firewalls below.

Active Data Connection

Via the control channel, the client sends a port number to the server (using the PORT command), then waits for the server to connect to it on this port, from port 20 on the server (the use of port 20 by the server is a convention, like the use of port 21 for the Control Channel).

The data (in either direction) is transported, and the data channel closed.

If more data is to be transported, the client will send ***another*** port number to the server, the server will connect through its port 20 with the specified port on the client, and that data transported. A ***different*** port is used because there is a delay between the instruction to close the original port and its actually closing, which mean it cannot be re-used immediately.

The client will choose port numbers using its own scheme, the details of which are unimportant here as long as ports are not re-used until they will definitely be free.

Passive Data Connection

Via the control channel, the client will issue an instruction to the server (using the PASV command) that the server should open a port to which the client can open a connection. The server will respond with a port number and await the client's initiating a connection to that port. Once established, this will be used for the transport of data, then closed again, just as with the Active method above.

The File Transfer Protocol (FTP) allows the exchange of files between computers. One may `get` files from or `put` files to a remote computer. For obvious security reasons these operations are allowed only after identification of the user. Thus an FTP session may be divided into the following operations:

1. Opening a connection to the remote computer and identification of the user
2. File exchange (`get` or `put` files)
3. Closing the connection

FTP allows the exchange of all types of files be they simple text, word processor documents, graphics, video, audio or executable files. Although it may possible to transfer a file from one machine to another, this does not ensure that the target machine will be able to use the file. For example, the file containing a program in executable form will never be compatible between processors belonging to a different family: an executable file for a Pentium cannot run on a DEC Alpha platform. Hopefully in the future more and more computers will use IEEE formats to store data, consequently easing the task of communication between computers.

FTP uses a client-server approach. The user sends requests from his computer through a ftp client program to a remote computer which receives it through a ftp server program. Thus the communication is asymmetric. Assuming one is on line to machine A and that one wants to exchange files with a remote computer B, it means that:

- Computer A must run a client ftp
- Computer B must run a server ftp (called ftpd on Unix machines).

However with the above programs running, a user on line to computer B, will not be able to exchange files with computer A since there is no client running on machine B and no server on machine A. To allow symmetric communications one needs both client and server programs running on the same machine.

Security considerations

As in general it is not possible to identify users on a microcomputer (PC or MacIntosh) it is advisable not to run a ftp server on such a machine. A client ftp is sufficient to connect the PC to a remote machine and exchange files. Since then no server is running on the PC, remote locations will not be able to connect to the PC. However, some ftp server software does allow the definition of passwords in a configuration file, making access to the PC impossible without its knowledge. Many users are not aware of this feature and thus leave a large hole in the security of their system.

Unix, VMS, OS/2, Windows NT... do not present this problem since a user is always identified.

On Unix machines the user may facilitate the access to an account by suppressing the need for a password from trusted locations and users. This is often used by persons with accounts on different machines to rapidly exchange files between the computers without the need of a full identification. However the system manager may disable this feature and it may not work at some sites.

Anonymous FTP

The need for identification prevents file exchange between computers for users not having a proper account identification and password on the remote computer. This is very frustrating: How can one retrieve a file without asking for a password? How can one send a file to a remote user without this information?

To overcome this difficulty on multiuser machines one may install a special identification called *anonymous*. Anybody can make a FTP connection giving *anonymous* as login identification. Netiquette recommends that ones e-mail address be entered as password but in many cases any string of characters will suffice. *anonymous* access is limited to a restricted set of files. Further it is frequent that the number of simultaneous *anonymous* connexions be limited and one may thus be bounced out with a message saying to try later.

Some servers not only allow to retrieve files but also permit to upload files onto the server, usually in a directory called *incoming* with special properties (*i.e.* the commands `dir` or `ls` do not work). This is very useful for exchanging information with other users. It avoids having to give a personal password to many people with its consequent increase in security risks. To exchange files, one person deposits the file in the *incoming* directory and the end-user fetches it out. Since everybody can fetch the file this mechanism must not be used for confidential information.

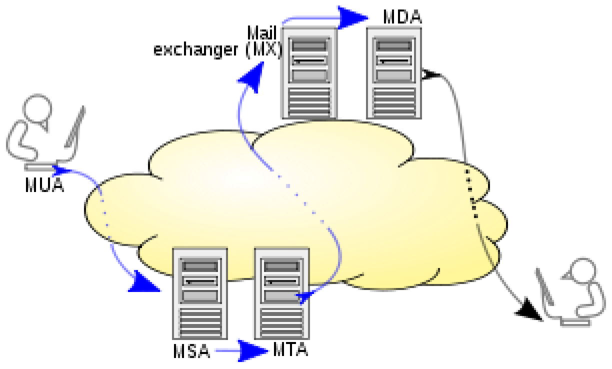
Unit 8 : Mail Server Basics

Simple Mail Transfer Protocol

Simple Mail Transfer Protocol (**SMTP**) is an Internet standard for electronic mail (e-mail) transmission across Internet Protocol (IP) networks. SMTP was first defined by RFC 821 (1982, eventually declared STD 10),[1] and last updated by RFC 5321 (2008)[2] which includes the extended SMTP (ESMTP) additions, and is the protocol in widespread use today. SMTP uses TCPport 25. The protocol for new submissions (MSA) is effectively the same as SMTP, but it uses port 587 instead. SMTP connections secured by SSL are known by the shorthand SMTPS, though SMTPS is not a protocol in its own right.

While electronic mail servers and other mail transfer agents use SMTP to send and receive mail messages, user-level client mail applications typically only use SMTP for sending messages to a mail server for relaying. For receiving messages, client applications usually use either the Post Office Protocol (POP) or the Internet Message Access Protocol (IMAP) or a proprietary system (such as Microsoft Exchange or Lotus Notes/Domino) to access their mail box accounts on a mail server.

Mail processing model



Email is submitted by a mail client (MUA, mail user agent) to a mail server (MSA, mail submission agent) using SMTP on TCP port 587. Most mailbox providers still allow submission on traditional port 25. From there, the MSA delivers the mail to its mail transfer agent (MTA, mail transfer agent). Often, these two agents are just different instances of the same software launched with different options on the same machine. Local processing can be done either on a single machine, or split among various appliances; in the former case, involved processes can share files; in the latter case, SMTP is used to transfer the message internally, with each host configured to use the next appliance as a smart host. Each process is an MTA in its own right; that is, an SMTP server.

The boundary MTA has to locate the target host. It uses the Domain name system (DNS) to look up the mail exchanger record (MX record) for the recipient's domain (the part of the address on the right of @). The returned MX record contains the name of the target host. The MTA next connects to the exchange server as an SMTP client. (The article on MX record discusses many factors in determining which server the sending MTA connects to.)

Once the MX target accepts the incoming message, it hands it to a mail delivery agent (MDA) for local mail delivery. An MDA is able to save messages in the relevant mailbox format. Again, mail reception can be done using many computers or just one —the picture displays two nearby boxes in either case. An MDA may deliver messages directly to storage, or forwardthem over a network using SMTP, or any other means, including the Local Mail Transfer Protocol (LMTP), a derivative of SMTP designed for this purpose.

Once delivered to the local mail server, the mail is stored for batch retrieval by authenticated mail clients (MUAs). Mail is retrieved by end-user applications, called email clients, using Internet Message Access Protocol (IMAP), a protocol that both facilitates access to mail and manages stored mail, or the Post Office Protocol (POP) which typically uses the traditional mbox mail file format or a proprietary system such as Microsoft Exchange/Outlook or Lotus Notes/Domino. Webmail clients may use either method, but the retrieval protocol is often not a formal standard.

SMTP defines message *transport*, not the message *content*. Thus, it defines the mail *envelope* and its parameters, such as the envelope sender, but not the header or the body of the message itself. STD 10 and RFC 5321 define SMTP (the envelope), while STD 11 and RFC 5322 define the message (header and body), formally referred to as the Internet Message Format.

IMAP and POP3

POP3 and IMAP are two different protocols used to access e-mail. POP3 and IMAP function very differently from each other and each has their own advantages. POP3 is useful for checking your e-mail from one computer at a single location. IMAP is the better option when you need to check your e-mail from multiple computers at different locations such as at work, home, or on the road.

POP3

POP3 works by accessing the inbox on the mail server and then downloading the new messages to your computer. With this type of account you do not have to stay logged on to the Internet. You can log on when you want to receive and send new messages. Once your new messages have been downloaded to your computer you can log off to read them. This option is good when you connect to the internet through a dial up service and are charged for your connection time.

IMAP

IMAP lets you view the headers of the new messages on the server and then retrieves the message you want to read when you click on them. When using IMAP, the mail is stored on the mail server. Unless you copy a message to a "Local Folder" the messages are never copied to your PC. Using this protocol, all your mail stays on the server in multiple folders, some of which you have created. This enables you to connect to any computer and see all your mail and mail folders. In general, IMAP is great if you have a dedicated connection to the internet or you like to check your mail from various locations.
It is important to point out that POP3 can be set to leave your e-mail on the server instead of downloading to your computer. This feature enables you to check your e-mail from multiple locations just like IMAP. So why choose IMAP rather than POP3 with the leave mail on server setting? With the POP3 leave mail on server setting only your e-mail messages are on the server, but with IMAP your e-mail folders are also on the server.
So in a nutshell the scenarios for using POP3 and IMAP can be summarized as follows…

When should I use POP3?

-When you only check e-mail from one computer at a single location.
-You want to remove your e-mail from the mail server.
-You connect to the internet through dial up and are charged for connection time.

When should I use IMAP?

-When you need to check e-mail from multiple computers at multiple locations.
-You use Web mail such as Gmail, Yahoo or Hotmail.
-You need to preserve the different e-mail folders you have created.

Tips for better performance

Keep your Inbox small! This is especially true for POP3 and will speed up your e-mail retrieval. Checking the e-mail is directly dependent on how many e-mail messages are in your inbox on the mail server.

POP3

- Try to reduce your inbox size. POP'ing large mail boxes consume excessive server resources.
- Set to remove mail from server at the end of every month or even sooner.
- Do not check your email more frequently than every 15 minutes.

IMAP

- Do NOT check all folders for new messages! This slows your e-mail substantially.
- Organize your mail into folders, and archive your older messages. This speeds up e-mail retrieval by minimizing the number of messages in the inbox.

Comparison of POP and IMAP

The following table lists some common e-mail tasks and compares how they are carried out with the POP protocol to how they're carried out with the IMAP protocol:

	POP	**IMAP**
What does it stand for?	Post Office Protocol	Internet Message Access Protocol
Which protocol would suit me best?	If you access mail using only one computer e.g. your office PC or a laptop.	If you want to access your mail from multiple computers or locations.
Which mail programs can I use?	All mail programs or clients have POP capability	Most mail programs have IMAP capability and you will also be able to access your mail via a web page using any web browser.
Some Common Tasks:		
Check for incoming mail	By default, incoming messages are transferred to your local machine when you check your incoming mail. Only new messages are available if you connect to the server using a PC other than your normal one. You are connected to the server only for the transfer of messages.	By default, incoming messages stay on the server when you check your mail - only headers are transferred with full messages only downloaded when selected for reading. All your messages are always available no matter where or how you connect to the server. You remain connected to the server whilst you deal with mail but some clients allow for off-line working.
Read and respond to mail	Reading and responding to messages is done on your local machine.	You can read and respond to messages directly on the server but you can also read and respond to messages on your local machine, after downloading for offline working (depending on client). When you reconnect, your mailboxes are

		resynchronised to reflect the changes you have made.
Create mailboxes for storing messages	Creating mailboxes can be done only on your local machine.	You can create mailboxes directly on the server. By default, an Inbox is automatically created on the server when you begin using IMAP. The Inbox functions as the master mailbox (or folder) as well as the mailbox for incoming messages. All other mailboxes, including a trash box, need to be created within the Inbox.
Move messages in and out of mailboxes	You can move messages in and out of mailboxes only on your local machine.	You can move messages in and out of mailboxes on the server and on your local machine.
Transfer messages from local machine to server and vice versa	You cannot transfer any messages from your local machine to the server. Messages are automatically transferred from the server to your local machine when you check your incoming mail.	You can transfer individual messages from mailboxes on your local machine into mailboxes on the server and vice versa.
Delete selected messages on the server	When using some clients (e.g. Eudora), if you specified to leave messages on the server, you can delete individual messages left there.	You can delete individual messages and groups of messages directly on the server as well as on your local machine.

SMTP relay

Many administrators misunderstand the concept of SMTP relay. Some over-cautious administrators block SMTP relay completely and others leave it open for any Internet user to misuse their servers. Problems exist in both extremes. Therefore, it is important to understand exactly what SMTP relay is and how to configure your SMTP server so that is does not leave you vulnerable to outside attacks and allows legitimate users to send and receive emails. This article should clear up some of the confusion on this topic and show how to effectively turn off an open relay.

SMTP

Before we dive into SMTP Relay, it is important to know how the SMTP protocol works. SMTP is an acronym for Simple Mail Transfer Protocol. Most of Internet service providers nowadays use this protocol to send email. Email clients, also known as Mail User Agents (MUA), utilize this protocol and act as an SMTP client to distribute email messages to the recipients. When a MUA sends an email messages, it connects to the configured SMTP server and communicates to

it using the SMTP protocol.
Internet mail works pretty much like our postal mail. When you wish to send a letter or a package via snail mail, you put the letter inside an envelope, write the recipient's as well as your return address and drop it off at your local post office. The local post office figures out the final destination of the package and sends it to the appropriate post office in the recipient's town. One important factor to notice here is that if both sender and recipient are in the same town no other post office gets involved.
Electronic mail works pretty much the same. SMTP servers act as local post offices. When a user wishes to send an email, he or she sends it to the SMTP server, which then forwards it to the recipient's SMTP server. Rather than street address and apartment numbers, electronic mail recipients are identified by unique email addresses. Every SMTP server is configured to handle one or more domain names. Analogous to snail mail, if both sender and recipient are in the same domain no other SMTP server gets involved. Following characteristics are common between snail mail and electronic mail.

Snail Mail	**Electronic Mail**
Every mail package is wrapped within an envelope that contains: • Sender's name and address. • Recipient's name and address. • Post office's stamp. • A timestamp when package was received.	Every electronic mail is wrapped within an envelope as well and contains: • Sender's name and email address. • List of recipients and their email address. • SMTP server's signature. There can be more than one SMTP server involved. • The date and time the email was received. • Electronic mail can have more elements than mentioned above.
There is no guarantee that the sender's name and address will always be correct. It is very easy to fake the sender's identity.	Similarly, it is very easy to hide the sender's true identity in an electronic mail.
If the sender and receiver are in the same town, your local post office will not send the package to any other post office.	If the sender and receiver are handled by the same SMTP server, no other server will get involved.
Although the sender's identity cannot be trusted, you can still find a few things about the package by looking at the envelope such as the town letter was mailed from and time.	Similarly, the SMTP envelope (also known as header) contains information such as sender's IP Address and date/time stamp the mail was sent.
Every post office is assigned a postal code or zip code, which is used to identify it location. It is possible that in one post office may handle multiple zip codes.	These postal/zip codes are known as domain names in SMTP speak. Every SMTP server is configured to handle one or more domains. Domain name is the text that appears after the @ sign in an email address.

Mail relay

In case of snail mail, the local post office is a government agency and there are no restrictions on who can send a package. Consider a scenario where you live in town A and you want to send a package to town B. When one town's post office accepts packages from another town it is said to "Relay" your message.
Similarly, if you work for company A and want to send an email to someone in company B, you connect to your SMTP server which then relays your message to the SMTP server owned by company B. The notion that an SMTP server accepts an email that is destined for a different SMTP server is called relaying.
It would be impossible to send email if every SMTP server in the world stopped relaying

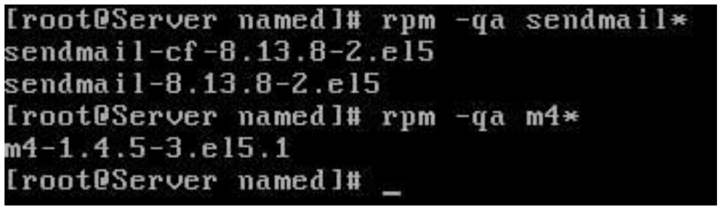
User authentication

The electronic world is a bit different than the real world: you can do things faster, cheaper and distances do not matter. Imagine every time you wanted to send a snail mail you were asked to show your passport or any other document that proved your identity. This would add extra security at a cost of frustration and time. However, the frustration level associated with asking for a user's id and password in an electronic transaction is much lower than the burden of having to carry your passport.
Most SMTP servers ask for the user's credentials in terms of their id and password. The SMTP server will allow users to relay their message to a different server only if these credentials are correct. This authentication mechanism ensures that no one outside the organization can use the company's SMTP server to send message to a third party recipient.

Mail Server Configuration(sendmail)

Configure sendmailserver side

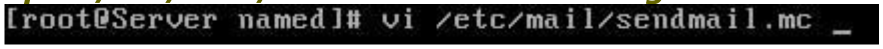
sendmail and m4 rpm are required to configure sendmail server check them for install if not found install them.



```
[root@Server named]# rpm -qa sendmail*
sendmail-cf-8.13.8-2.el5
sendmail-8.13.8-2.el5
[root@Server named]# rpm -qa m4*
m4-1.4.5-3.el5.1
[root@Server named]# _
```

Mail server program reads the **/etc/mail/sendmail.cf**. To change the configuration on mail server, we should edit the **/etc/mail/sendmail.mc** file. When Sendmail is started or restarted with the **service sendmail restart** command a new **sendmail.cf** file is automatically generated if **sendmail.mc** has been modified. In exam you should generate it with **m4** command.

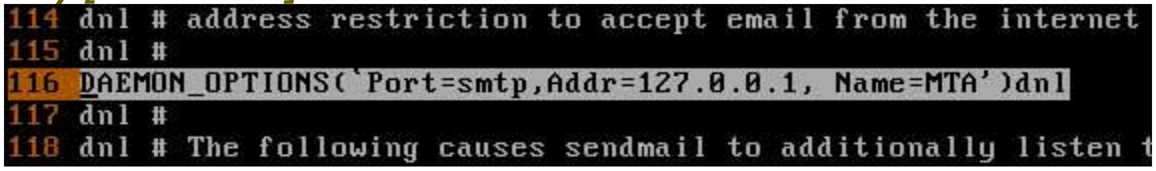
open /etc/mail/sendmail.mc for editing



```
[root@Server named]# vi /etc/mail/sendmail.mc _
```

show hidden line with : set nu option on vi command mode

By default, the following line limits sendmail access to connect local host only [line no 116]

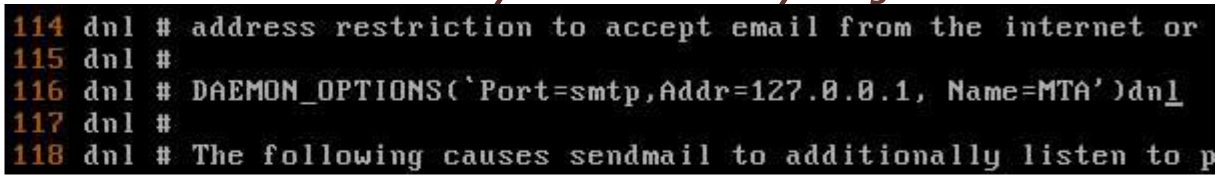


```
114 dnl # address restriction to accept email from the internet
115 dnl #
116 DAEMON_OPTIONS(`Port=smtp,Addr=127.0.0.1, Name=MTA')dnl
117 dnl #
118 dnl # The following causes sendmail to additionally listen t
```

You can allow other computers to use your sendmail server by commenting out this line.

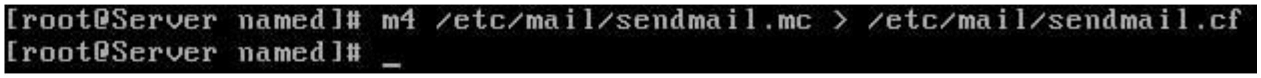
In the **sendmail.mc**file , lines that begin with **dnl**, which stands for delete to new line, are considered comments. Some lines end with dnl, but lines ending in dnl are not comments

comment this line with dnl keyword followed by # sign



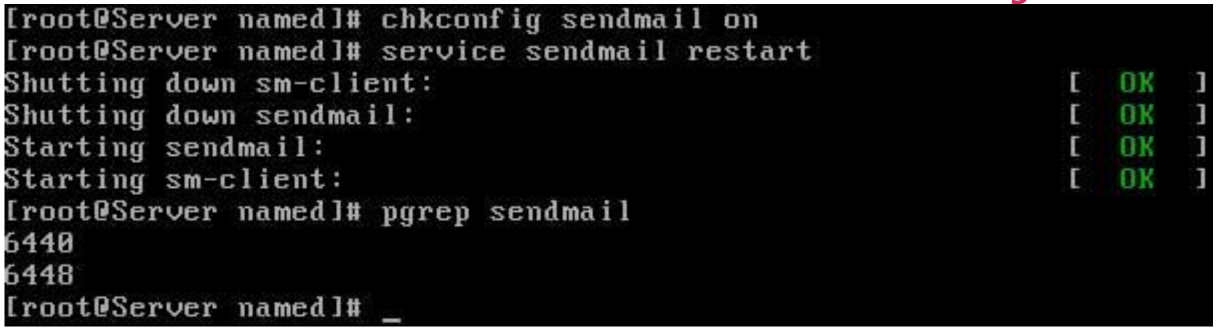
```
114 dnl # address restriction to accept email from the internet or
115 dnl #
116 dnl # DAEMON_OPTIONS(`Port=smtp,Addr=127.0.0.1, Name=MTA')dnl
117 dnl #
118 dnl # The following causes sendmail to additionally listen to p
```

save this file with :wq and exit.
Now generate new sendmail.cf file by using m4 command as shown here



```
[root@Server named]# m4 /etc/mail/sendmail.mc > /etc/mail/sendmail.cf
[root@Server named]# _
```

Now restart sendmail service and also set it on with chkconfig



```
[root@Server named]# chkconfig sendmail on
[root@Server named]# service sendmail restart
Shutting down sm-client:                                   [  OK  ]
Shutting down sendmail:                                    [  OK  ]
Starting sendmail:                                         [  OK  ]
Starting sm-client:                                        [  OK  ]
[root@Server named]# pgrep sendmail
6440
6448
[root@Server named]# _
```

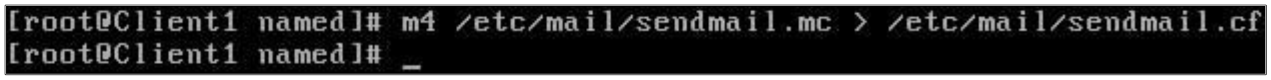
if sendmail service restart without any error means you have configured sendmail successfully.

Configure sendmail client side

We are using another linux system to test **sendmail server**. All configuration are same as you have done on server system.

Check sendmail and m4 rpm for install. Open /etc/mail/sendmail.mc file and locate line no 116 and put a dnl with # sing and save file. All step are same which you have done on server.

Now generate new sendmail.cf file by using m4 command as shown here



```
[root@Client1 named]# m4 /etc/mail/sendmail.mc > /etc/mail/sendmail.cf
[root@Client1 named]# _
```

Now restart sendmail service and also set it on with chkconfig

Controlling e-mail spam

Email spam blocking techniques fall into one of two broad areas. The first area, common in small to midsize sites, is to add spam blocking technology into the Mail Transfer Agent (MTA) - Exchange, Sendmail, Postfix and Communigate are all examples of MTAs. The second technique is more commonly used by large sites and sites with dedicated mail administrators, and that is to put a mail-blocking appliance between the MTA and the Internet. Both techniques have in common some sort of automatic update mechanism so that the MTA or appliance is kept up-to-date against the latest spam sites, patterns and attacks. One of the more popular anti-spam software titles for these purposes is GFI MailEssentials. GFI MailEssentials can be installed directly on the mail server or be installed on a dedicated machine to create a low-cost appliance solution.

Blocking in the MTA has the advantage that no additional hardware is required. Also, the learning curve can be quite short, as the better packages just plug right into the mail server and need relatively little configuration. On the other hand, for the do-it-yourselfers running UNIX and sendmail, the learning curve can be just as long as you like.

Appliances are more commonly used by large and very large sites because they can handle extremely high volumes (millions of messages per day) and can be configured in redundant configurations so that no single failure will disable spam filtering. The appliances are basically high-quality PCs with custom mail software and special configuration front-ends. They are complex devices and generally require expert mail administrators to get the most out of them, although the vendors will configure and maintain them, for a fee. The very largest sites (AOL, Yahoo, Google and MSN Hotmail) essentially build their own custom appliances. Appliances will work in front of any MTA.

Anti-spam techniques

To prevent email spam (aka unsolicited bulk email), both end users and administrators of email systems use various anti-spam techniques. Some of these techniques have been embedded in products, services and software to ease the burden on users and administrators. No one technique is a complete solution to the spam problem, and each has trade-offs between incorrectly rejecting legitimate email vs. not rejecting all spam, and the associated costs in time and effort.

Anti-spam techniques can be broken into four broad categories: those that require actions by individuals, those that can be automated by email administrators, those that can be automated by email senders and those employed by researchers and law enforcement officials.

Security and spamming

The original SMTP specification did not include a facility for authentication of senders. Subsequently, the SMTP-AUTH extension was defined by RFC 2554.[19] The SMTP extension (ESMTP) provides a mechanism for email clients to specify a security mechanism to a mail server, authenticate the exchange, and negotiate a security profile (Simple Authentication and Security Layer, SASL) for subsequent message transfers.

Microsoft products implement the proprietary Secure Password Authentication (SPA) protocol through the use of the SMTP-AUTH extension.

However, the impracticality of widespread SMTP-AUTH implementation and management means that E-mail spamming is not and cannot be addressed by it.

Modifying SMTP extensively, or replacing it completely, is not believed to be practical, due to the network effects of the huge installed base of SMTP. Internet Mail 2000 was one such proposal for replacement.

Spam is enabled by several factors, including vendors implementing MTAs that are not standards-compliant, which make it difficult for other MTAs to enforce standards, security vulnerabilities within the operating system (often exacerbated by always-on broadband connections) that allow spammers to remotely control end-user PCs and cause them to send spam, and a lack of "intelligence" in many MTAs.

There are a number of proposals for sideband protocols that will assist SMTP operation. The Anti-Spam Research Group (ASRG) of the Internet Research Task Force (IRTF) is working on a number of E-mail authentication and other proposals for providing simple source authentication that is flexible, lightweight, and scalable. Recent Internet Engineering Task Force (IETF) activities include MARID (2004) leading to two approved IETF experiments in 2005, and DomainKeys Identified Mail in 2006.

Unit 9: Remote Administration and Management 4hrs

Router Configuration – *RIP/OSPF Router*

Open Shortest Path First

OSPF is an interior gateway protocol that routes Internet Protocol (IP) packets solely within a single routing domain (autonomous system). It gathers link state information from available routers and constructs a topology map of the network. The topology determines the routing table presented to the Internet Layer which makes routing decisions based solely on the destination IP address found in IP packets. OSPF was designed to support variable-length subnet masking (VLSM) or Classless Inter-Domain Routing (CIDR) addressing models.

OSPF detects changes in the topology, such as link failures, very quickly and converges on a new loop-free routing structure within seconds. It computes the shortest path tree for each route using a method based on Dijkstra's algorithm, a shortest path first algorithm.

The **link-state** information is maintained on each router as a link-state database (LSDB) which is a tree-image of the entire network topology. Identical copies of the LSDB are periodically updated through flooding on all OSPF routers.

The OSPF routing policies to construct a route table are governed by link cost factors (*external metrics*) associated with each routing interface. Cost factors may be the distance of a router (round-trip time), network throughput of a link, or link availability and reliability, expressed as simple unitless numbers. This provides a dynamic process of traffic load balancing between routes of equal cost.

An OSPF network may be structured, or subdivided, into routing *areas* to simplify administration and optimize traffic and resource utilization. Areas are identified by 32-bit numbers, expressed either simply in decimal, or often in octet-based dot-decimal notation, familiar from IPv4 address notation.

By convention, area 0 (zero) or `0.0.0.0` represents the core or *backbone* region of an OSPF network. The identifications of other areas may be chosen at will; often, administrators select the IP address of a main router in an area as the area's identification. Each additional area must have a direct or virtual connection to the backbone OSPF area. Such connections are maintained by an interconnecting router, known as *area border router* (ABR). An ABR maintains separate link state databases for each area it serves and maintains summarized routes for all areas in the network.

OSPF does not use a TCP/IP transport protocol (UDP, TCP), but is encapsulated directly in IP datagrams with protocol number 89. This is in contrast to other routing protocols, such as the Routing Information Protocol (RIP), or the Border Gateway Protocol (BGP). OSPF handles its own error detection and correction functions.

OSPF uses multicast addressing for route flooding on a broadcast network link. For non-broadcast networks special provisions for configuration facilitate neighbor discovery. OSPF multicast IP packets never traverse IP routers, they never travel more than one hop. OSPF reserves the multicast addresses`224.0.0.5` for IPv4 or `FF02::5` for IPv6 (all SPF/link state routers, also known as `AllSPFRouters`) and `224.0.0.6` for IPv4 or `FF02::6` for IPv6 (all Designated Routers, `AllDRouters`), as specified in RFC 2328 and RFC 5340.

For routing multicast IP traffic, OSPF supports the Multicast Open Shortest Path First protocol (MOSPF) as defined in RFC 1584. Neither Cisco nor Juniper Networks include MOSPF in their OSPF implementations. PIM (Protocol Independent Multicast) in conjunction with OSPF or other IGPs, (Interior Gateway Protocol), is widely deployed.

The OSPF protocol, when running on IPv4, can operate securely between routers, optionally using a variety of authentication methods to allow only trusted routers to participate in routing. OSPFv3, running on IPv6, no longer supports protocol-internal authentication. Instead, it relies on IPv6 protocol security (IPsec).

OSPF version 3 introduces modifications to the IPv4 implementation of the protocol. Except for virtual links, all neighbor exchanges use IPv6 link-local addressing exclusively. The IPv6 protocol runs per link, rather than based on the subnet. All IP prefix information has been removed from the link-state advertisements and from the *Hello* discovery packet making OSPFv3 essentially protocol-independent. Despite the expanded IP addressing to 128-bits in IPv6, area and router identifications are still based on 32-bit values.

OSPF router types

OSPF defines the following router types:

1. **Area border router (ABR)**

   An area border router (ABR) is a router that connects one or more areas to the main backbone network. It is considered a member of all areas it is connected to. An ABR keeps multiple copies of the link-state database in memory, one for each area to which that router is connected.

2. **Autonomous system boundary router (ASBR)**

   An autonomous system boundary router (ASBR) is a router that is connected to more than one Routing protocol and that exchanges routing information with routers in other protocols. ASBRs typically also run an exterior routing protocol (e.g., BGP), or use static routes, or both. An ASBR is used to distribute routes received from other, external ASs throughout its own autonomous system. (An interactive ASBR simulation shows how an ASBR creates External LSA for external addresses and floods them to all areas via ABR.) Routers in other areas use ABR as next hop to access external addresses. Then ABR forwards packets to the ASBR that announces the external addresses.

3. **Internal router (IR)**

   An internal router is a router that has OSPF neighbor relationships with interfaces in the same area. An internal router has all its interfaces in a single area.

4. **Backbone router (BR)**

   Backbone routers are all routers that are connected to the OSPF backbone, irrespective of whether they are also area border routers or internal routers of the backbone area. An area border router is always a backbone router, since all areas must be either directly connected to the backbone or connected to the backbone via a virtual link (spanning across another area to get to the backbone).

Applications

OSPF was the first widely deployed routing protocol that could converge a network in the low seconds, and guarantee loop-free paths. It has many features that allow the imposition of policies about the propagation of routes that it may be appropriate to keep local, for load sharing, and for selective route importing more than IS-IS. IS-IS, in contrast, can be tuned for lower overhead in a stable network, the sort more common in ISP than enterprise networks. There are some historical accidents that made IS-IS the preferred IGP for ISPs, but ISP's today may well choose to use the features of the now-efficient implementations of OSPF, after first considering the pros and cons of IS-IS in service provider environments.

As mentioned, OSPF can provide better load-sharing on external links than other IGPs. When the default route to an ISP is injected into OSPF from multiple ASBRs as a Type I external route and the same external cost specified, other routers will go to the ASBR with the least path cost from its location. This can be tuned further by adjusting the external cost.

In contrast, if the default route from different ISPs is injected with different external costs, as a Type II external route, the lower-cost default becomes the primary exit and the higher-cost becomes the backup only.

The only real limiting factor that may compel major ISPs to select IS-IS over OSPF is if they have a network with more than 850 routers. There is mention of an OSPF network with over 1000 routers, but that is quite uncommon and the network must be specifically designed to minimize overhead to achieve stable operation.

Routing Information Protocol

The **Routing Information Protocol** (**RIP**) is a distance-vector routing protocol, which employs the hop count as a routing metric. RIP prevents routing loops by implementing a limit on the number of hops allowed in a path from the source to a destination. The maximum number of hops allowed for RIP is 15. This hop limit, however, also limits the size of networks that RIP can support. A hop count of 16 is considered an infinite distance and used to deprecate inaccessible, inoperable, or otherwise undesirable routes in the selection process.

RIP implements the split horizon, route poisoning and holddown mechanisms to prevent incorrect routing information from being propagated. These are some of the stability features of RIP. It is also possible to use the so called RMTI (**R**outing Information Protocol with **M**etric-based **T**opology **I**nvestigation) algorithm to cope with the count-to-infinity problem. With its help, it is possible to detect every possible loop with a very small computation effort.

Originally each RIP router transmitted full updates every 30 seconds. In the early deployments, routing tables were small enough that the traffic was not significant. As networks grew in size, however, it became evident there could be a massive traffic burst every 30 seconds, even if the routers had been initialized at random times. It was thought, as a result of random initialization, the routing updates would spread out in time, but this was not true in practice. Sally Floyd and Van Jacobson showed in 1994 that, without slight randomization of the update timer, the timers synchronized over time. In most current networking environments, RIP is not the preferred choice for routing as its time to converge and scalability are poor compared to EIGRP, OSPF, or IS-IS (the latter two being link-state routing protocols), and (without RMTI) a hop limit severely limits the size of network it can be used in. However, it is easy to configure, because RIP does not require any parameters on a router unlike other protocols (see here for an animation of basic RIP simulation visualizing RIP configuration and exchanging of Request and Response to discover new routes).

RIP uses the User Datagram Protocol (UDP) as its transport protocol, and is assigned the reserved port number 520.

Limitations

- Without using RMTI, Hop count cannot exceed 15, in the case that it exceeds this limitation, it will be considered invalid.
- Most RIP networks are flat. There is no concept of areas or boundaries in RIP networks.
- Variable Length Subnet Masks were not supported by RIP version 1.
- Without using RMTI, RIP has slow convergence and count to infinity problems.

Introduction ToWebmin

Webmin is a web-based interface for system administration for Unix. Using any browser that supports tables and forms (and Java for the File Manager module), you can setup user accounts, Apache, DNS, file sharing and so on.

Webmin consists of a simple web server, and a number of CGI programs which directly update system files like `/etc/inetd.conf` and `/etc/passwd`. The web server and all CGI programs are written in Perl version 5, and use no non-standard Perl modules.

Who developed Webmin?

Almost all the development of Webmin was done by Jamie Cameron, though many people have contributed patches and translations into additional languages. There are also many third-party modules that were developed by other people separately.

What licence is Webmin distributed under?

All recent versions of Webmin are under a BSD-like licence, meaning that it may be freely distributed and modified for commercial and non-commercial use.

Because Webmin supports the concept of modules (like PhotoShop plugins), anyone can develop and distribute their own Webmin modules for any purpose, and distribute them under any licence (such as GPL, commercial or shareware).

What is Usermin?

Usermin is a web-based interface for webmail, password changing, mail filters, fetchmail and much more. It is designed for use by regular non-root users on a Unix system, and limits them to tasks that they would be able to perform if logged in via SSH or at the console.

Who can use Usermin?

Most users of Usermin are sysadmins looking for a simple webmail interface to offer their customers. Unlike most other webmail solutions, it can be used to change passwords, read email with no additional servers installed (like IMAP or POP3), and setup users' Procmail configurations for forwarding, spam filtering and autoreponders.

Usermin also provides web interfaces for viewing and managing data in MySQL and PostgreSQL databases, editing Apache `.htaccess` configuration files, and running commands on the server. The administrator has full control over which of these modules are available to users.

Usermin and Webmin integration

By far the easiest way to configure Usermin is via the **Usermin Configuration** module in Webmin. All functionality can be managed via a browser, and because both products come from the same developer the management user interface is always up to date.

TeamViewer

TeamViewer is a proprietary computer software package for remote control, desktop sharing, online meetings, web conferencing and file transfer between computers. The software operates with the Microsoft Windows, Mac OS X, Linux, iOS, and Androidoperating systems. It is possible to access a machine running TeamViewer with a web browser. While the main focus of the application is remote control of computers, collaboration and presentation features are included.

TeamViewer GmbH was founded in 2005 in Uhingen, Germany. TeamViewer is now owned by GFI.

Establishing connections

TeamViewer may be installed with an installation procedure, although the 'Quick Support' version will run without installation. To connect to another computer, TeamViewer has to be running on both machines. To install TeamVieweradministratoraccess is required, but once installed it can be run by any user. When TeamViewer is started on a computer, it generates a partner ID and password (user-defined passwords are also supported). To establish a connection from a local client to a remote host machine, the local operator must communicate with the remote operator, request the ID and password, then enter these into the local TeamViewer.

To start an online meeting the presenter gives the Meeting ID to their participants. They join the meeting by using the TeamViewer full version or by logging on to http://go.teamviewer.com/ and enter the Meeting ID.

It is also possible to schedule a meeting in advance.

Telnet

Telnet is a program that allows users to log into your server and get a command prompt just as if they were logged into the VGA console. The Telnet server RPM is installed and disabled by default on Fedora Linux.

One of the disadvantages of Telnet is that the data is sent as clear text. This means that it is possible for someone to use a network analyzer to peek into your data packets and see your username and password. A more secure method for remote logins would be via Secure Shell (SSH) which uses varying degrees of encryption.

In spite of this, the older Telnet application remains popular. Many network devices don't have SSH clients, making telnet the only means of accessing other devices and servers from them.

The command to do remote logins via telnet from the command line is simple. You enter the word telnet and then the IP address or server name to which you want to connect.

Installing Telnet Server Software

Older versions of Red Hat had the Telnet server installed by default. Fedora Linux doesn't do this and you will have to install it yourself.

Most Linux software products are available in a precompiled package format. Downloading and installing packages isn't hard.

When searching for the file, remember that the Telnet server RPM's filename usually starts with the word "`telnet-server`" followed by a version number as in `telnet-server-0.17-28.i386.rpm`.

Setting Up a Telnet Server

Setting up the telnet server is easy to do, but the procedure differs between Linux distributions.

Redhat / Fedora

To set up a Telnet server use the `chkconfig` command to activate Telnet.

```
[root@bigboytmp]# chkconfig telnet on
```

You can also use the `chkconfig --list` command to verify that telnet will be started on the next reboot.

```
[root@bigboytmp]# chkconfig --list | grep telnet
telnet: on
[root@bigboytmp]#
```

Use the chkconfig command to deactivate telnet, even after the next reboot.

```
[root@bigboytmp]# chkconfig telnet off
```

SSH

SSH is some kind of an abbreviation of Secure SHell. It is a protocol that allows secure connections between computers. In this tutorial, we'll be dealing with the ssh command on Linux, the OpenSSH version. Most Linux distributions feature the OpenSSH client today, but if you want to be sure, have a look at the SSH manpage on your system. You can do this by typing:

```
[pinehead@localhost ~]$ man ssh
```

Note: this should be done in a terminal. This tutorial assumes that you have some basic terminal knowledge, like knowing how to start a terminal session on your system and being familiar with the basic commands and syntaxes.

If it displays something like this

```
NAME
ssh – OpenSSH SSH client (remote login program)
```

then you can be quite sure you're running the OpenSSH version.

The most simple case

In the most simple case, you can connect to a server that supports ssh with a syntax as short as this:

```
[pineehad@localhost ~]$ sshyourserver
```

Note: If you do not have any ssh server nearby that you can access, you can also try this command with your own computer as a server. To do this, replace "yourserver" with "localhost".

Of course, yourserver should be replaced by a hostname or an ip address of the server you want to connect to. As you can see in the terminal snippet, I am logged in as pineehad. If you do not specify a username (I'll explain how to do that later in this tutorial), SSH will assume that you want to login with the username you're currently logged in with. So, in this case, SSH will try the username pineehad.

Of course, you need to be sure that the server supports ssh connections. The ssh client tries to connect to port 22 defaultly. This means that, if you want to connect to a remote host with the default settings, you should make sure that, if applicable, port 22 is forwarded to the server you're trying to connect to. You will find more regarding the SSH port further in this tutorial.

Now, back to the command we ran. If the server supports SSH connections and you can reach it by port 22, you should be prompted for a password (if this is the first time you try to connect to the server, ssh will first ask the question if you want to continue connecting, which can generally just be answered with a 'yes'). If you type a password here, you won't see asterisks appearing. Don't panic, this is ssh's normal behaviour. It makes connecting using ssh even more safe, because any accidental spectators won't be able to see the length of the password. After entering the password, if the username and the password were correct, you should be running a shell on the server. If not, make sure you are connecting to a server of which you know that you should be able to login with your username and the specified password. You could try connecting to your own computer (see the note beneath the terminal quote) or read on to learn how to specify another username.

Once you're done trying the ssh shell, you can exit it by pressing Ctrl + D.

Specifying a username

It's actually quite simple to specify a different username. You might even already be familiar with it. See the following example:

```
[pinehead@localhost ~]$ sshyourusername@yourserver
```

The above will make ssh try to connect with the username "yourusername" instead of (in my case) pineehad. This syntax is also used by a lot of other protocols, so it'll always come in handy to know it. By the way, you will still be asked for a password. For security reasons, it is not even possible to directly specify the password in the syntax. You will always be asked interactively, unless you start configuring the server in an advanced way (which is exactly why that topic is out of this tutorials scope: this tutorial documents how to use the clients, not how to configure the server).

Specifying a port

There are many reasons to move the ssh service to another port. One of them is avoiding brute-force login attempts. Certain hackers try to get access to ssh servers by trying a lot of common usernames with common passwords (think of a user "john" with password "doe"). Although it is very unlikely that these hackers will ever get access to the system, there is another aspect of the brute-force attacks that you'll generally want to avoid: the system and connection load. The brute-force attacks usually are done with dozens or even thousands of tries a second, and this unnecessarily slows down the server and takes some bandwidth which could've been used a lot better. By changing the port to a non-default one, the scripts of the hackers will just be refused and most of the bandwidth will be saved.

As the ssh command can't just guess the port, we will have to specify it if it's not the default 22 one. You can do that this way:

```
[pineehad@localhost ~]$ ssh -p yourportyourusername@yourserver
```

Of course, you will have to replace "yourport" with the port number. These is an important difference between ssh and scp on this point. I'll explain it further on.

Running a command on the remote server

Sometimes, especially in scripts, you'll want to connect to the remote server, run a single command and then exit again. The ssh command has a nice feature for this. You can just specify the command after the options, username and hostname. Have a look at this:

```
[pineehad@localhost ~]$ sshyourusername@yourserverupdatedb
```

This will make the server update its searching database. Of course, this is a very simple command without arguments. What if you'd want to tell someone about the latest news you read on the web? You might think that the following will give him/her that message:

```
[pineehad@localhost ~]$ sshyourusername@yourserver wall "Hey, I just found out something
great! Have a look at www.examplenewslink.com!"
```

However, bash will give an error if you run this command:

```
bash: !": event not found
```

What happened? Bash (the program behind your shell) tried to interpret the command you wanted to give ssh. This fails because there are exclamation marks in the command, which bash will interpret as special characters that should initiate a bash function. But we don't want this, we just want bash to give the command to ssh! Well, there's a very simple way to tell bash not to worry about the contents of the command but just pass it on to ssh already: wrapping it in single quotes. Have a look at this:

```
[pineehad@localhost ~]$ sshyourusername@yourserver 'wall "Hey, I just found out something
great! Have a look at www.examplenewslink.com!"'
```

The single quotes prevent bash from trying to interpret the command, so ssh receives it unmodified and can send it to the server as it should. Don't forget that the single quotes should be around the whole command, not anywhere else.

SCP

The scp command allows you to copy files over ssh connections. This is pretty useful if you want to transport files between computers, for example to backup something. The scp command uses the ssh command and they are very much alike. However, there are some important differences.

The scp command can be used in three* ways: to copy from a (remote) server to your computer, to copy from your computer to a (remote) server, and to copy from a (remote) server to another (remote) server. In the third case, the data is transferred directly between the servers; your own computer will only tell the servers what to do. These options are very useful for a lot of things that require files to be transferred, so let's have a look at the syntax of this command:

```
[pineehad@localhost ~]$ scpexamplefileyourusername@yourserver:/home/yourusername/
```

Looks quite familiar, right? But there are differences. The command above will transfer the file "examplefile" to the directory "/home/yourusername/" at the server "yourserver", trying to get sshacces with the username "yourusername". That's quite a lot information, but scp really needs it all. Well, almost all of it. You could leave out the "yourusername@" in front of "yourserver", but only if you want to login on the server with your current username on your own computer. Let's have a closer look at the end of the command. There's a colon over there, with a directory after it. Just like Linux's normal cp command, scp will need to know both the source file(s) and the target directory (or file). For remote hosts, the file(s)/directory are given to the scp command is this way.

You can also copy a file (or multiple files) from the (remote) server to your own computer. Let's have a look at an example of that:

```
[pineehad@localhost ~]$ scpyourusername@yourserver:/home/yourusername/examplefile .
```

Note: The dot at the end means the current local directory. This is a handy trick that can be used about everywhere in Linux. Besides a single dot, you can also type a double dot (..), which is the parent directory of the current directory.

This will copy the file "/home/yourusername/examplefile" to the current directory on your own computer, provided that the username and password are correct and that the file actually exists.

You probably already guessed that the following command copies a file from a (remote) server to another (remote) server:

```
[pineehad@localhost ~]$ scpyourusername@yourserver:/home/yourusername/examplefile
yourusername2@yourserver2:/home/yourusername2/
```

Please note that, to make the above command work, the servers *must* be able to reach each other, as the data will be transferred directly between them. If the servers somehow can't reach each other (for example, if port 22 is not open on one of the sides) you won't be able to copy anything. In that case, copy the files to your own computer first, then to the other host. Or make the servers able to reach each other (for example by opening the port).

Well, those are the main uses of scp. We'll now go a bit more in-depth about the differences between ssh and scp.

*: *Actually you can also use it just like the normal cp command, withhout any ssh connections in it, but that's quite useless. It requires you to type an extra 's' =).*

Specifying a port with scp

The scp command acts a little different when it comes to ports. You'd expect that specifying a port should be done this way:

```
[pineehad@localhost ~]$ scp -p
yourportyourusername@yourserver:/home/yourusername/examplefile .
```

However, that will not work. You will get an error message like this one:

```
cp: cannot stat `yourport': No such file or directory
```

This is caused by the different architecture of scp. It aims to resemble cp, and cp also features the -p option. However, in cp terms it means 'preserve', and it causes the cp command to preserve things like ownership, permissions and creation dates. The scp command can also preserve things like that, and the -p option enables this feature. The port specification should be done with the -P option. Therefore, the following command will work:

```
[pineehad@localhost ~]$ scp -P
yourportyourusername@yourserver:/home/yourusername/examplefile .
```

Also note that the -P option *must* be in front of the (remote) server. The ssh command will still work if you put -p yourport behind the host syntax, but scp won't. Why? Because scp also supports copying between two servers and therefore needs to know *which* server the -P option applies to.

Another difference between scp and ssh

Unlike ssh, scp cannot be used to run a command on a (remote) server, as it already uses that feature of ssh to start the scp server on the host. The scp command does have an option that accepts a program (the -S option), but this program will then be used instead of ssh to establish the encrypted connection, and it will not be executed on the remote host.

rsync

rsync is a software application and network protocol for Unix-like systems with ports to Windows that synchronizes files and directories from one location to another while minimizing data transfer by using delta encoding when appropriate. Quoting the official website: "rsync is a file transfer program for Unixsystems.rsync uses the 'rsync algorithm' which provides a very fast method for bringing remote files into sync." An important feature of rsync not found in most similar programs/protocols is that the mirroring takes place with only one transmission in each direction. rsync can copy or display directory contents and copy files, optionally using compression and recursion.

In daemon mode, rsync listens on the default TCPport of 873, serving files in the native rsync protocol or via a remote shell such as RSH or SSH. In the latter case, the rsync client executable must be installed on the remote machine as well as on the local machine.

rsync was originally written as a replacement for **rcp** and **scp**. As such, it has a similar syntax to its parent programs. Like its predecessors, it still requires a source and a destination to be specified, either of which may be remote. Because of the flexibility, speed and scriptability of rsync, it has become a standard Linux utility and is included in all popular Linux distributions. As a result, rsync has been ported to Windows (via commercial Cygwin, freeware Grsync or SFU[11]) and Mac OS.

Possible uses:

```
rsync [OPTION] … SRC [SRC] … [USER@]HOST:DEST
rsync [OPTION] … [USER@]HOST:SRC [DEST]
```

One of the earliest applications of rsync was to implement mirroring or backup for multiple Unix clients to a central Unix server using rsync/ssh and standard Unix accounts.

With a scheduling utility such as cron, one can schedule automated encrypted rsync-based mirroring between multiple hosts and a central server.

Examples

A command line to mirror FreeBSD might look like:

```
% rsync -avz --delete ftp4.de.FreeBSD.org::FreeBSD/ /pub/FreeBSD/
```

The Apache HTTP Server supports only rsync for updating mirrors.

```
rsync -avz --delete --safe-links rsync.apache.org::apache-dist
/path/to/mirror
```

The preferred (and simplest) way to mirror the PuTTY website to the current directory is to use rsync.

```
rsync -auH rsync://rsync.chiark.greenend.org.uk/ftp/users/sgtatham/putty-
website-mirror/ .
```

A way to mimic the capabilities of Time Machine (Mac OS).

```
#date=`date "+%Y-%m-%dT%H:%M:%S"`
date=`date "+%FT%T"`
rsync -aP --link-dest=$HOME/Backups/current /path/to/important_files
$HOME/Backups/back-$date
rm -f $HOME/Backups/current
ln -s $HOME/Backups/back-$date $HOME/Backups/current
```