Application level services

1. High level services
2. Proxies and agents
3. Installing a new service
4. Summoning daemons
5. Setting up the DNS nameservice
6. Setting up a WWW server
    1. Choosing a server host
    2. Installation
    3. Configuration

7. Email configuration
8. OpenLDAP directory service
9. Mounting NFS disks
10. Samba
11. Printer service
12. Java web and enterprise services

High Level Services

FTP.
• HTTP.
• S-HTTP is a superset of HTTP, which allows messages to be encapsulated for increased security. Encapsulations include encryption, signing and MAC-based authentication. An S-HTTP message can have several security transformations applied to it. S-HTTP also It is generally regarded as being superior to HTTPS, a pure SSL encryption.
• HTTPS.
• SSH. The secure shell. A replacement for the remote shell (rsh) Unix protocol. The secure shell provides full encryption and forwarding of X11 display data through a secure pipe.
• LDAP
• NTP is the Network Time Protocol, used for synchronizing clocks throughout the network.
• IMAP. *Internet Mail Access Protocol provides a modern mailbox format and a number of* network services for reading and transferring mail over the network.
• RPC

Proxies and agents

- Security
- caching

Proxy Servers

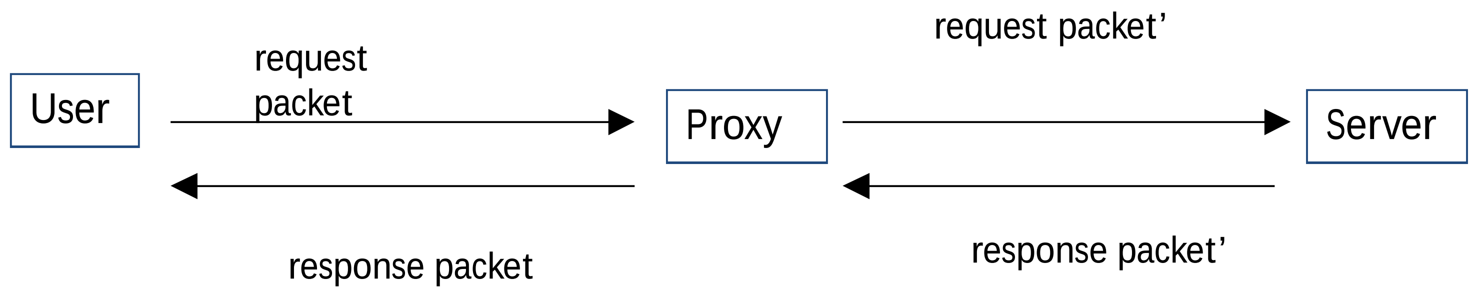
- Part of an overall Firewall strategy
- Sits between the local network and the external network
  - Originally used primarily as a caching strategy to minimize outgoing URL requests and increase perceived browser performance
  - Primary mission is now to insure anonymity of internal users
    - Still used for caching of frequently requested files
    - Also used for content filtering
- Acts as a go-between, submitting your requests to the external network
  - Requests are translated from your IP address to the Proxy's IP address
  - E-mail addresses of internal users are removed from request headers
  - Cause an actual break in the flow of communications

Security Advantages

- Terminates the TCP connection before relaying to target host (in and out)
- Hide internal clients from external network
- Blocking of dangerous URLs
- Filter dangerous content
- Check consistency of retrieved content
- Eliminate need for transport layer routing between networks
- Single point of access, control and logging

TCP Connection Termination

- Both the outgoing and incoming TCP connections are terminated
- prevents a hacker from hijacking a stale connection on a service that is being proxied
- ex . HTTP page request



Connection left open until the proxy closes it after receiving response packet and sending it back to user

Connection only left open until server closes the connection after sending the response packet

TCP Connection Termination

- Transport layer packets don’t need to be routed because the entire request must be regenerated
  - Prevents transport layer exploits
    - source routing
    - fragmentation
    - several DoS attacks
- Since some protocols don’t have proxies available many admins will enable routing , this alleviates any benefit gained
- Most good proxy servers will allow you to create generic proxies using SOCKS or the redir utility

Performance Aspects

- Caching
  - By keeping local copies of frequently accessed file the proxy can serve those files back to a requesting browser without going to the external site each time, this dramatically improves the performance seen by the end user
  - Only makes sense to implement this at the ISP rather than the small business level because of the number of pages available
  - Because of dynamic content many pages are invalidated in the cache right away
- Load balancing
  - A proxy can be used in a reverse direction to balance the load amongst a set of identical servers (servers inside the firewall and users outside)
  - Used especially with web dynamic content (.asp, .php,.cfm,.jsp)

Proxy Liabilities

- Single point of failure
  - if the proxy dies , no one can get to the external network
- Client software must usually be designed to use a proxy
- Proxies must exist for each service
- Doesn’t protect the OS
  - proxies run at the application level
- Usually optimized for performance rather than security
  - WINGATE was installed to be easy to configure; opened a winsock proxy to the external interface, which let hackers essentially hijack the machine
- Create a service bottleneck
  - solved via parallelism (more proxies, and load balance)

Transparent / Opaque

- Transparent – both parties (local/remote) are unaware that the connection is being proxied
  - Zorp - application layer proxy is transparent
- Opaque – the local party must configure client software to use the proxy
  - client software must be proxy-aware software
  - Netscape proxy server is opaque
- With all of the things modern firewalls can do in the area of redirection you could configure the firewall to redirect all http requests to a proxy
  - no user configuration required (transparent)

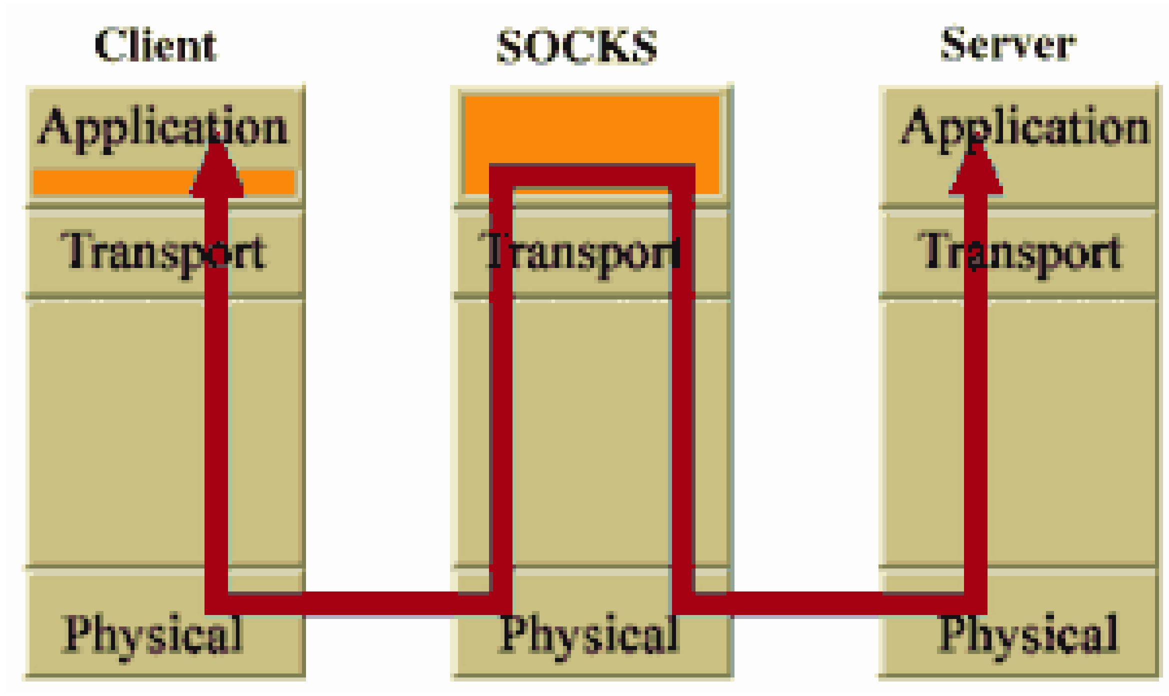
Circuit Level Proxies

- Since some protocols require a real connection between the client and server, a regular proxy can't be used
  - Windows Media Player, Internet Relay Chat (IRC), or Telnet
- Circuit-level proxy servers were devised to simplify matters.
  - Instead of operating at the Application layer, they work as a "shim" between the Application layer and the Transport layer, monitoring TCP handshaking between packets from trusted clients or servers to untrusted hosts, and vice versa. The proxy server is still an intermediary between the two parties, but this time it establishes a virtual circuit between them.
- By using SOCKS (RFC 1928) this can be done
  - SOCKS defines a cross-platform standard for accessing circuit-level proxies
  - SOCKS Version 5 also supports both username/password (RFC 1929) and API-based (RFC 1961) authentication. It also supports both public and private key encryption.
  - SOCKS 5 is capable of solving this problem by establishing TCP connections and then using these to relay UDP data.

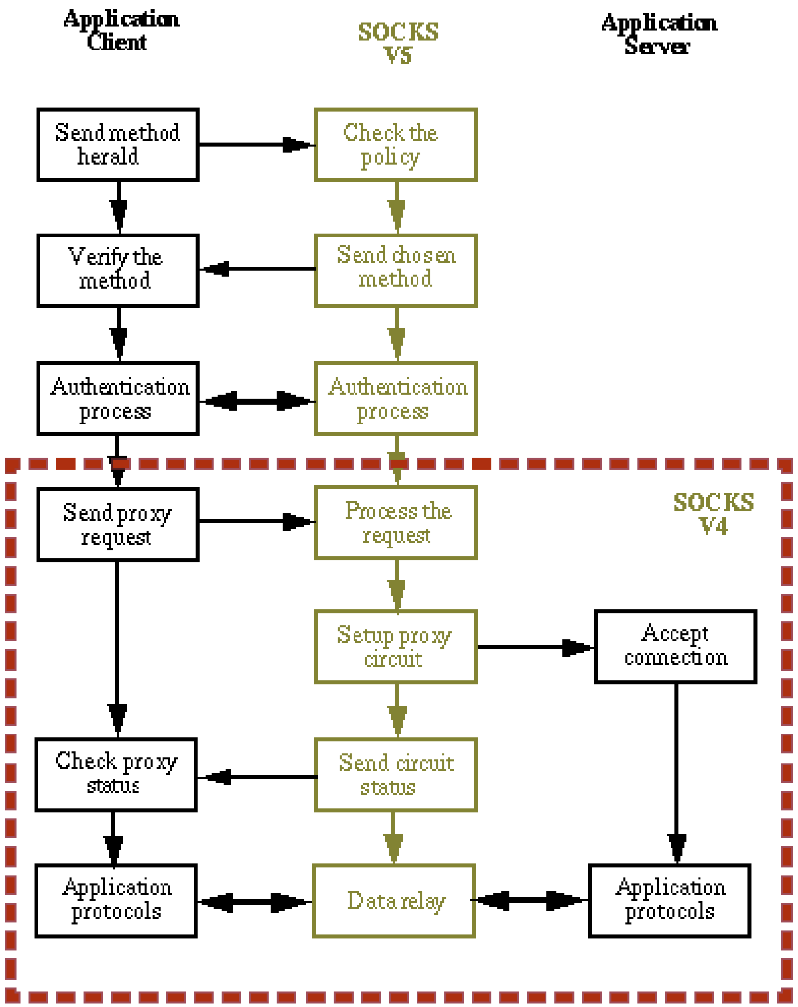
SOCKS based Proxying

- RFC 1928
- Not a true application layer proxy
- SOCKS protocol provides a framework for developing secure communications by easily integrating other security technologies
- SOCKS includes two components
  - SOCKS server
    - implemented at the application layer
  - SOCKS client
    - implemented between the application and transport layers
- The basic purpose of the protocol is to enable hosts on one side of a SOCKS server to gain access to hosts on the other side of a SOCKS Server, without requiring direct IP-reachability.
- Copies packet payloads through the proxy

Socks Architecture



Socks Functionality



GNU ZORP Proxy Firewall Suite

- Protocol Analyzing Firewall
- core framework allows:
  - the administrator to fine tune proxy decisions (Python based)
  - fully analyze complex protocols with an application-level gateway:
    - SSH with several forwarded TCP connections
    - SSL with an embedded POP3 protocol).
    - FTP, TTP, finger, whois, SSL .
- Usually integrated into the network topology as routers, this means that they have an IP address in all their subnets, and hosts on different subnets

  use the firewall as their gateway to the outside world.
- Proxy based but uses a packet filter to preprocess the packet stream and provide transparency.

How Zorp Works

- A TCP session is established in the following way:
- client initiates a connection by sending a SYN packet destined to the server
- the firewall behaves as a router between the client and the server, receives the SYN packet on one of its interfaces and consults the packet filter
- the packet filter rulebase is checked whether the given packet is permitted
- if the given connection is to be processed by a proxy, then the packet filter rulebase contains a REDIRECT (ipchains) or TPROXY (iptables) target. Both REDIRECT and TPROXY requires a port parameter which tells the local port of the firewall host where the proxy is listening.
- Zorp accepts the connection, checks its own access control rules and starts the appropriate proxy
- the proxy connects to the server on its own as needed (the server side connection is not necessarily established immediately)
- the proxy mediates protocol requests and responses between the communicating hosts while analyzing the ongoing stream

Best Practices

- Use a Real Firewall
- Disable Routing
- Secure the Base Operating System
  - harden the OS
- Disable External Access
- Disable unneeded Services

Address Conversion Functions and The Domain Name System

Hostnames

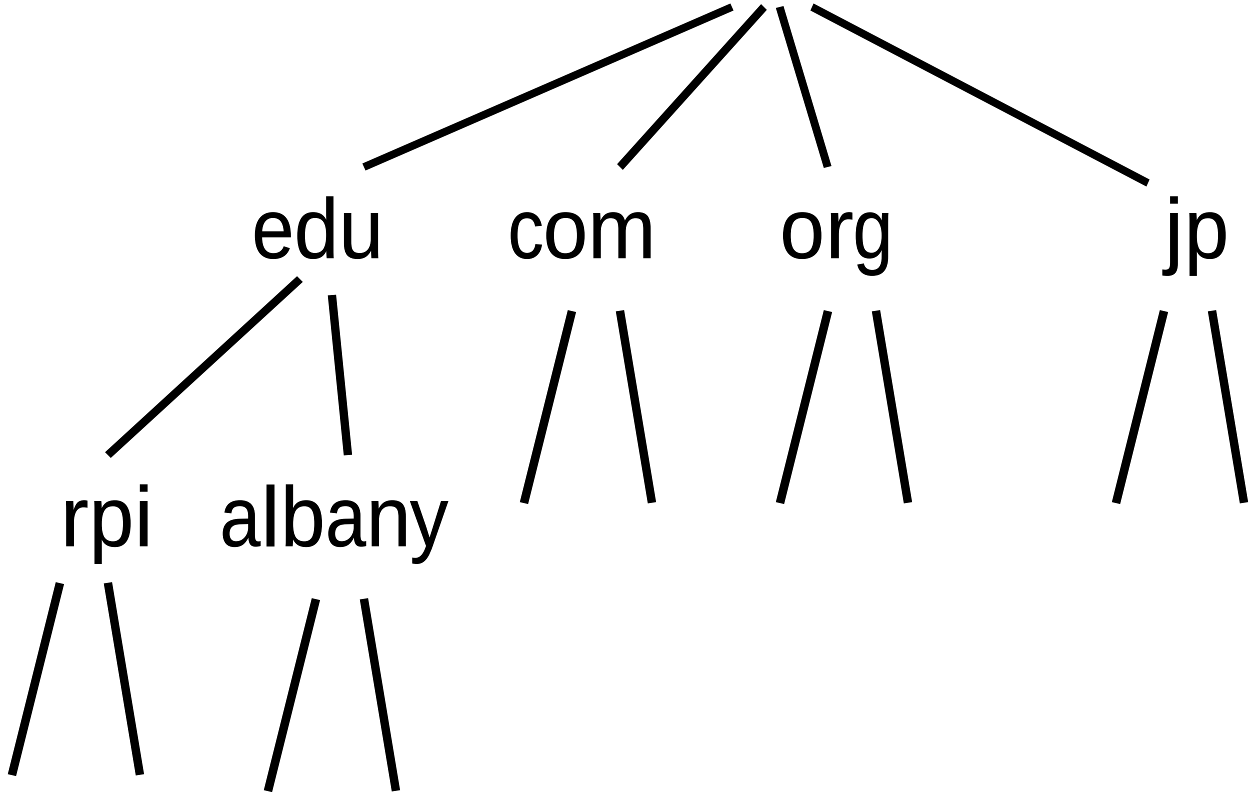
- IP Addresses are great for computers
  - IP address includes information used for routing.
- IP addresses are tough for humans to remember.
- IP addresses are impossible to guess.
  - ever guessed at the name of a WWW site?

The Domain Name System

- The *domain name system* is usually used to translate a host name into an IP address .

- Domain names comprise a hierarchy so that names are unique, yet easy to remember.

DNS Hierarchy



Host name structure

- Each host name is made up of a sequence of *labels* separated by periods.
  - Each label can be up to 63 characters
  - The total name can be at most 255 characters.
- Examples:
  - whitehouse.gov
  - barney.the.purple.dinosaur.com
  - monica.cs.rpi.edu

Domain Name

- The domain name for a host is the sequence of labels that lead from the host (leaf node in the naming tree) to the top of the worldwide naming tree.

- A domain is a subtree of the worldwide naming tree.

Top level domains

- **`edu, gov, com, net, org, mil`**, ...
- Countries each have a top level domain (2 letter domain name).
- New top level domains include:

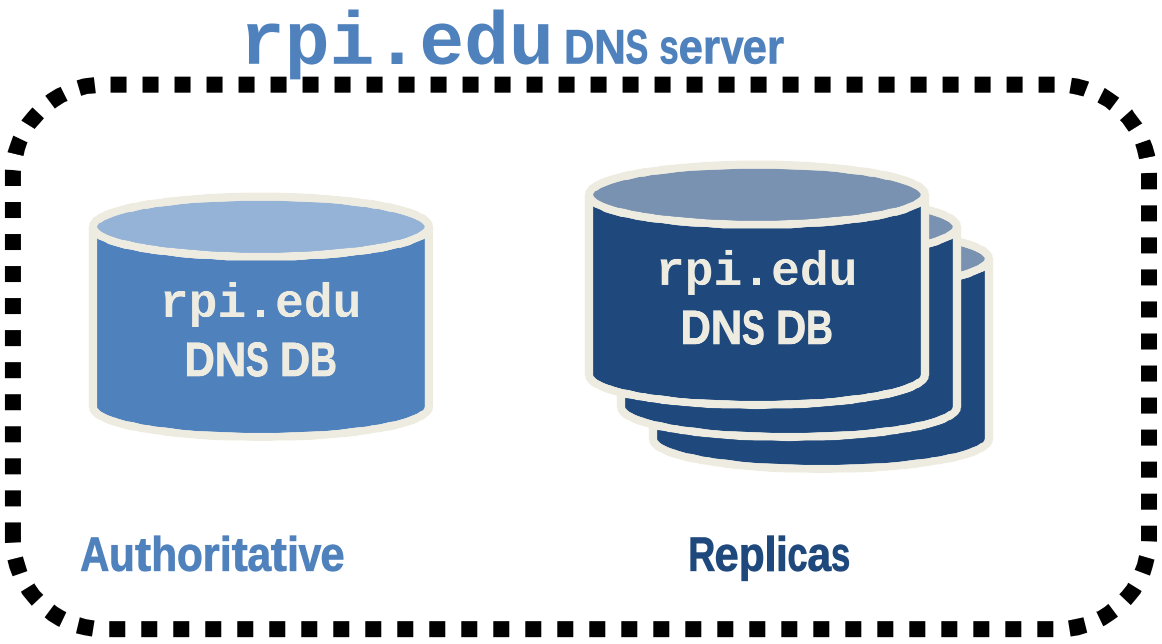
  .aero .biz .coop .info .name .pro

DNS Organization

- Distributed Database
  - The organization that owns a domain name is responsible for running a DNS server that can provide the mapping between hostnames within the domain to IP addresses.
  - So - some machine run by RPI is responsible for everything within the rpi.edu domain.

DNS Distributed Database

- There is one primary server for a domain, and typically a number of secondary servers containing replicated databases.



DNS Clients

- A DNS client is called a *resolver*.
- A call to **`gethostbyname()`**is handled by a resolver (typically part of the client).
- Most Unix workstations have the file **`/etc/resolv.conf`** that contains the local domain and the addresses of DNS servers for that domain.

/etc/resolv.conf

```
domain rpi.edu
128.113.1.5
128.113.1.3
```

`nslookup`

- **`nslookup`** is an interactive resolver that allows the user to communicate directly with a DNS server.

- **`nslookup`** is usually available on Unix workstations. (**`dig`** and **`host`** are also DNS clients).

DNS Servers

- Servers handle requests for their domain directly.
- Servers handle requests for other domains by contacting remote DNS server(s).
- Servers cache external mappings.

Server - Server Communication

- If a server is asked to provide the mapping for a host outside it’s domain (and the mapping is not in the server cache):
  - The server finds a nameserver for the target domain.
  - The server asks the nameserver to provide the host name to IP translation.
- To find the right nameserver, use DNS!

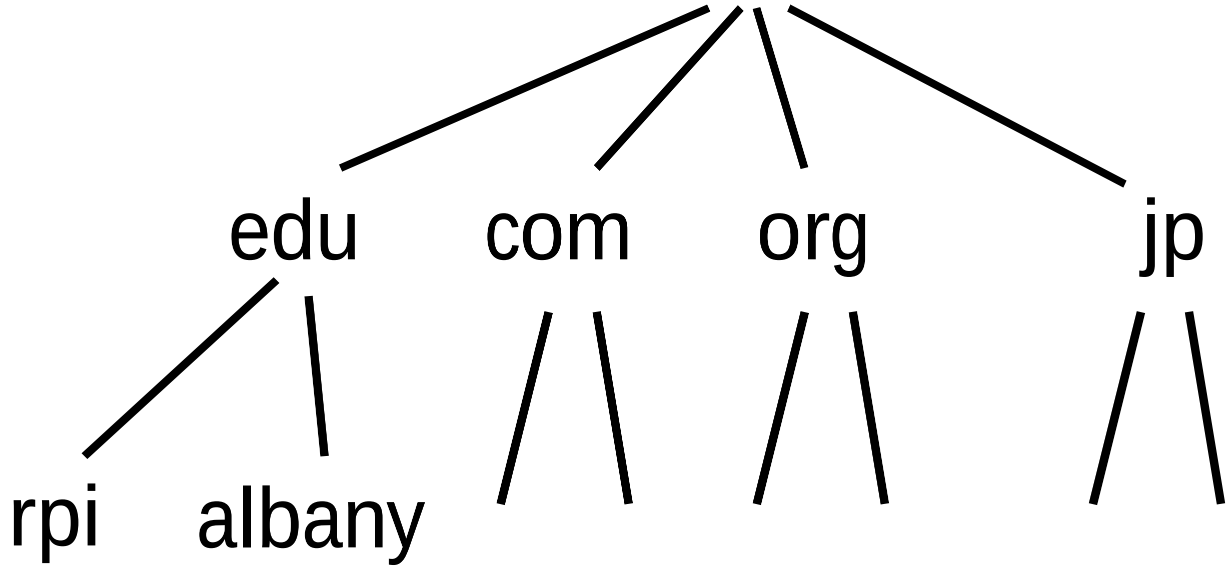
DNS Data

- DNS databases contain more than just hostname-to-address records:

– Name server records	NS
– Hostname aliases	CNAME
– Mail Exchangers	MX
– Host Information	HINFO

The Root DNS Server

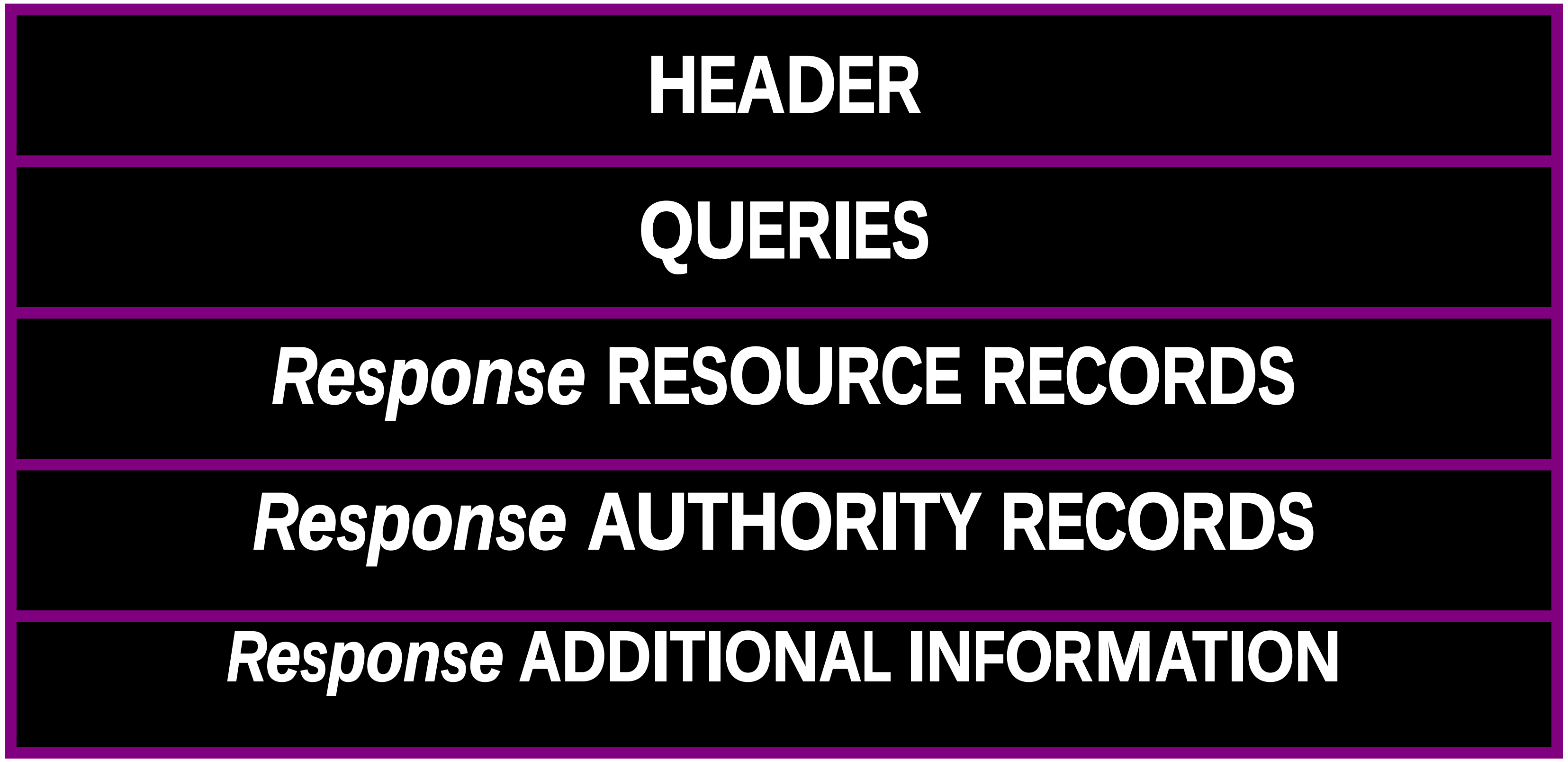
- The root server needs to know the address of 1st (and many 2nd) level domain nameservers.



Server Operation

- If a server has no clue about where to find the address for a hostname, ask the root server.
- The root server will tell you what nameserver to contact.
- A request may get forwarded a few times.

DNS Message Format


HEADER
QUERIES
Response RESOURCE RECORDS
Response AUTHORITY RECORDS
Response ADDITIONAL INFORMATION

DNS Message Header

16 bit fields

- query identifier
- flags
- # of questions
- # of RRs
- # of authority RRs
- # of additional RRs

} Response

Message Flags

- QR: Query=0, Response=1
- AA: Authoritative Answer
- TC: response truncated (> 512 bytes)
- RD: recursion desired
- RA: recursion available
- rcode: return code

Recursion

- A request can indicate that recursion is desired - this tells the server to find out the answer (possibly by contacting other servers).
- If recursion is not requested - the response may be a list of other name servers to contact.

Question Format

- Name: domain name (or IP address)
- Query type (A, NS, MX, …)
- Query class (1 for IP)

Response Resource Record

- Domain Name
- Response type
- Class (IP)
- Time to live (in seconds)
- Length of resource data
- Resource data

UDP & TCP

- Both UDP and TCP are used:
  - TCP for transfers of entire database to secondary servers (replication).
  - UDP for lookups
  - If more than 512 bytes in response - requestor resubmits request using TCP.

Lots more

- This is not a complete description !
- If interested - look at:
  - RFC 1034: DNS concepts and facilities.
  - RFC 1035: DNS implementation and protocol specification.
  - play with nslookup.
  - Look at code for BIND (DNS server code).

Name to Address Conversion

- There is a library of functions that act as DNS client (resolver).
  - you don't need to write DNS client code to use DNS!
- With some OSs you need to explicitly link with the DNS resolver library:

  `-lnsl` (**`nsl`** is "Name Server Library")

Suns (Solaris) need this!

DNS library functions

`gethostbyname`

`gethostbyaddr`

`gethostbyname2` ← **IPV6!**

gethostbyname

```
struct hostent *gethostbyname(
  const char *hostname);
```

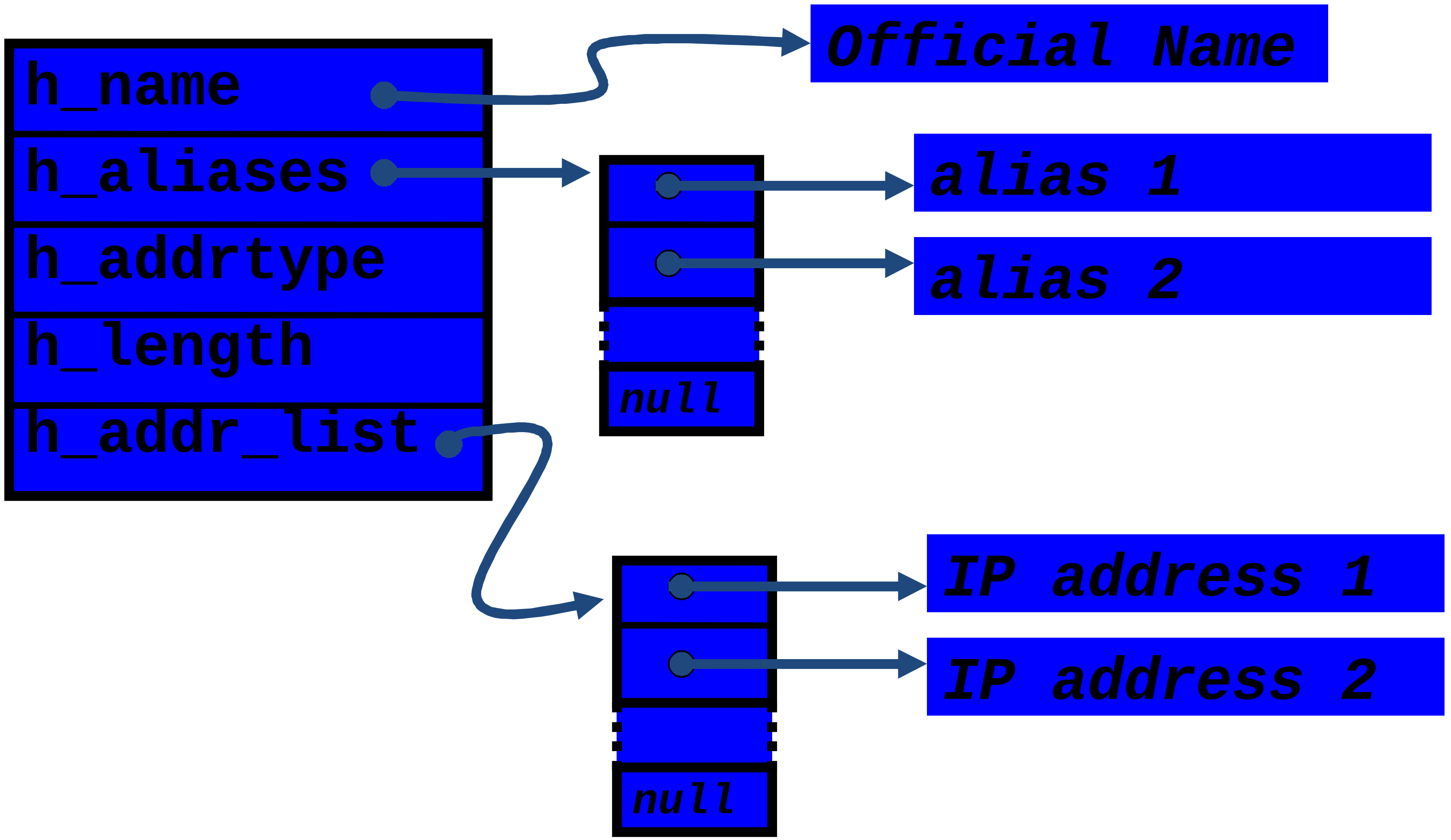
`struct hostent` is defined in netdb.h:

```
#include <netdb.h>
```

struct hostent

```
struct hostent {
  char *h_name;         official name (canonical)
  char **h_aliases;     other names
  int h_addrtype;       AF_INET or AF_INET6
  int h_length;         address length (4 or 16)
  char **h_addr_list;
};                      array of ptrs to addresses
```

`hostent` picture



Which Address?

On success, gethostbyname returns the address of a hostent that has been created.

- has an array of ptrs to IP addresses
- Usually use the first one:

```
#define h_addr h_addr_list[0]
```

`gethostbyname` and errors

- On error **`gethostbyname`** return null.
- **`Gethostbyname`** sets the global variable **`h_errno`** to indicate the exact error:
  - **`HOST_NOT_FOUND`**
  - **`TRY_AGAIN`**
  - **`NO_RECOVERY`**
  - **`NO_DATA`**
  - **`NO_ADDRESS`**

All defined in **`netdb.h`**

Getting at the address:

`char **h_addr_list;`

```
h = gethostbyname("joe.com");
sockaddr.sin_addr.s_addr =
  *(h->h_addr_list[0]);
```

This won't work!!!!

`h_addr_list[0]` is a `char*` !

Using **memcpy**

- You can copy the 4 bytes (IPv4) directly:

```
h = gethostbyname("joe.com");

memcpy(&sockaddr.sin_addr,
       h->h_addr_list[0],
       sizeof(struct in_addr));
```

Network Byte Order

- All the IP addresses returned via the hostent are in network byte order!

- Repeat after me:

  "Thank you **`gethostbyname`**!"

gethostbyaddr

```
struct hostent *gethostbyaddr(
  const char *addr
  size_t len,
  int family);
```

sizeof(struct in_addr)

AF_INET (could be AF_INET6)

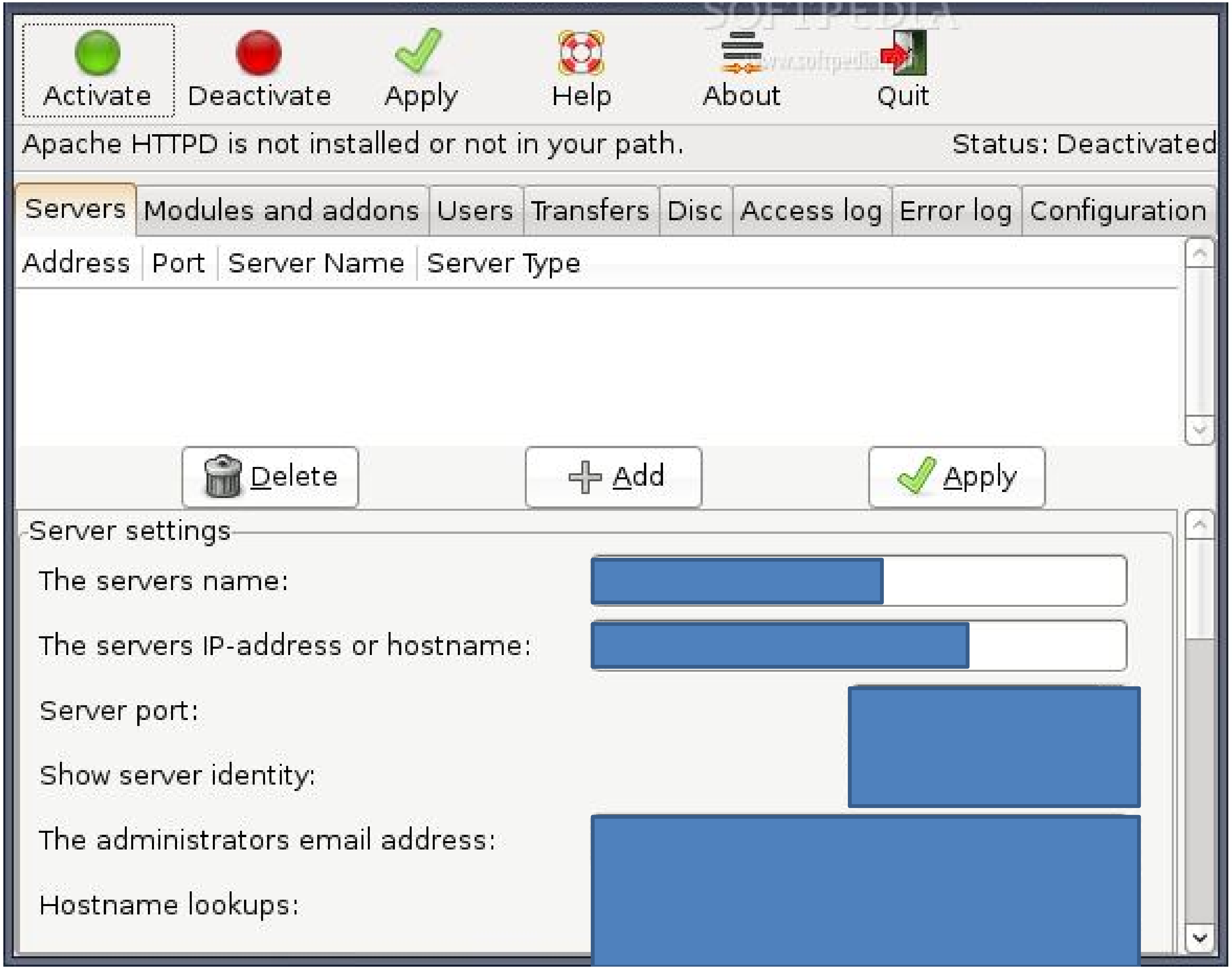
Some other functions

`uname` : get hostname of local host

`getservbyname` : get port number for a named service

`getservbyaddr` : get name for service associated with a port number

Using httpd to setup a www server



Apache HTTPD is not installed or not in your path. Status: Deactivated

Address	Port	Server Name	Server Type

Server settings

The servers name: [redacted]

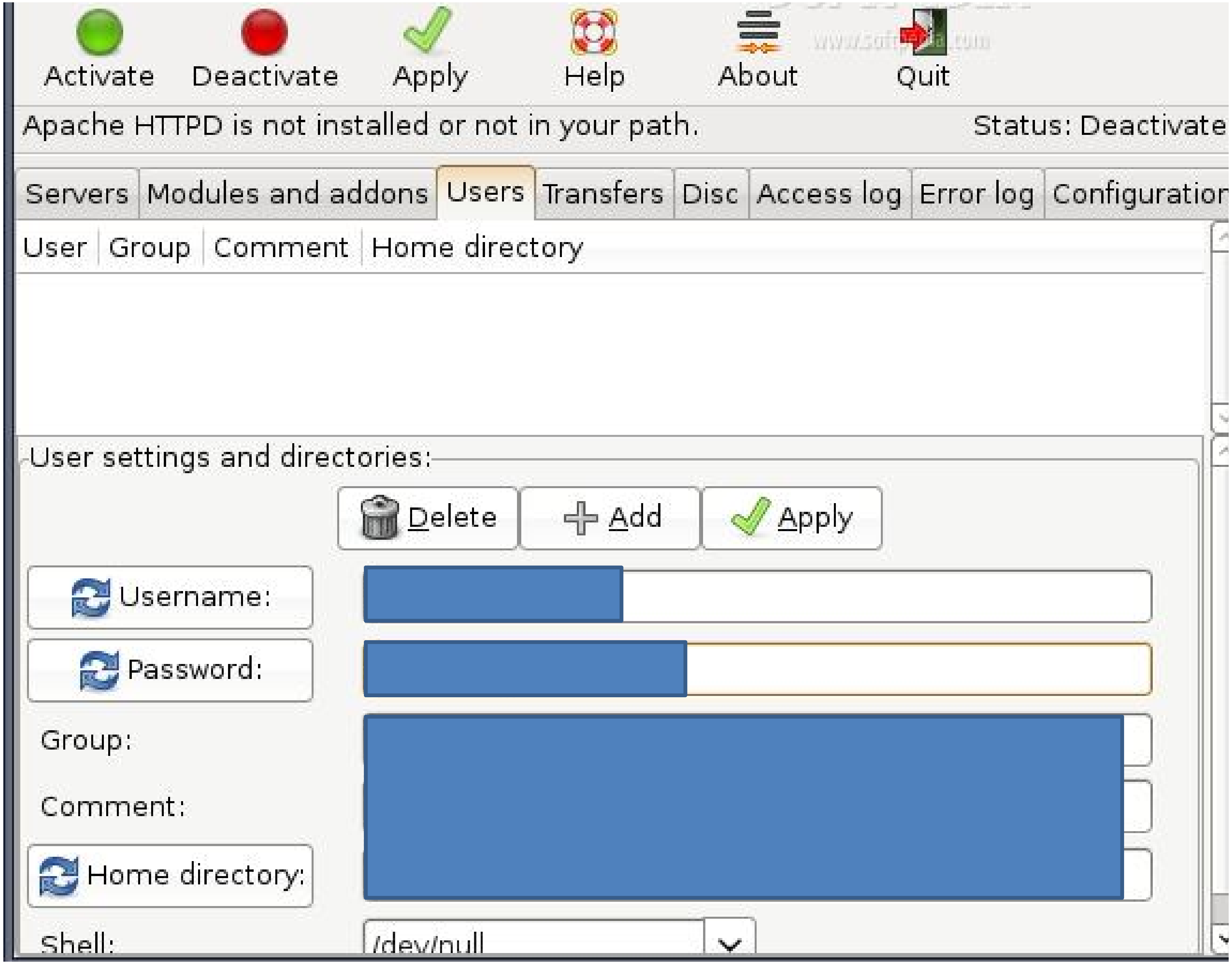
The servers IP-address or hostname: [redacted]

Server port: [redacted]

Show server identity:

The administrators email address: [redacted]

Hostname lookups:



Apache HTTPD is not installed or not in your path.   Status: Deactivate

User	Group	Comment	Home directory

User settings and directories:

Username: [redacted]

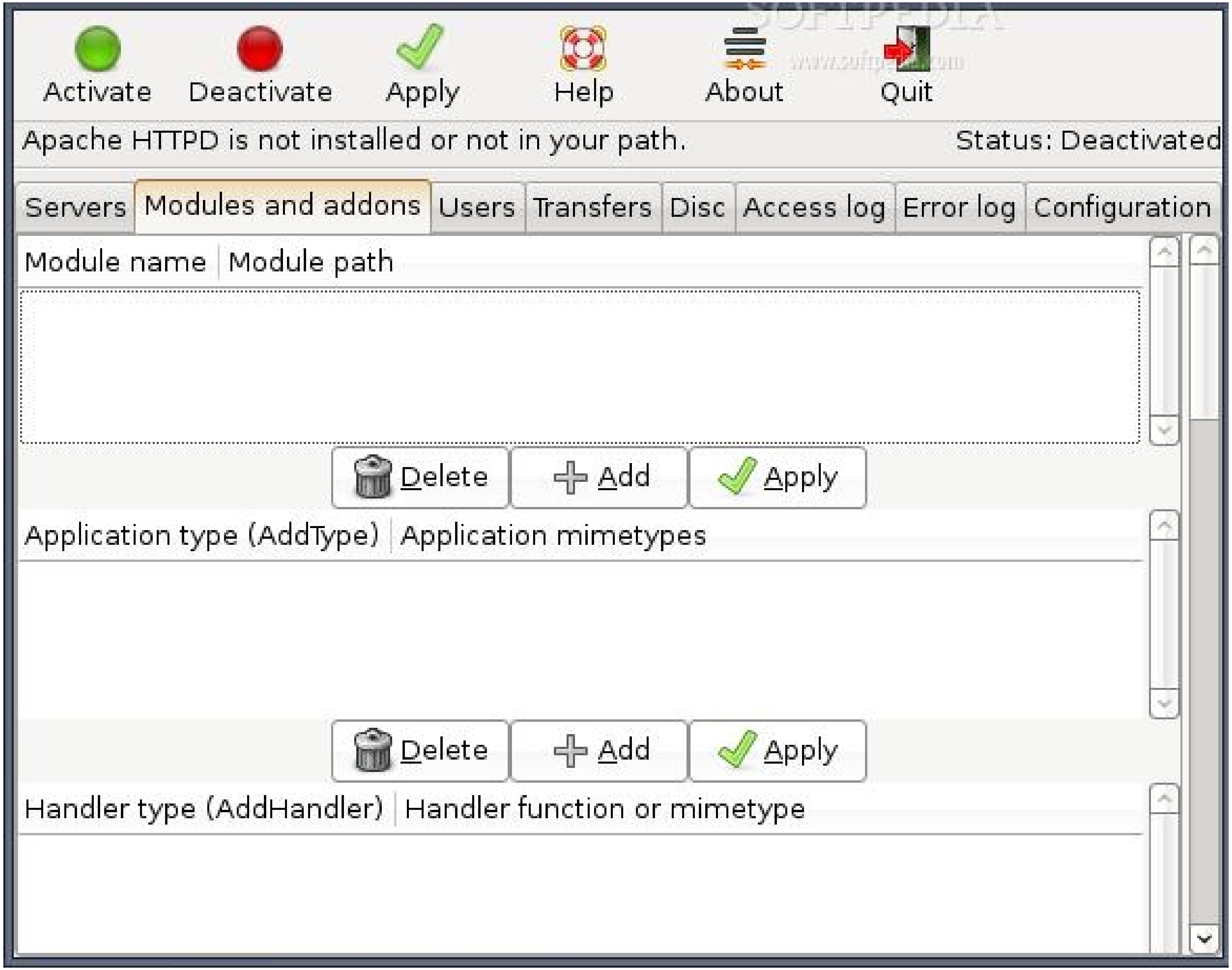
Password: [redacted]

Group: [redacted]

Comment:

Home directory:

Shell: /dev/null



Apache HTTPD is not installed or not in your path.

Status: Deactivated

Email Protocols

SMTP Commands

HELO <sp> <domain><crlf>

MAIL <sp>FROM:<reverse path><crlf>

RCPT <sp>TO:<forward path><crlf>

DATA<crlf>   terminates with <crlf>.<crlf>

RSET<cflf>

SEND<sp>FROM:<reverse path><crlf>

SOML<sp>FROM:<reverse path><crlf>

SAML<sp>FROM:<reverst path><crlf>

VRFY<sp><string<crlf>

EXPN<sp> <string><crlf>

HELP<sp><string><crlf>

NOOP<crlf>

QUIT<cflf>

TURN<cflf>

SMTP - Commands

- HELO
  - identifies the client to the server, fully qualified domain name, only sent once per session
- MAIL
  - initiate a message transfer, fully qualified domain of originator
- RCPT
  - follows MAIL, identifies an addressee, typically the fully qualified name of the addressee
  - for multiple addressees use one RCPT for each addressee
- DATA
  - send data line by line
  - <cr>.<cr> tells server data transfer is over

SMTP - Commands

- RSET
  - tells server to abort current message and clear all of it buffers
  - same state as after HELO
- SEND , SOML , SAML
  - like MAIL, outdated not used any more
- VRFY
  - ask server to verify a user name
  - server replies positively of it knows user, negatively if not
- EXPN
  - ask server to confirm mailing list alias
  - server reply is multi-line, one per user

SMTP - Commands

- HELP
  - ask server for help
    - by itself get a list of server supported commands
    - <string> get help for that command
- NOOP
  - ask server to respond with a positive reply
- QUIT
  - tell server that client is ending session
  - server replies positively and closes connection
- TURN
  - reverse roles of client and server
    - outdated, rarely used on modern internet

SMTP - Reply codes

- 211 - System status or help ready
- 214 - Help message
- 220 - <domain> Service ready
- 221 - <domain> Service closing transmission channel
- 250 - Requested mail action OK, ready
- 251 - User not local, will forward to <forward path>
- 354 - Start mail input; end with <crlf>.<crlf>
- 421 - <domain> Service not avail, closing transmission channel
- 450 - Requested mail action not taken, mailbox not available
- 451 - Requested action aborted, local error
- 452 - Requested action not taken, insufficient storage
- 500 - Syntax error, command unrecognized
- 501 - Syntax error in parameters
- 502 - Command not implemented
- 503 - Bad sequence of commands
- 504 - Command Parameter not implemented

Reply codes (more)

- 550 - Requested action not taken, mailbox unavailable
- 551 - User not local, please try <forward path>
- 552 - Requested mail action not taken; exceeded storage allocation
- 553 - Requested action not taken, mailbox name not allowed
- 554 - Transaction failed

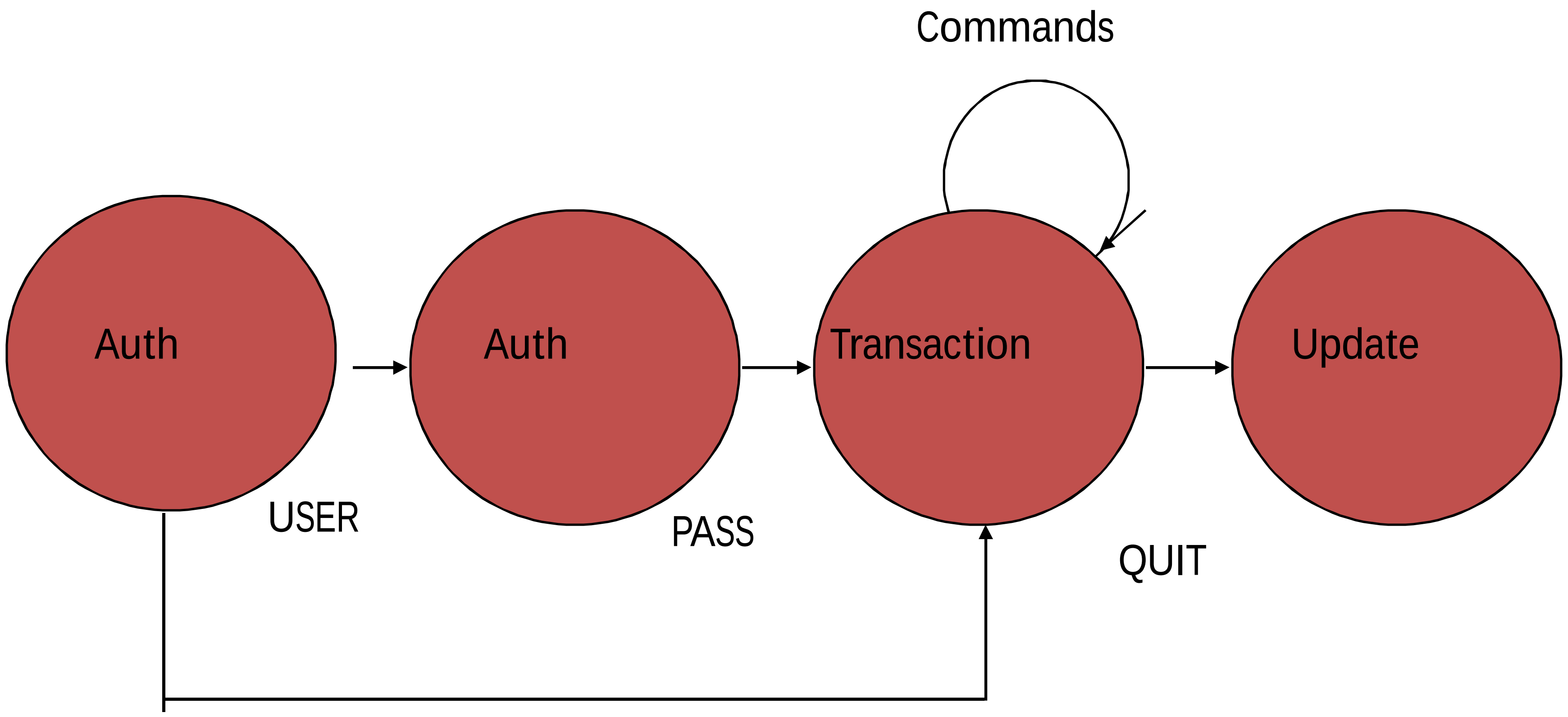
Post Office Protocol (POP3)

- Used in conjunction with anSMTP Host
  - SMTP Host sends and receives e-mail for remote users, POP allows users to retreive their mail from the host.
  - SMTP stores mail for unconnected hosts
- RFC 1730
- TCP Port 110

POP3

- protocol is relatively simple
  - connect to port 110 of remote host
    - read back a response check for OK or ERR
    - over and over again
  - close the connection

POP3 - State Machine



POP3 - Commands

- Commands
  - USER name
    - terminate with <crlf>
    - identifies the user/mail drop name
  - PASS string
    - user password
    - usually the same as the user’s logon password
  - STAT
    - request number of messages on server and size of mail drop

POP3 - Commands

- LIST
  - return a list <crlf> of all msgs on server
    - format   msg size
- LIST [msg_no]
  - request size of msg_no
    - format   msg_no   size
- RETR msg_no
  - return the message identified by msg_no

POP3 - Commands

- DELE msg_no
  - delete msg_no from server
  - happens in UPDATE State
- NOOP
  - nothing except a positive reply from server
- RSET
  - reset all deletions pending on server
- QUIT
  - quit session, UPDATE, enter AUTH1 State

IMAP

- Developed after POP and attempts to fix POP deficiencies
  - allows keeping all mail on the server
  - allows mail categorization via folder metaphor
  - mail is easily flagged (answered, draft, deleted, seen, recent); this isn’t the same on all servers
  - provides for multiple connections to the server

IMAP - process

- make connection
- send user credentials (userid and password)
  - repeat until done
    - send a command
    - read response
  - disconnect

IMAP Command

- tag command argurment(s)
  - tag, either a “.” or a text string that can be sequentialized (a0001, a0002, a0003….); if only a single connection use “.” , if multiple connections use text string (this allows matching commands with responses).

IMAP - Commands

- login
- list
- status
- examine
- select
- create, delete, rename
- fetch
- store
- close
- expunge
- copy
- idle
- lsub, subscribe, unsubscribe
- logout
- capability, getquotaroot, getacl

IMAP - Commands

- login
  - userid@address
  - password

  - example
    . login steflik@binghamton.edu xyz123
    (if not using ssl this goes as plain text just like pop)
  - response – server should acknowledge with OK

IMAP Commands

- logout command
  - no arguments
- The command is sent to the server, the server replys with a BYE message followed by an OK message and closes the connection.

IMAP - Commands

- list
  - retrieves a list of the mailboxes/folders
  - argruments :
    reference name:
    mailbox name w/possible wildcards
  - returns an untagged list of the mailboxes/folders along with a separator char and an indication of hierarchy followed by a tagged OK
  - for details refer to rfc 1730

IMAP Commands

- status command
  - arguments:
    mailbox/folder
    (space delimited list of flags)
  - return info you asked for; untagged list
  - ex:

```
a006 status inbox (messages uidnext)
* STATUS “inbox” (MESSAGES 404 UIDNEXT 1001)
 a006 OK Success
```

IMAP Commands

- examine & select commands
  - arguments:
    mailbox/folder
  - returns: Flags information, how many of messages each flag type are in the folder; then allows access to the messages in the folder
  - examine allows read-only
    select allows read-write access

IMAP Commands

- create, delet and rename commands
- create newfoldernane – creates a new folder
- delete foldername – deletes the named folder
- rename oldname newname – renames the folder
- foldernames must be fully qualified using the separator char from the info returned by the list command
-

IMAP Commands

- fetch command – used to actually access e-mails
- arguments:

  message number or range of numbers

  (1 , 1:2 , 1:last)

  what it is you want to retrieve

  (fast, all, text, rfc822.header….see rfc)

IMAP Commands

- store – lets you set the flags for messages
- arguments:

  message number/range

  flags to be set

  (\Answered \Flagged \Draft \Deleted \Seen hasatt Junk)

IMAP Commands

- close & expunge commands – used to permanently delete a message(s) in the current folder that has the \Deleted flag set.

IMAP Commands

- copy command – copy a message or range of messages to another folder then delete the originals
- arguments:
  - message number or range
  - target folder
- ex: copy 1:3 linux.debian

IMAP Commands

- idle command – lets you monitor a folder until something new is added to it

IMAP Commands

- capability command – untagged list of the servers capabilities
- getquotaroot – returns the amount of space you are using and how much is available
  - ex: getquitaroot inbox
- getacl command – returns the access control list for the specified folder (l,r,s,w,I,p,c,d,a)

MIME

- Important RFCs
  - RFC-822 Standard for the format for ARPA Internet test messages
  - RFC-2045 MIME Part 1: Format of Internet Message Bodies
  - RFC-2046 MIME Part 2: Media Types
  - RFC-2047 MIME Part 3: Message Header Extensions
  - RFC-2048 MIME Part 4: Registration Procedure
  - RFC-2049 MIME Part 5: Conformance Criteria

MIME – What is it?

- MIME refers to an official Internet standard that specifies how messages must be formatted so that they can be exchanged between different email systems.
- MIME permits the inclusion of virtually any type of file or document in an email message.
- Specifically, MIME messages can contain
  - text
  - images
  - audio
  - video
  - application-specific data.
    - spreadsheets
    - word processing documets

MIME - Features

- Support of character sets other than ASCII
- Content type labeling System
- Support of non-text content in e-mail messages
- Support for compound documents

MIME - Non-ASCII Character support

- Message header
  - content-type field
    - put in the header by the client program creating the e-mail for use by the client program used to display the received message
    - charset= optional parameter
      - if absent ASCII is assumed

- Content-Type: text/plain; charset="ISO-8859-1"
  - ISO-8859-1 extends the basic character set of ASCII to include many of the accented characters used in languages such as Spanish, French, and German.
  - US-ASCII is the standard character set used in the US

MIME - Content Labeling

- a set of registered MIME Types that map to specific file types
  - MIME Types consist of :
    - a primary type
    - a sub type separated by a / (as text/html)
- Common Mime Types:

FileExtension	MIME Type	Description
.txt	text/plain	Plain text
.htm	text/html	Styled text in HTML format
.jpg	image/jpeg	Picture in JPEG format
.gif	image/gif	Picture in GIF format
.wav	audio/x-wave	Sound in WAVE format
.mp3	audio/mpeg	Music in MP3 format
.mpg	video/mpeg	Video in MPEG format
.zip	application/zip	Compressed file in PK-ZIP format

Non-text Content

- To be sent through the e-mail system non-textual content must be converted (encoded) to ASCII for transmission and unencode back to its original format for display upon receipt.
  - originally done via uuencode
  - MIME uses base 64 encoding (RFC 2045)
    - binary to text encoding scheme
    - targets A-Z, a-z,0-9, +,/
    - scheme:
      - take three byte of data, put into a 24 bit buffer
      - extract 4 six bit values
      - use each value as an index into:
        - » ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
      - this yields 4 ASCII characters

MIME - base64 encoding example

Man is distinguished, not only by his reason, but by this singular passion from other animals, which is a lust of the mind, that by a perseverance of delight in the continued and indefatigable generation of knowledge, exceeds the short vehemence of any carnal pleasure.

base64 encoded:

TWFuIGlzIGRpc3Rpbmd1aXNoZWQsIG5vdCBvbmx5IGJ5IGhpcyByZWFzb24sIGJ1dCBieSB0
aGlzIHNpbmd1bGFyIHBhc3Npb24gZnJvbSBvdGhlciBhbmltYWxzLCB3aGljaCBpcyBhIGx1
c3Qgb2YgdGhlIG1pbmQsIHRoYXQgYnkgYSBwZXJzZXZlcmFuY2Ugb2YgZGVsaWdodCBpbiB0
aGUgY29udGludWVkIGFuZCBpbmRlZmF0aWdhYmxlIGdlbmVyYXRpb24gb2Yga25vd2xlZGdl
LCBleGNlZWRzIHRoZSBzaG9ydCB2ZWhlbWVuY2Ugb2YgYW55IGNhcm5hbCBwbGVhc3VyZS4=

MIME - Multipart Messages

- use content-type = multipart/sub type
  - sub types :
    - related
    - mixed
- see examples at http://www.hunnysoft.com/mime/samples/index.htm

Security

There are four basic elements in security:

- ***Privacy,* restriction of access.**
- ***Authentication*: verification of identity.**
- ***Trust*: trusting the source.**
- ***Integrity*, protection against corruption or loss (redundancy).**

Security Policy

A **network security policy** is a generic document that outlines rules for computer_network access, determines how policies are enforced and lays out some of the basic architecture of the company security/ network security environment - wiki

How secure must we be

- From outside the organization?
- From inside the organization (different host)?
- From inside the organization (same host)?
- Against the interruption of services?
- From user error?

Principle 44 (<u>Work defensively</u>) ***Expect the worst, do your best, preferably in advance of a problem.***

Principle 45 (Network Security) ***Extremely sensitive data should not be placed on a computer which is attached in any way to a public network***

What resources are we trying to protect?

Who are we trying to protect them from?

What is the Intrusion Detection

- Intrusions are the activities that violate the security policy of system.
- Intrusion Detection is the process used to identify intrusions.

Types of Intrusion Detection System(1)

Based on the sources of the audit information used by each IDS, the IDSs may be classified into

- Host-base IDSs
- Distributed IDSs
- Network-based IDSs

Types of Intrusion Detection System(2)

- Host-based IDSs
  - Get audit data from host audit trails.
  - Detect attacks against a single host
- Distributed IDSs
  - Gather audit data from multiple host and possibly the network that connects the hosts
  - Detect attacks involving multiple hosts
- Network-Based IDSs
  - Use network traffic as the audit data source, relieving the burden on the hosts that usually provide normal computing services
  - Detect attacks from network.

What is a Firewall?

- A **choke point** of control and monitoring
- Interconnects networks with differing trust
- Imposes restrictions on network services
  - only authorized traffic is allowed
- Auditing and controlling access
  - can implement alarms for abnormal behavior
- Itself immune to penetration
- Provides **perimeter defence**

Methods for detection/protection/defense:

- Firewall: The Traffic cop
- IDS: detection and alert

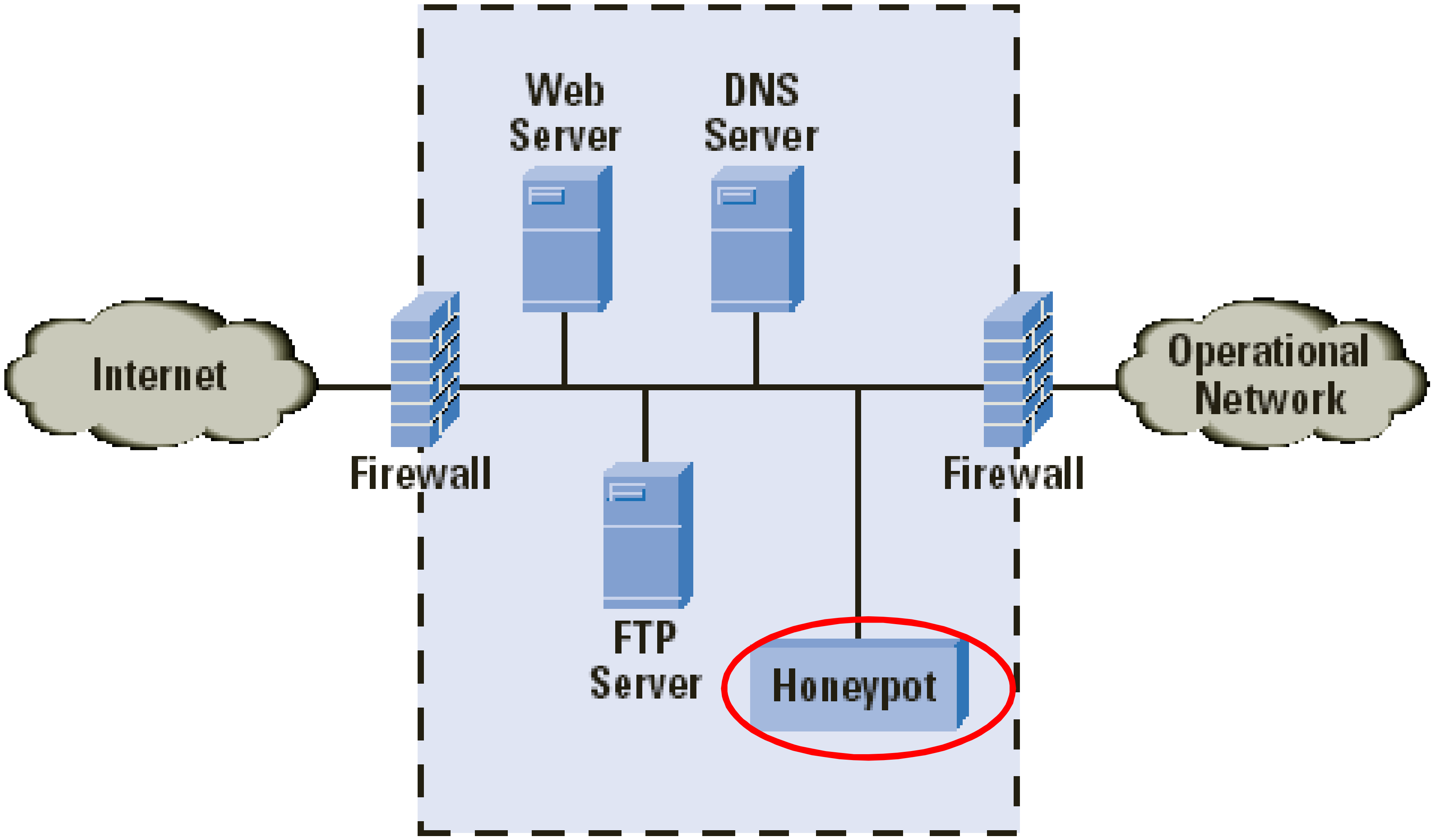
These have shortcomings:

- Internal threats
- Virus laden programs
- False Positives and False negatives

Honeynet: An additional layer

Any security resource who's value lies in being probed, attacked, or compromised

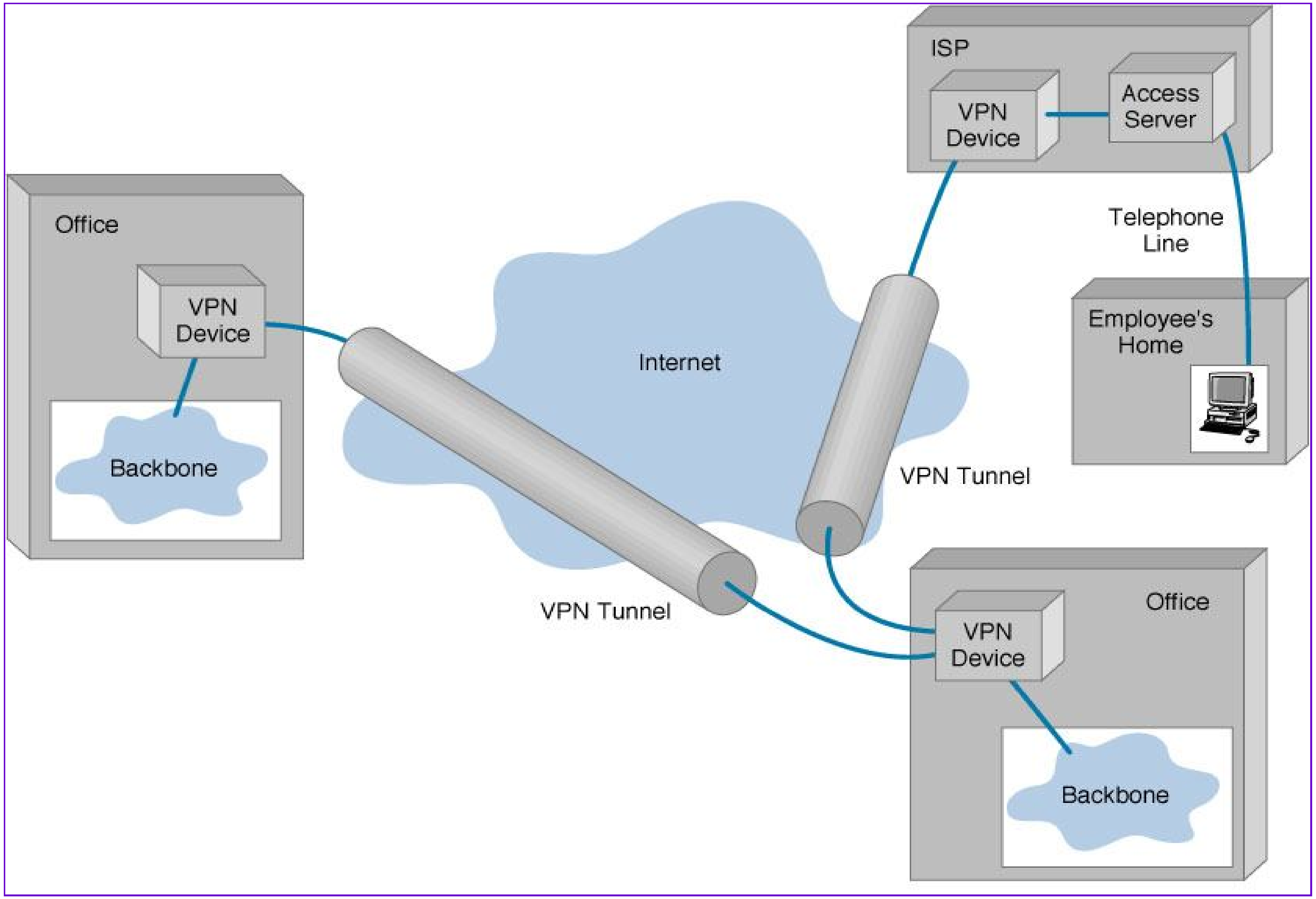
A resource that expects no data, so any traffic to or from it is most likely unauthorized activity


Web Server
DNS Server
Internet
Firewall
Firewall
Operational Network
FTP Server
Honeypot

What is VPN?

- Virtual Private Network is a type of private network that uses public telecommunication, such as the Internet, instead of leased lines to communicate.
- Became popular as more employees worked in remote locations.
- Terminologies to understand how VPNs work.

Basic Architecture



Analytical System Administration

Faults

The IEEE classification of software anomalies is

- Operating system crash.
- Program hang-up.
- Program crash.
- Input problem.
- Output problem.
- Failed required performance.
- Perceived total failure.
- System error message.
- Service degraded.
- Wrong output.
- No output

Reliability

average (mean) time before failure

$$R = \frac{\text{Mean uptime}}{\text{Total elapsed time}}$$

Rate of service (delivery) = change in information/rate of failure

Metrics for evaluating network performance

- *Total number of packets*
- *Amount of IP fragmentation*
- *Density of broadcast messages*
- *Number of collisions*
- *Number of sockets (TCP) in and out*
- *Number of malformed packets*
- *Latency of services*

Metrics for evaluating system performance

- *Disk usage in bytes*
- *Disk operations per second*
- *Paging (out) rate (free memory and thrashing):*
- *Number of processes & privileged processes*
- *Maximum percentage CPU used in processes*
- *Number of users logged on*
- *Average time spent logged on per user*
- *Load average*
- *Latency of services*

Network-level services

Network-level services

1. The Internet
2. A recap of networking concepts
3. Getting traffic to its destination
    1. Multiplexing
    2. From bridges to switches
    3. Virtual circuits and wide area switching

4. Alternative network transport technologies
    1. Medium sharing
    2. Token rings
    3. Ethernet
    4. Digital Subscriber Line (DSL)
    5. Integrated Services Digital Network (ISDN)
    6. Fiber: SONET/SDH
    7. T1 and E1

5. Alternative network connection technologies
    1. X.25
    2. Frame Relay
    3. Asynchronous Transfer Mode (ATM)
6. IP routing and forwarding
    1. Static routing
    2. Source routing
    3. Routing protocols – DV, LS
7. Multi-Protocol Label Switching (MPLS)
8. Quality of Service