UNIT V

APPLICATION LAYER

DNS

The **Domain Name System** (**DNS**) is a hierarchical distributed naming system for computers, services, or any resource connected to the Internet or a private network. It associates various information with domain names assigned to each of the participating entities. Most prominently, it translates domain names, which can be easily memorized by humans, to the numerical IP addresses needed for the purpose of computer services and devices worldwide. The Domain Name System is an essential component of the functionality of most Internet services because it is the Internet's primary directory service.

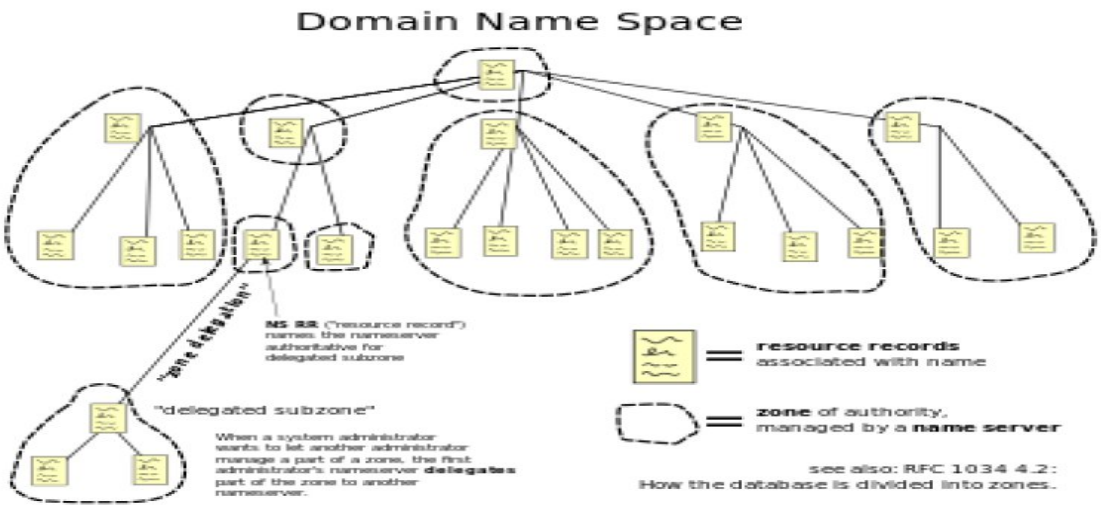
The Domain Name System distributes the responsibility of assigning domain names and mapping those names to IP addresses by designating authoritative name servers for each domain. Authoritative name servers are assigned to be responsible for their supported domains, and may delegate authority over sub-domains to other name servers. This mechanism provides distributed and fault tolerant service and was designed to avoid the need for a single central database.

The Domain Name System also specifies the technical functionality of the database service which is at its core. It defines the DNS protocol, a detailed specification of the data structures and data communication exchanges used in DNS, as part of the Internet Protocol Suite. Historically, other directory services preceding DNS were not scalable to large or global directories as they were originally based on text files, prominently the HOSTS.TXT resolver. DNS has been in wide use since the 1980s.

The Internet maintains two principal namespaces, the domain name hierarchy and the Internet Protocol (IP) address spaces. The Domain Name System maintains the domain namehierarchy and provides translation services between it and the address spaces. Internet name servers and a communication protocol implement the Domain Name System.[3] A DNS nameserver is a server that stores the DNS records for a domain name; a DNS name server responds with answers to queries against its database.

Domain name space

The domain name space consists of a tree of domain names. Each node or leaf in the tree has zero or more resource records, which hold information associated with the domain name. The tree sub-divides into zones beginning at the root zone. A DNS zone may consist of only one domain, or may consist of many domains and sub-domains, depending on the administrative authority delegated to the manager



The hierarchical Domain Name System, organized into zones, each served by a name server Administrative responsibility over any zone may be divided by creating additional zones. Authority is said to be delegated for a portion of the old space, usually in the form of sub-domains, to another name server and administrative entity. The old zone ceases to be authoritative for the new zone.

Domain name syntax

The definitive descriptions of the rules for forming domain names appear in RFC 1035, RFC 1123, and RFC 2181. A domain name consists of one or more parts, technically called labels, that are conventionally concatenated, and delimited by dots, such as example.com.
The hierarchy of domains descends from right to left; each label to the left specifies a subdivision, or subdomain of the domain to the right. For example: the label example specifies a subdomain of the com domain, and www is a subdomain of example.com. This tree of subdivisions may have up to 127 levels.
Each label may contain up to 63 characters. The full domain name may not exceed the length of 253 characters in its textual representation. In the internal binary representation of theDNS the maximum length requires 255 octets of storage, since it also stores the length of the name. DNS names may technically consist of any character representable in an octet. However, the allowed formulation of domain names in the DNS root zone, and most other sub domains, uses a preferred format and character set. The characters allowed in a label are a subset of the ASCII character set, and includes the characters a through z, A through Z, digits 0 through 9, and the hyphen. This rule is known as the LDH rule (letters, digits, hyphen). Domain names are interpreted in case-independent manner. Labels may not start or end with a hyphen. Anadditional rule requires that top-level domain names should not be all-numeric.

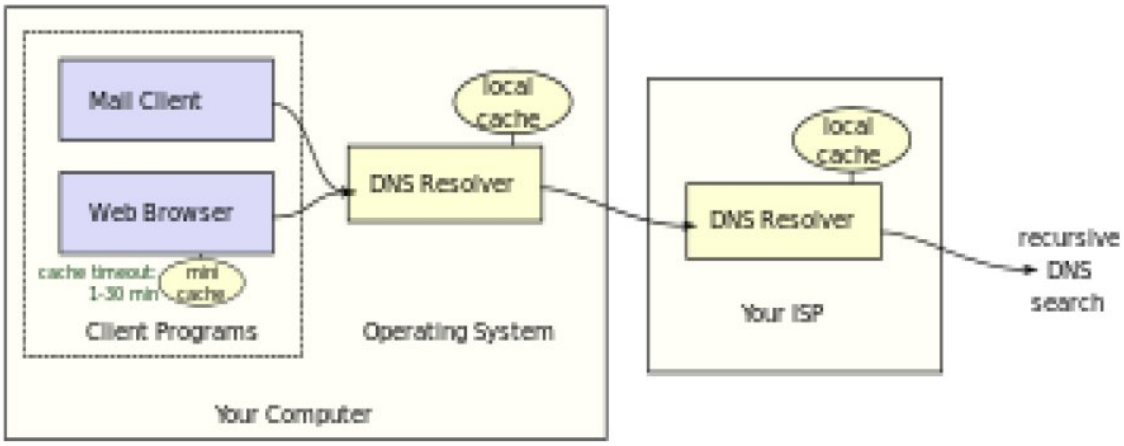
Internationalized domain names

The limited set of ASCII characters permitted in the DNS prevented the representation of names and words of many languages in their native alphabets or scripts. To make this possible, ICANN approved the Internationalizing Domain Names in Applications (IDNA) system, by which user applications, such as web browsers, map Unicode strings into the valid DNS character set using Punycode. In 2009 ICANN approved the installation of internationalized domain name country code top-level domains. In addition, many registries of the existing top level domain names (TLD)s have adopted the IDNA system.

Name servers
The Domain Name System is maintained by a distributed database system, which uses the client–server model. The nodes of this database are the name servers. Each domain has at least one authoritative DNS server that publishes information about that domain and the name servers of any domains subordinate to it. The top of the hierarchy is served by the root name servers, the servers to query when looking up (resolving) a TLD.

Authoritative name server

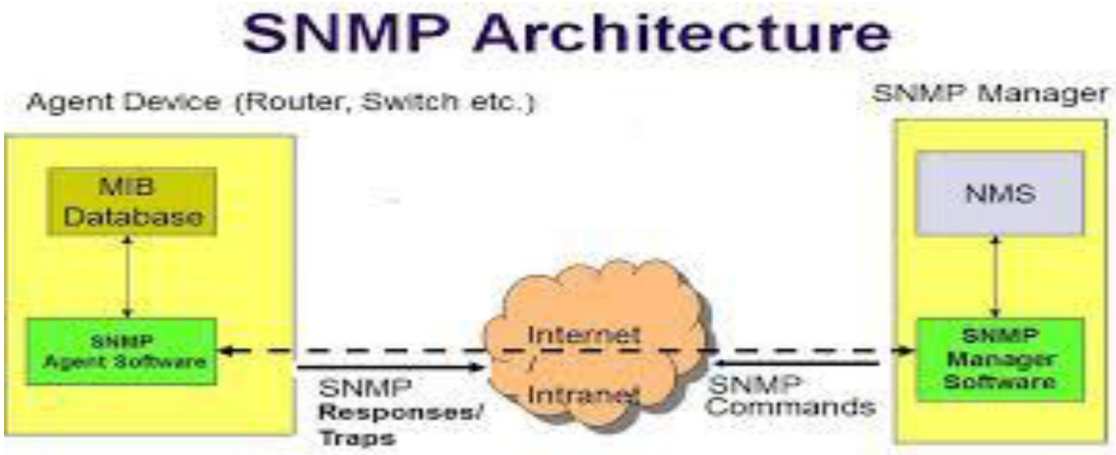
An authoritative name server is a name server that gives answers that have been configured by an original source, for example, the domain administrator or by dynamic DNS methods, in contrast to answers that were obtained via a regular DNS query to another name server. An authoritative-only name server only returns answers to queries about domain names that have been specifically configured by the administrator.
In other words, an authoritative name server lets recursive name servers know what DNS data (the IPv4 IP, the IPv6 IP, a list of incoming mail servers, etc.) a given host name (such as "www.example.com") has. As just one example, the authoritative name server for "example.com" tells recursive name servers that "www.example.com" has the IPv4 IP address 192.0.43.10. An authoritative name server can either be a master server or a slave server. A master server is a server that stores the original (master) copies of all zone records. A slave server uses an automatic updating mechanism of the DNS protocol in communication with its master to maintain an identical copy of the master records. A set of authoritative name servers has to be assigned for every DNS zone. An NS record about addresses of that set must be stored in the parent zone and servers themselves (as self-reference).
When domain names are registered with a domain name registrar, their installation at the domain registry of a top level domain requires the assignment of a primary name server and at least one secondary name server. The requirement of multiple name servers aims to make the domain still functional even if one name server becomes inaccessible or inoperable.[10] Thedesignation of a primary name server is solely determined by the priority given to the domain name registrar. For this purpose, generally only the fully qualified domain name of the name server is required, unless the servers are contained in the registered domain, in which case the corresponding IP address is needed as well. Primary name servers are often master name servers, while secondary name servers may be implemented as slave servers. An authoritative server indicates its status of supplying definitive answers, deemed authoritative, by setting a software flag (a protocol structure bit), called the Authoritative Answer (AA) bit in its responses.



Domain **N**ame **S**ystem (or **S**ervice or **S**erver), an Internet service that translates domain names into IP addresses. Because domain names are alphabetic, they're easier to remember. The Internet however, is really based on IP addresses. Every time you use a domain name, therefore,aDNS service must translate the name into the corresponding IP address. For example, the domain name www.example.com might translate to 198.105.232.4

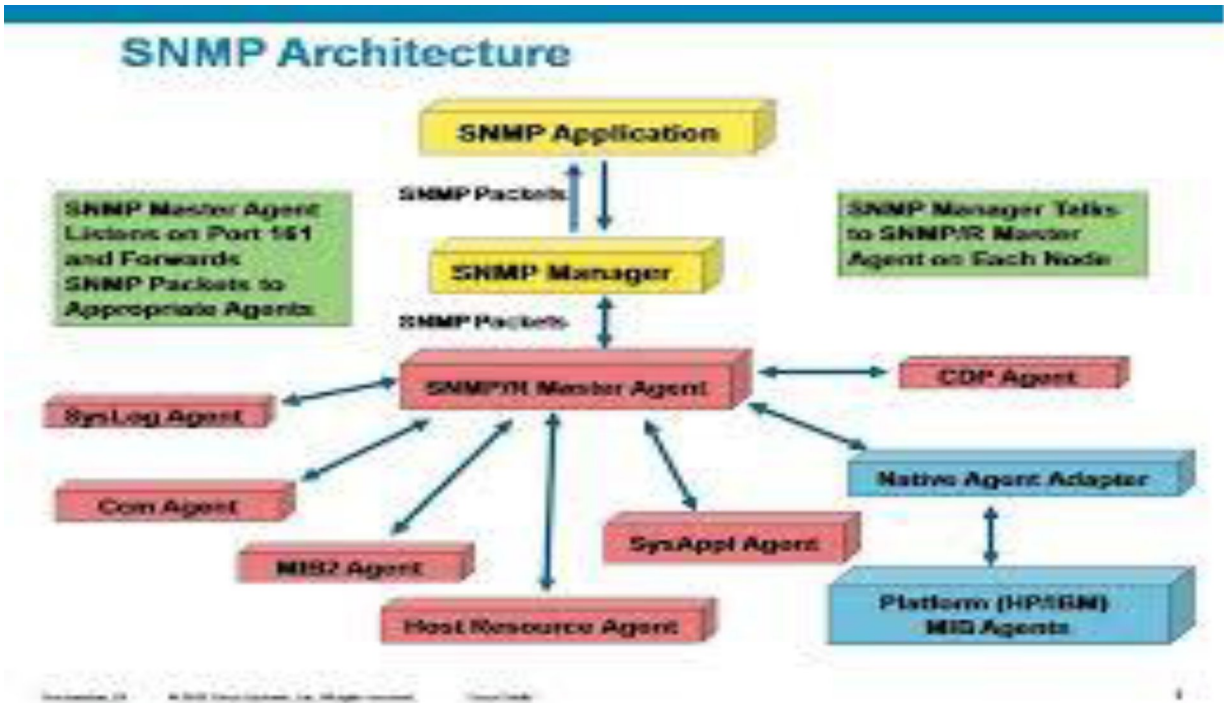
SNMP

Simple **N**etwork **M**anagement **P**rotocol, a set of protocols for managing complex networks. The first versions of SNMP were developed in the early 80s. SNMP works by sending messages, called protocol data units (PDUs), to different parts of a network. SNMP-compliant devices, called agents, store data about themselves in Management Information Bases (MIBs) and return this data to the SNMP requesters.



Simple Network Management Protocol (**SNMP**) is an "Internet-standard protocol for managing devices on IP networks". Devices that typically support SNMP include routers, switches, servers, workstations, printers, modem racks and more. SNMP is widely used in network management systems to monitor network-attached devices for conditions that warrant administrative attention. SNMP is a component of the Internet Protocol Suite as defined by the Internet Engineering Task Force (IETF). It consists of a set of standards for network

management, including an application layer protocol, a database schema, and a set of data objects.



SNMP exposes management data in the form of variables on the managed systems, which describe the system configuration. These variables can then be queried (and sometimes set) by managing applications.

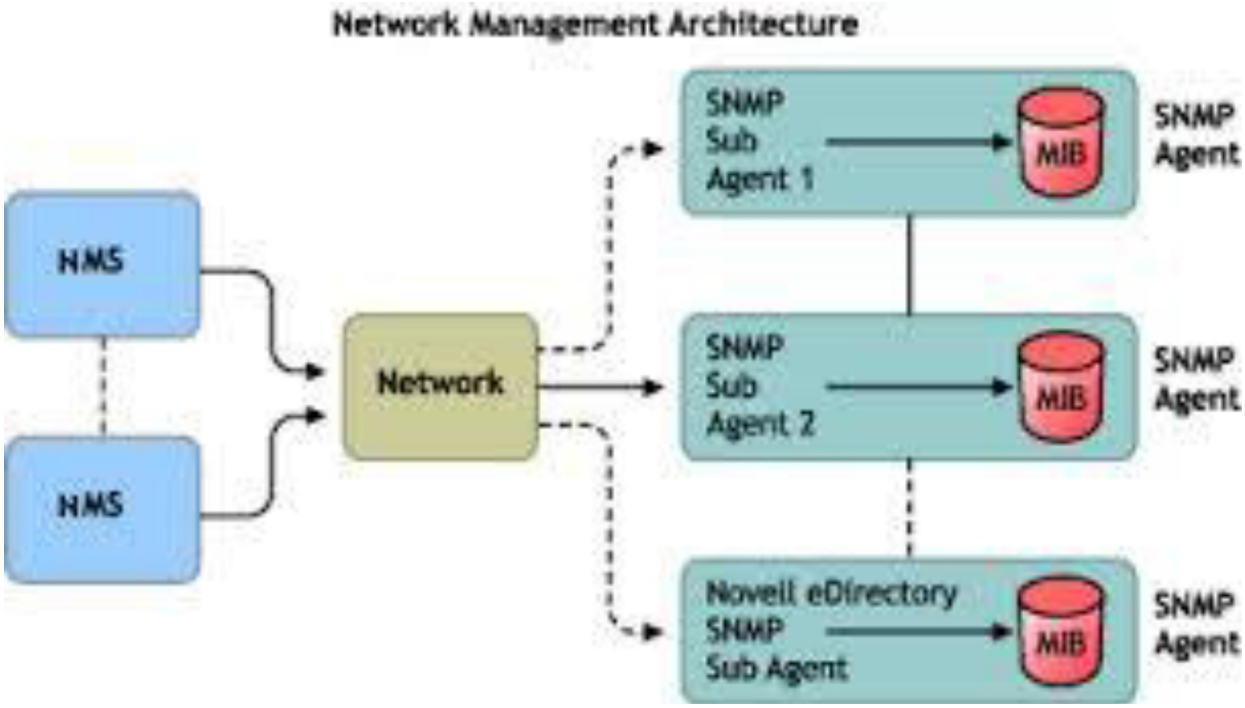
In typical uses of SNMP one or more administrative computers, called managers, have the task of monitoring or managing a group of hosts or devices on a computer network. Each managed system executes, at all times, a software component called an agent which reports information via SNMP to the manager.

SNMP agents expose management data on the managed systems as variables. The protocol also permits active management tasks, such as modifying and applying a new configuration through remote modification of these variables. The variables accessible via SNMP are organized in hierarchies. These hierarchies, and other metadata (such as type and description of the variable), are described by Management Information Bases (MIBs). An SNMP-managed network consists of three key components:

- Managed device
- Agent — software which runs on managed devices
- Network management station (NMS) — software which runs on the manager

A managed device is a network node that implements an SNMP interface that allows unidirectional (read-only) or bidirectional (read and write) access to node-specific information. Managed devices exchange node-specific information with the NMSs. Sometimes called network elements, the managed devices can be any type of device, including, but not limited to, routers, access servers, switches, bridges, hubs, IP telephones, IP video cameras, computer hosts, and printers. An agent is a network-management software module that resides on a managed device. An agent has local knowledge of management information and translates that information to or from an SNMP-specific form.

A network management station (NMS) executes applications that monitor and control managed devices. NMSs provide the bulk of the processing and memory resources required for network management. One or more NMSs may exist on any managed network.



All SNMP PDUs are constructed as follows:

IP header

UDP header

version

community

PDU-type

request-id

error-status

error-index

variable bindings

The seven SNMP protocol data unit (PDU) types are as follows:

GetRequest
A manager-to-agent request to retrieve the value of a variable or list of variables. Desired variables are specified in variable bindings (values are not used). Retrieval of the specified variable values is to be done as an atomic operation by the agent. A Response with current values is returned.

SetRequest
A manager-to-agent request to change the value of a variable or list of variables. Variable bindings are specified in the body of the request. Changes to all specified variables are to be made as an atomic operation by the agent. A Response with (current) new values for the variables is returned.

GetNextRequest
A manager-to-agent request to discover available variables and their values. Returns a Response with variable binding for the lexicographically next variable in the MIB. The entire MIB of an agent can be walked by iterative application of
GetNextRequest starting at OID 0. Rows of a table can be read by specifying column OIDs in the variable bindings of the request.
GetBulkRequest Optimized version of GetNextRequest. A manager-to-agent request for multiple iterations of GetNextRequest. Returns a Response with multiple variable bindings walked from the variable binding or bindings in the request. PDU specific non-repeaters and max-repetitions fields are used to control response behavior. GetBulkRequest was introduced in SNMPv2. *Response* Returns variable bindings and acknowledgement from agent to manager for GetRequest, SetRequest, GetNextRequest, GetBulkRequest and InformRequest. Error reporting is provided by error-status and error-index fields. Although it was used as a response to both gets and sets, this PDU was called GetResponse in SNMPv1. *Trap*
Asynchronous notification from agent to manager. SNMP traps enable an agent to notify the management station of significant events by way of an unsolicited SNMP message. Includes current sysUpTime value, an OID identifying the type of trap and optional variable bindings. Destination addressing for traps is determined in an application-
specific manner typically through trap configuration variables in the MIB. The format of the trap message was changed in SNMPv2 and the PDU was renamed SNMPv2-Trap. While in classic communication the client always actively requests information from the server, SNMP allows the additional use of so-called "traps". These are data packages that are sent from the SNMP client to the server without being explicitly requested. *InformRequest*
Acknowledged asynchronous notification. This PDU was introduced in SNMPv2 and was originally defined as manager to manager communication.[4] Later implementationshave loosened the original definition to allow agent to manager communications.[5][6][7]Manager-to-manager notifications were already possible in SNMPv1 (using a Trap), but as SNMP commonly runs over UDP where delivery is not assured and dropped packets are not reported, delivery of a Trap was not guaranteed. InformRequest fixes this by sending back an acknowledgement on receipt.

Electronic Mail: SMTP

Major application layer components

- user agent : the software you use to read, compose, and organize email
- mail server : the server that interacts with user agents and other mail servers to deliverand store email
- Snail-mail analogy: Each post office is a mail server, post boxes are user agents.
- Note that mail server acts as both client and server; client when it sends to another mailserver, server when it receives from another mail server
- Mail servers communicate using application layer protocol **SMTP**
- Note that email is "push" oriented, while Web is "pull" oriented

SMTP : Simple Mail Transfer Protocol

- Set of protocols used by mail servers to communicate with each other
- Defined in RFC 2821
- It is old (like 1970s old), and indeed very simple!
- Based on 7-bit ASCII codes (codes 0-127)
- Anything other than ASCII text has to be translated into ASCII by sending server, andback by receiver
- Not fun for transmitting multimedia!
- Basic protocol for message delivery

1.sender's user agent sends message (M) to sender's mail server (SMS)
2.SMS places M in its message queue
3.SMS attempts TCP connection with recipient's mail server (RMS)
4.If RMS valid but connection not possible, SMS backs off while and tries againlater
5.Once connected, SMS and RMS exchange pleasantries and SMS transmits M toRMS
6.RMS places M into recipient's mailbox, located in RMS.
7.Recipient accesses mailbox via user agent and reads M.

- Protocol takes place in plain text, see textbook example
- Client sends SMTP commands such as HELO, MAIL FROM, RCPT TO, DATA, QUIT
- Server responds to each command with one-liner containing response code and message
- It is possible to spoof being a sending mail server by using port 25 (gotta know the mailserver names)

SMTP mail message format

- Message starts with first character sent after receiving code 354 response to DATAcommand
- Message ends with period character '.' on a line by itself!
- Message consists of header and body, separated by a blank line.
- Header line syntax resembles HTTP header line: attribute then colon then value (e.g. To:PSanderson@otterbein.edu)
- Example header lines are: To:, From:, Subject:
- The header line values are the ones displayed by mail reader
- Note that header lines are not part of SMTP delivery protocol, so you can give them anyvalue!
- **MIME**, MultImedia Mail Extensions, are used to transmit non-ASCII content

oHeader lines describe the content type so receiver knows how to interpret thecontent

oThe Content-Transfer-Encoding: line tells receiver how original content wastranslated to ASCII, so it will know how to undo the translation

oThe Content-Type: line tells receiver what kind of content it is: HTML, JPEG,Word document, whatever

oReceiver's mail agent is responsible for rendering the content

Multiple MIME components can be included in the same message

- Non-text email message bodies and attachments are both handled this way
- For more info about MIME, see RFC 2045 and RFC 2046

Mail Access Protocols

- Delivery of received message from receiver's mail server to receiver's user agent is adifferent matter
- For one thing, this final delivery is a "pull" not a "push" operation and SMTP is push-only
- For another, authentication is of utmost importance
- Several such protocols have been defined: **POP3**, **IMAP**, **HTTP**-based
- Post Office Protocol (POP)

oThe original mail access protocol, see RFC 1939

oVery primitive; only 12 possible commands and little security

oServer only maintains in-box, any saved-message organization must be performedby the user agent (client) - very inflexible

oSession is established by TCP connection to port 110

oSession proceeds through several states:

- AUTHORIZATION - entered upon TCP connect, establish client identityas valid mailbox owner (name and password transmitted in plain text)
- TRANSACTION - entered upon authorization, interact with server toread/delete messages
- UPDATE - entered upon QUIT command, resource clean-up on serverside and TCP disconnect

oNo information state is maintained between sessions

- Internet Mail Access Protocol (IMAP)

oSignificantly more features, complexity, and security than POP; see see RFC3501

oServer maintains user-created folders for organizing and search for messages

oMaintains state, such as folder organizations, between sessions

oLike other protocols, uses plain-text commands from client and responses fromserver

oSession is based on TCP connection and has several states

oHere is state-transition diagram, copied directly from RFC 3501

(1) connection without pre-authentication (OK greeting)

(2) pre-authenticated connection (PREAUTH greeting)

(3) rejected connection (BYE greeting)

(4) successful LOGIN or AUTHENTICATE command

(5) successful SELECT or EXAMINE command

(6) CLOSE command, or failed SELECT or EXAMINE command

(7) LOGOUT command, server shutdown, or connection closed

Simple Mail Transfer Protocol (SMTP) •Defined in RFC 2821 (April 2001)–Original definition in RFC 821 (August 1982)•Designed for:–Direct transfer from the sender to the receiver(rather then go through a set of relays) –Destination system is always up and connected–Use TCP to transfer email from client to destination serverThe **SMTP (Simple Mail Transfer Protocol)** protocol is used by the Mail Transfer Agent (MTA) to deliver your eMail to the recipient's mail server. The SMTP protocol can only be used to send emails, not to receive them. Depending on your network / ISP settings, you may only be able to use the SMTP protocol under certain conditions

Simple Mail Transfer Protocol (**SMTP**) is an Internet standard for electronic mail (email) transmission. First defined by RFC 821 in 1982, it was last updated in 2008 with the Extended SMTP additions by RFC 5321 - which is the protocol in widespread use today.

SMTP by default uses TCP port 25. The protocol for mail submission is the same, but uses port 587. SMTP connections secured by SSL, known as SMTPS, default to port 465 (nonstandard, but sometimes used for legacy reasons).

Although electronic mail servers and other mail transfer agents use SMTP to send and receive mail messages, user-level client mail applications typically use SMTP only for sending messages to a mail server for relaying. For receiving messages, client applications usually use either POP3 or IMAP.

Although proprietary systems (such as Microsoft Exchange and IBM Notes) and webmail systems (such as Outlook.com, Gmail and Yahoo! Mail) use their own non-standard protocols to access mail box accounts on their own mail servers, all use SMTP when sending or receiving email from outside their own systems.

IMAP:

The **Internet Message Access Protocol** (**IMAP**) is an Internet standard protocol used by e-mail clients to retrieve e-mail messages from a mail server over a TCP/IP connection.IMAP is defined by RFC 3501.

IMAP was designed with the goal of permitting complete management of an email box by multiple email clients, Therefore, clients generally leave messages on the server until the user explicitly deletes them. An IMAP server typically listens on port number 143. IMAP over SSL (**IMAPS**) is assigned the port number 993.

Virtually all modern e-mail clients and servers support IMAP. IMAP and the earlier POP3 (Post Office Protocol) are the two most prevalent standard protocols for email retrieval, with many webmail service providers such as Gmail, Outlook.com and Yahoo! Mail also providing support for either IMAP or POP3.

IMAP was designed by Mark Crispin in 1986 as a remote mailbox protocol, in contrast to the widely used POP, a protocol for retrieving the contents of a mailbox.[5]

IMAP was previously known as **Internet Mail Access Protocol**, **Interactive Mail Access Protocol** (RFC 1064), and **Interim Mail Access Protocol**.[6]

IMAP stands for Internet Message Access Protocol. IMAP shares many similar features with POP3. It, too, is a protocol that an email client can use to download email from an email server. However, IMAP includes many more features than POP3. The IMAP protocol is designed to let users keep their email on the server. IMAP requires more disk space on the server and more CPU resources than POP3, as all emails are stored on the server. IMAP normally uses port 143. Here is more information about IMAP.

Original IMAP

The original Interim Mail Access Protocol was implemented as a Xerox Lisp machine client and a TOPS-20 server.

No copies of the original interim protocol specification or its software exist.[7] Althoughsome of its commands and responses were similar to IMAP2, the interim protocol lacked command/response tagging and thus its syntax was incompatible with all other versions of IMAP.

IMAP2

The interim protocol was quickly replaced by the Interactive Mail Access Protocol (IMAP2), defined in RFC 1064 (in 1988) and later updated by RFC 1176 (in 1990). IMAP2 introduced the command/response tagging and was the first publicly distributed version.

IMAP3

IMAP3 is an extremely rare variant of IMAP.[8] It was published as RFC 1203 in 1991. Itwas written specifically as a counter proposal to RFC 1176, which itself proposed modifications to IMAP2.[9] IMAP3 was never accepted by the marketplace.[10][11] The IESG reclassifiedRFC1203 "Interactive Mail Access Protocol - Version 3" as a Historic protocol in 1993. The IMAP Working Group used RFC1176 (IMAP2) rather than RFC1203 (IMAP3) as its starting point.[12][13]

IMAP2bis

With the advent of MIME, IMAP2 was extended to support MIME body structures and add mailbox management functionality (create, delete, rename, message upload) that was absent from IMAP2. This experimental revision was called IMAP2bis; its specification was never published in non-draft form. An internet draft of IMAP2bis was published by the IETF IMAP Working Group in October 1993. This draft was based upon the following earlier specifications: unpublished IMAP2bis.TXT document, RFC1176, and RFC1064 (IMAP2).[14] TheIMAP2bis.TXT draft documented the state of extensions to IMAP2 as of December 1992.[15]Early versions of Pine were widely distributed with IMAP2bis support[8] (Pine 4.00 and latersupports IMAP4rev1).

IMAP4

An IMAP Working Group formed in the IETF in the early 1990s took over responsibility for the IMAP2bis design. The IMAP WG decided to rename IMAP2bis to IMAP4 to avoid confusion

MIME:

Multipurpose Internet Mail Extensions (**MIME**) is an Internet standard that extends the format of email to support:

- Text in character sets other than ASCII
- Non-text attachments: audio, video, images, application programs etc.
- Message bodies with multiple parts
- Header information in non-ASCII character sets

Virtually all human-written Internet email and a fairly large proportion of automated email is transmitted via SMTP in MIME format.

MIME is specified in six linked RFC memoranda: RFC 2045, RFC 2046, RFC 2047, RFC 4288, RFC 4289 and RFC 2049; with the integration with SMTP email specified in detail in RFC 1521 and RFC 1522.
Although MIME was designed mainly for SMTP, the content types defined by MIME standards are also of importance outside of email, such as in communication protocols like HTTP for the World Wide Web. Servers insert the MIME header at the beginning of any Web transmission. Clients use this content type or Internet media type header to select an appropriate "player" application for the type of data the header indicates. Some of these players are built into the Web client or browser (for example, almost all browsers come with GIF and JPEG image players as well as the ability to handle HTML files);

MIME headers
MIME-Version
The presence of this header indicates the message is MIME-formatted. The value is typically "1.0" so this header appears as *MIME-Version: 1.0*
According to MIME co-creator Nathaniel Borenstein, the intention was to allow MIME to change, to advance to version 2.0 and so forth, but this decision led to the opposite outcome, making it nearly impossible to create a new version of the standard.
Content-Type
This header indicates the Internet media type of the message content, consisting of a type and subtype, for example

Content-Type: text/plain
Through the use of the multipart type, MIME allows mail messages to have parts arranged in a tree structure where the leaf nodes are any non-multipart content type and the non-leaf nodes are any of a variety of multipart types. This mechanism supports:
- **simple text messages using text/plain** (the default value for "Content-Type: ")
- text plus attachments (multipart/mixed with a text/plain part and other non-text parts).

AMIME message including an attached file generally indicates the file's original name withthe "Content-disposition:" header, so the type of file is indicated both by the MIMEcontent-type and the (usually OS-specific) filename extension

- reply with original attached (multipart/mixed with a text/plain part and the originalmessage as a message/rfc822 part)
- alternative content, such as a message sent in both plain text and another format such asHTML (multipart/alternative with the same content in text/plain and text/html forms)
- image, audio, video and application (for example, image/jpeg, audio/mp3, video/mp4,and application/msword and so on)
- many other message constructs

Content-Disposition

The original MIME specifications only described the structure of mail messages. They did not address the issue of presentation styles. The content-disposition header field was added in RFC 2183 to specify the presentation style. A MIME part can have:

- an inline content-disposition, which means that it should be automatically displayed whenthe message is displayed, or
- an attachment content-disposition, in which case it is not displayed automatically andrequires some form of action from the user to open it.

In addition to the presentation style, the content-disposition header also provides fields for specifying the name of the file, the creation date and modification date, which can be used by the reader's mail user agent to store the attachment.

In HTTP, the Content-Disposition: attachment response header is usually used to hint to the client to present the response body as a downloadable file. Typically, when receiving such a response, a Web browser will prompt the user to save its content as a file instead of displaying it as a page in a browser window, with the filename parameter suggesting the default file name (this is useful for dynamically generated content, where deriving the filename from the URL may be meaningless or confusing to the user).

Content-Transfer-Encoding

MIME (RFC 1341, since made obsolete by RFC 2045) defined a set of methods for representing binary data in formats other than ASCII text format. The content-transfer-encoding: MIME header has 2-sided significance:

- It indicates whether or not a binary-to-text encoding scheme has been used on top of theoriginal encoding as specified within the Content-Type header:

1.If such a binary-to-text encoding method has been used, it states which one.

2.If not, it provides a descriptive label for the format of content, with respect to thepresence of 8-bit or binary content.

The RFC and the IANA's list of transfer encodings define the values shown below, which are not case sensitive. Note that '7bit', '8bit', and 'binary' mean that no binary-to-text encoding on top of the original encoding was used. In these cases, the header is actually redundant for the email client to decode the message body, but it may still be useful as an indicator of what type of object is being sent. Values 'quoted-printable' and 'base64' tell the email client that a binary-to-text encoding scheme was used and that appropriate initial decoding is necessary before the message can be read with its original encoding (e.g. UTF-8).

- Suitable for use with normal SMTP:

o**7bit** – up to 998 octets per line of the code range 1..127 with CR and LF (codes13 and 10 respectively) only allowed to appear as part of a CRLF line ending.This is the default value.

o**quoted-printable** – used to encode arbitrary octet sequences into a form thatsatisfies the rules of 7bit. Designed to be efficient and mostly human readablewhen used for text data consisting primarily of US-ASCII characters but alsocontaining a small proportion of bytes with values outside that range.
o**base64** – used to encode arbitrary octet sequences into a form that satisfies therules of 7bit. Designed to be efficient for non-text 8 bit and binary data.Sometimes used for text data that frequently uses non-US-ASCII characters.
Suitable for use with SMTP servers that support the 8BITMIME SMTP extension (RFC6152):
o**8bit** – up to 998 octets per line with CR and LF (codes 13 and 10 respectively)only allowed to appear as part of a CRLF line ending.
Suitable for use with SMTP servers that support the BINARYMIME SMTP extension(RFC 3030):
o**binary** – any sequence of octets.

There is no encoding defined which is explicitly designed for sending arbitrary binary data through SMTP transports with the 8BITMIME extension. Thus, if BINARYMIME isn't supported, base64 or quoted-printable (with their associated inefficiency) are sometimes still useful. This restriction does not apply to other uses of MIME such as Web Services with MIME attachments or MTOM.
Multipurpose **I**nternet **M**ail **E**xtensions, a specification for formatting non-ASCII messages so that they can be sent over the Internet. Many e-mail clients now support MIME, which enables them to send and receive graphics, audio, and video files via the Internet mail system. In addition, MIME supports messages in character sets other than ASCII.
There are many predefined MIME types, such as GIF graphics files and PostScript files. It is also possible to define your own MIME types. In addition to e-mail applications, Web browsers also support various MIME types. This enables the browser to display or output files that are not in HTML format.
MIME (Multi-Purpose Internet Mail Extensions) is an extension of the original Internet e-mail protocol that lets people use the protocol to exchange different kinds of data files on the Internet: audio, video, images, application programs, and other kinds, as well as the ASCII text handled in the original protocol, the Simple Mail Transport Protocol (SMTP).
MIME (Multi-Purpose Internet Mail Extensions) is an extension of the original Internet e-mail protocol that lets people use the protocol to exchange different kinds of data files on the Internet: audio, video, images, application programs, and other kinds, as well as the ASCII text handled in the original protocol, the Simple Mail Transport Protocol (SMTP). In 1991, Nathan Borenstein of Bellcore proposed to the IETF that SMTP be extended so that Internet (but mainly Web) clients and servers could recognize and handle other kinds of data than ASCII text. As a result, new file types were added to "mail" as a supported Internet Protocol file type.
Servers insert the MIME header at the beginning of any Web transmission. Clients use this header to select an appropriate "player" application for the type of data the header indicates. Some of these players are built into the Web client or browser (for example, all browsers come with GIF and JPEG image players as well as the ability to handle HTML files); other players may need to be downloaded.
New MIME data types are registered with the Internet Assigned Numbers Authority (IANA).MIME is specified in detail in Internet Request for Comments 1521 and 1522, which amend the original mail protocol specification, RFC 821 (the Simple Mail Transport Protocol) and the ASCII messaging header, RFC 822.

Content Delivery Networks

Server farms and Web proxies help to build large sites and to improve Web performance, but they are not sufficient for truly popular Web sites that must serve content on a global scale. For these sites, a different approach is needed.

CDNs (**Content Delivery Networks**) turn the idea of traditional Web caching on its head. Instead, of having clients look for a copy of the requested page in a nearby cache, it is the provider who places a copy of the page in a set of nodes at different locations and directs the client to use a nearby node as the server. An example of the path that data follows when it is distributed by a CDN is shown in Fig. 7-67. It is a tree. The origin server in the CDN distributes a copy of the content to other nodes in the CDN, in Sydney, Boston, and Amsterdam, in this example. This is shown with dashed lines. Clients then fetch pages from the nearest node in the CDN. This is shown with solid lines. In this way, the clients in Sydney both fetch the page copy that is stored in Sydney; they do not both fetch
the page from the origin server, which may be in Europe.

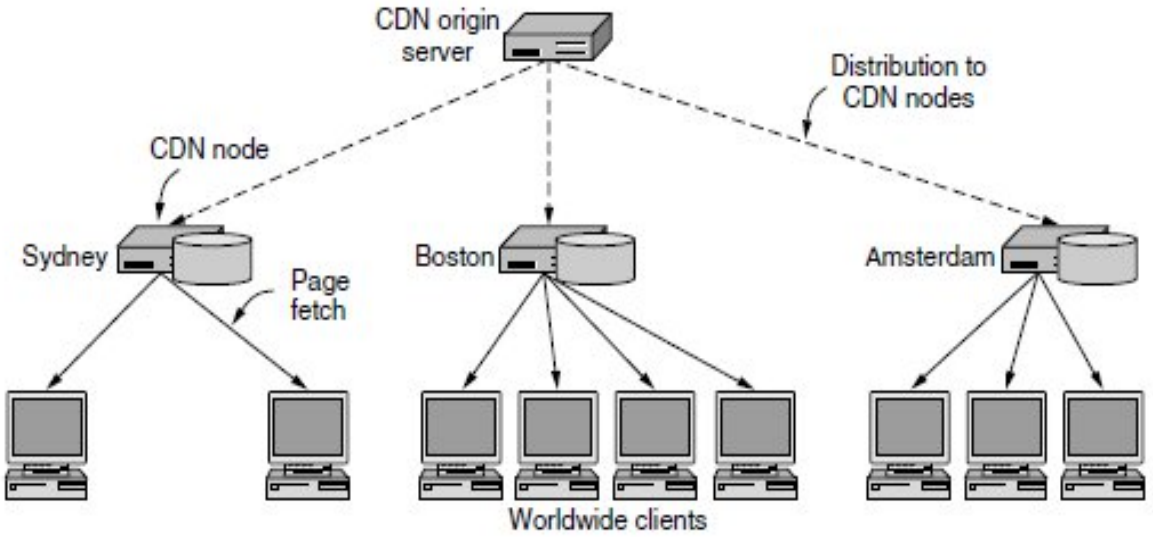


Figure 7-67. CDN distribution tree.

Using a tree structure has three virtues. First, the content distribution can be scaled up to as many clients as needed by using more nodes in the CDN, and more levels in the tree when the distribution among CDN nodes becomes the bottleneck. No matter how many clients there are, the tree structure is efficient. The origin server is not overloaded because it talks to the many clients via the tree of CDN nodes; it does not have to answer each request for a page by itself. Second, each client gets good performance by fetching pages from a nearby server instead of a distant server. This is because the round-trip time for setting up a connection is shorter, TCP slow-start ramps up more quickly because of the shorter round-trip time, and the shorter network path is less likely to pass through regions
of congestion in the Internet. Finally, the total load that is placed on the network is also kept at a minimum. If the CDN nodes are well placed, the traffic for a given page should pass over each part of the network only once. This is important because someone pays for network bandwidth, eventually.

The idea of using a distribution tree is straightforward. What is less simple is how to organize the clients to use this tree. For example, proxy servers would seem to provide a solution. Looking at Fig. 7-67, if each client was configured to use the Sydney, Boston or Amsterdam CDN node as a caching Web proxy, the distribution would follow the tree. However, this

strategy falls short in practice, for three reasons. The first reason is that the clients in a given part of the network probably belong to different organizations, so they are probably using different Web proxies. Recall that caches are not usually shared across organizations because of the limited benefit of caching over a large number of clients, and for security reasons too. Second, there can be multiple CDNs, but each client uses only
a single proxy cache. Which CDN should a client use as its proxy? Finally, perhaps the most practical issue of all is that Web proxies are configured by clients. They may or may not be configured to benefit content distribution by a CDN, and there is little that the CDN can do about it.

Another simple way to support a distribution tree with one level is to use **mirroring**. In this approach, the origin server replicates content over the CDN nodes as before. The CDN nodes in different network regions are called **mirrors**. The Web pages on the origin server contain explicit links to the different mirrors, usually telling the user their location. This design lets the user manually select a
nearby mirror to use for downloading content. A typical use of mirroring is to place a large software package on mirrors located in, for example, the East and West coasts of the U.S., Asia, and Europe. Mirrored sites are generally completely static, and the choice of sites remains stable for months or years. They are a tried and tested technique. However, they depend on the user to do the distribution
as the mirrors are really different Web sites, even if they are linked together.

The third approach, which overcomes the difficulties of the previous two approaches, uses DNS and is called **DNS redirection**. Suppose that a client wants to fetch a page with the URL *http://www.cdn.com/page.html*. To fetch the page, the browser will use DNS to resolve *www.cdn.com* to an IP address. This DNS lookup proceeds in the usual manner. By using the DNS protocol, the browser learns the IP address of the name server for *cdn.com*, then contacts the name server to ask it to resolve *www.cdn.com*. Now comes the really clever bit. The name server is run by the CDN. Instead, of returning the same IP address for each request, it will look at the IP address of the client making the request and return different answers. The answer will be the IP address of the CDN node that is nearest the client. That is, if a client in Sydney asks the CDN name server to resolve *www.cdn.com*, the name server will return the IP address of the Sydney CDN node, but if a client in Amsterdam makes the same request, the name server will return the IP address of the Amsterdam CDN node instead. This strategy is perfectly legal according to the semantics of DNS. We have previously seen that name servers may return changing lists of IP addresses. After the name resolution, the Sydney client will fetch the page directly from the Sydney CDN node. Further pages on the same ‘‘server’’ will be fetched directly from the Sydney CDN node as well because of DNS caching. The overall sequence of steps is shown in Fig. 7-68.

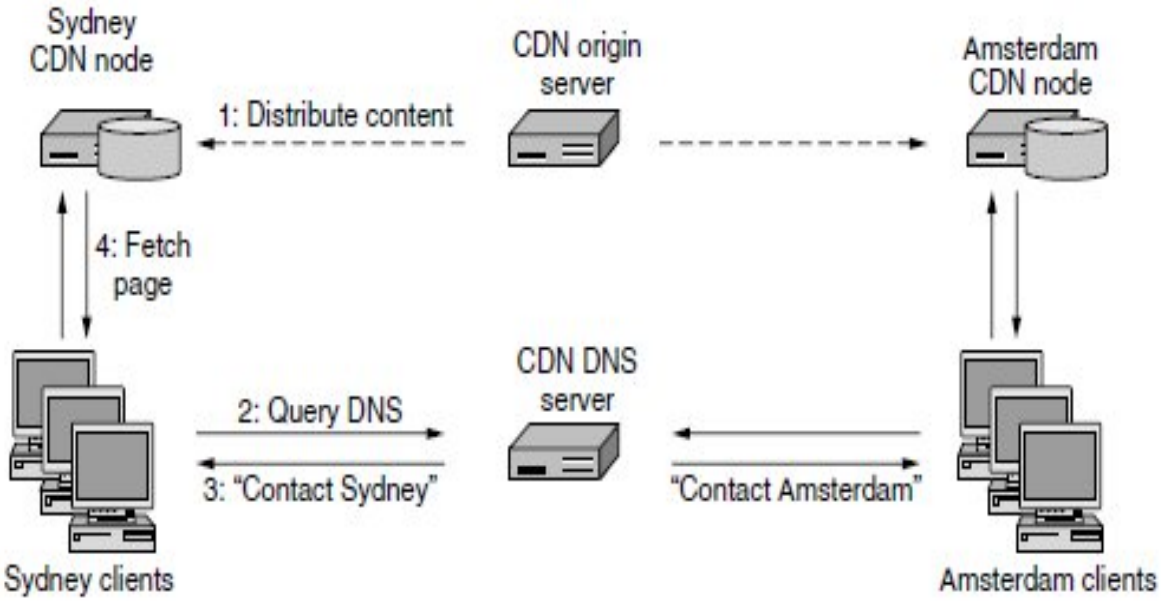


Figure 7-68. Directing clients to nearby CDN nodes using DNS.

A complex question in the above process is what it means to find the nearest CDN node, and how to go about it. To define nearest, it is not really geography that matters. There are at least two factors to consider in mapping a client to a CDN node. One factor is the network distance. The client should have a short and high-capacity network path to the CDN node. This situation will produce quick
downloads. CDNs use a map they have previously computed to translate between the IP address of a client and its network location. The CDN node that is selected might be the one at the shortest distance as the crow flies, or it might not. What matters is some combination of the length of the network path and any capacity limits along it. The second factor is the load that is already being carried by the CDN node. If the CDN nodes are overloaded, they will deliver slow responses,
just like the overloaded Web server that we sought to avoid in the first place. Thus, it may be necessary to balance the load across the CDN nodes, mapping some clients to nodes that are slightly further away but more lightly loaded.

The techniques for using DNS for content distribution were pioneered by Akamai starting in 1998, when the Web was groaning under the load of its early growth. Akamai was the first major CDN and became the industry leader. Probably even more clever than the idea of using DNS to connect clients to nearby nodes was the incentive structure of their business. Companies pay Akamai to deliver
their content to clients, so that they have responsive Web sites that customers like to use. The CDN nodes must be placed at network locations with good connectivity, which initially meant inside ISP networks. For the ISPs, there is a benefit to having a CDN node in their networks, namely that the CDN node cuts down the amount of upstream network bandwidth that they need (and must pay for), just as with proxy caches. In addition, the CDN node improves responsiveness for the ISP's customers, which makes the ISP look good in their eyes, giving them a competitive advantage over ISPs that do not have a CDN node. These benefits (at no cost to the ISP) makes installing a CDN node a no brainer for the ISP. Thus, the content

provider, the ISP, and the customers all benefit and the CDN makes money. Since 1998, other companies have gotten into the business, so it is now a
competitive industry with multiple providers.
As this description implies, most companies do not build their own CDN. Instead, they use the services of a CDN provider such as Akamai to actually deliver their content. To let other companies use the service of a CDN, we need to add one last step to our picture.
After the contract is signed for a CDN to distribute content on behalf of a Web site owner, the owner gives the CDN the content. This content is pushed to the CDN nodes. In addition, the owner rewrites any of its Web pages that link to the content. Instead of linking to the content on their Web site, the pages link to the content via the CDN. As an example of how this scheme works, consider the
source code for Fluffy Video's Web page, given in Fig. 7-69(a). After preprocessing, it is transformed to Fig. 7-69(b) and placed on Fluffy Video's server as *www.fluffyvideo.com/index.html*. When a user types in the URL *www.fluffyvideo.com* to his browser, DNS returns the IP address of Fluffy Video's own Web site, allowing the main (HTML) page to be fetched in the normal way. When the user clicks on any of the hyperlinks, the browser asks DNS to look up *www.cdn.com*. This lookup contacts the
CDN's DNS server, which returns the IP address of the nearby CDN node. The browser then sends a regular HTTP request to the CDN node, for example, for */fluffyvideo/koalas.mpg*. The URL identifies the page to return, starting the path with *fluffyvideo* so that the CDN node can separate requests for the different companies that it serves. Finally, the video is returned and the user sees cute fluffy animals.
The strategy behind this split of content hosted by the CDN and entry pages hosted by the content owner is that it gives the content owner control while letting the CDN move the bulk of the data. Most entry pages are tiny, being just HTML text. These pages often link to large files, such as videos and images. It is precisely these large files that are served by the CDN, even though the use of a CDN
is completely transparent to users. The site looks the same, but performs faster

PEER to PEER Networks

A peer-to-peer network is a simple network of computers. It first came into existence in the late 1970s. Here each computer acts as a node for file sharing within the formed network. Here each node acts as a server and thus there is no central server in the network. This allows the sharing of a huge amount of data. The tasks are equally divided amongst the nodes. Each node connected in the network shares an equal workload. For the network to stop working, all the nodes need to individually stop working. This is because each node works independently.

History of P2P Networks

Before the development of P2P, USENET came into existence in 1979. The network enabled the users to read and post messages. Unlike the forums we use today, it did not have a central server. It is used to copy the new messages to all the servers of the node.

- In the 1980s the first use of P2P networks occurred after personal computers were introduced.
- In August 1988, the internet relay chat was the first P2P network built to share text and chat.
- In June 1999, Napster was developed which was a file-sharing P2P software. It could be used to share audio files as well. This software was shut down due to

the illegal sharing of files. But the concept of network sharing i.e P2P became popular.

- In June 2000, Gnutella was the first decentralized P2P file sharing network. This allowed users to access files on other users' computers via a designated folder.

Types of P2P networks

1. **Unstructured P2P networks:** In this type of P2P network, each device is able to make an equal contribution. This network is easy to build as devices can be connected randomly in the network. But being unstructured, it becomes difficult to find content. For example, Napster, Gnutella, etc.
2. **Structured P2P networks:** It is designed using software that creates a virtual layer in order to put the nodes in a specific structure. These are not easy to set up but can give easy access to users to the content. For example, P-Grid, Kademlia, etc.
3. **Hybrid P2P networks:** It combines the features of both P2P networks and client-server architecture. An example of such a network is to find a node using the central server.

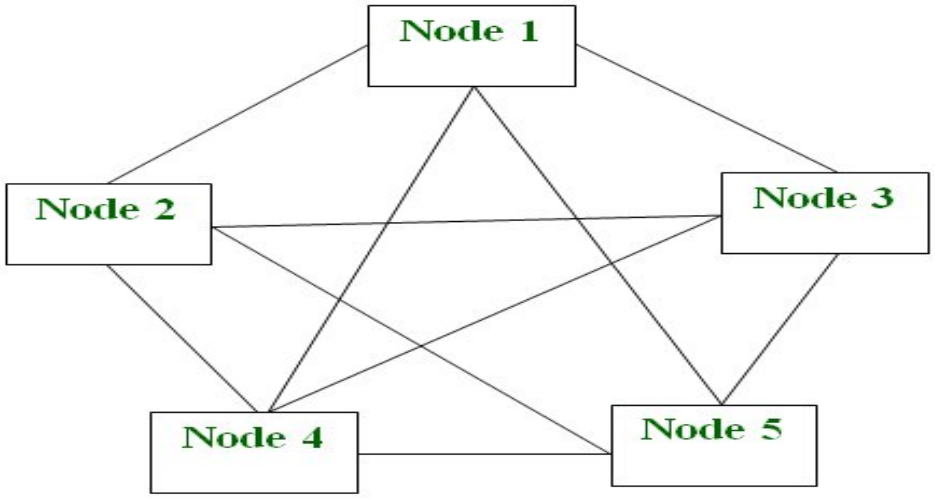
Features of P2P network

- These networks do not involve a large number of nodes, usually less than 12. All the computers in the network store their own data but this data is accessible by the group.
- Unlike client-server networks, P2P uses resources and also provides them. This results in additional resources if the number of nodes increases. It requires specialized software. It allows resource sharing among the network.
- Since the nodes act as clients and servers, there is a constant threat of attack.
- Almost all OS today support P2P networks.

P2P Network Architecture

In the P2P network architecture, the computers connect with each other in a workgroup to share files, and access to internet and printers.

- Each computer in the network has the same set of responsibilities and capabilities.
- Each device in the network serves as both a client and server.
- The architecture is useful in residential areas, small offices, or small companies where each computer act as an independent workstation and stores the data on its hard drive.
- Each computer in the network has the ability to share data with other computers in the network.
- The architecture is usually composed of workgroups of 12 or more computers.



P2P Architecture

- If the peer-to-peer software is not already installed, then the user first has to install the peer-to-peer software on his computer.
- This creates a virtual network of peer-to-peer application users.
- The user then downloads the file, which is received in bits that come from multiple computers in the network that have already that file.
- The data is also sent from the user's computer to other computers in the network that ask for the data that exist on the user's computer.

Thus, it can be said that in the peer-to-peer network the file transfer load is distributed among the peer computers.

Firstly secure your network via privacy solutions. Below are some of the measures to keep the P2P network secure:

- **Share and download legal files:** Double-check the files that are being downloaded before sharing them with other employees. It is very important to make sure that only legal files are downloaded.
- **Design strategy for sharing:** Design a strategy that suits the underlying architecture in order to manage applications and underlying data.
- **Keep security practices up-to-date:** Keep a check on the cyber security threats which might prevail in the network. Invest in good quality software that can sustain attacks and prevent the network from being exploited. Update your software regularly.
- **Scan all downloads:** This is used to constantly check and scan all the files for viruses before downloading them. This helps to ensure that safe files are being downloaded and in case, any file with potential threat is detected then report to the IT Staff.
- **Proper shutdown of P2P networking after use:** It is very important to correctly shut down the software to avoid unnecessary access to third persons to the files in the network. Even if the windows are closed after file sharing but the software is still active then the unauthorized user can still gain access to the network which can be a major security breach in the network.

Applications of P2P Network

Below are some of the common uses of P2P network:

- **File sharing:** P2P network is the most convenient, cost-efficient method for file sharing for businesses. Using this type of network there is no need for intermediate servers to transfer the file.
- **Blockchain:** The P2P architecture is based on the concept of decentralization. When a peer-to-peer network is enabled on the blockchain it helps in the maintenance of a complete replica of the records ensuring the accuracy of the data at the same time. At the same time, peer-to-peer networks ensure security also.
- **Direct messaging:** P2P network provides a secure, quick, and efficient way to communicate. This is possible due to the use of encryption at both the peers and access to easy messaging tools.
- **Collaboration:** The easy file sharing also helps to build collaboration among other peers in the network.
- **File sharing networks:** Many P2P file sharing networks like G2, and eDonkey have popularized peer-to-peer technologies.
- **Content distribution:** In a P2P network, unline the client-server system so the clients can both provide and use resources. Thus, the content serving capacity of the P2P networks can actually increase as more users begin to access the content.
- **IP Telephony:** Skype is one good example of a P2P application in VoIP.

Advantages of P2P Network

- **Easy to maintain:** The network is easy to maintain because each node is independent of the other.
- **Less costly:** Since each node acts as a server, therefore the cost of the central server is saved. Thus, there is no need to buy an expensive server.
- **No network manager:** In a P2P network since each node manages his or her own computer, thus there is no need for a network manager.
- **Adding nodes is easy:** Adding, deleting, and repairing nodes in this network is easy.
- **Less network traffic:** In a P2P network, there is less network traffic than in a client/ server network.

Disadvantages of P2P Network

- **Data is vulnerable:** Because of no central server, data is always vulnerable to getting lost because of no backup.
- **Less secure:** It becomes difficult to secure the complete network because each node is independent.
- **Slow performance:** In a P2P network, each computer is accessed by other computers in the network which slows down the performance of the user.
- **Files hard to locate:** In a P2P network, the files are not centrally stored, rather they are stored on individual computers which makes it difficult to locate the files.

Examples of P2P networks

P2P networks can be basically categorized into three levels.

- The first level is the basic level which uses a USB to create a P2P network between two systems.
- The second is the intermediate level which involves the usage of copper wires in order to connect more than two systems.
- The third is the advanced level which uses software to establish protocols in order to manage numerous devices across the internet.