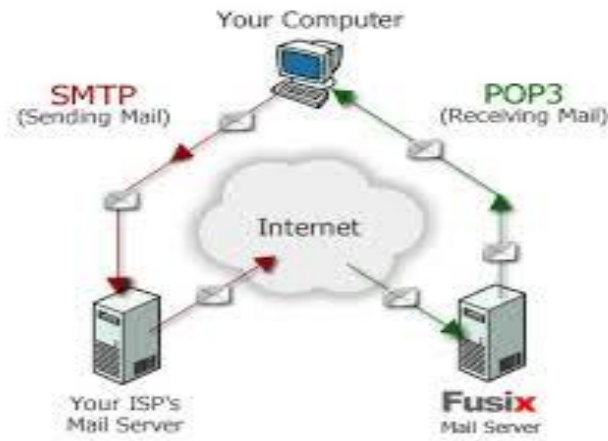<div align="center">

UNIT V

APPLICATION LAYER

</div>

Traditional Applications:

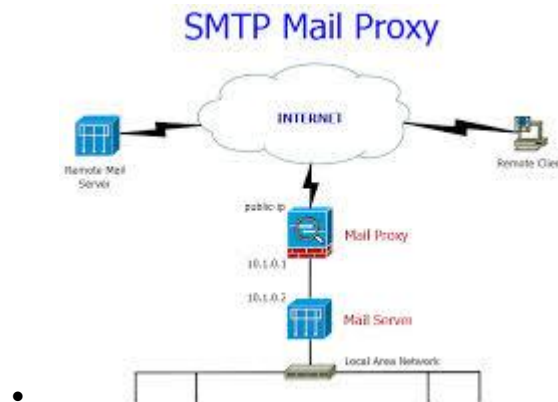

Electronic Mail: SMTP

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 deliver and 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 mail server, 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, and back 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 again later
5. Once connected, SMS and RMS exchange pleasantries and SMS transmits M to RMS
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 mail server names)
-

SMTP mail message format

- Message starts with first character sent after receiving code 354 response to DATA command
- 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 any value!
- **MIME**, MultImedia Mail Extensions, are used to transmit non-ASCII content
    o Header lines describe the content type so receiver knows how to interpret the content
    o The Content-Transfer-Encoding: line tells receiver how original content was translated to ASCII, so it will know how to undo the translation
    o The Content-Type: line tells receiver what kind of content it is: HTML, JPEG, Word document, whatever
    o Receiver'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 a different 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)
  - The original mail access protocol, see RFC 1939
  - Very primitive; only 12 possible commands and little security
  - Server only maintains in-box, any saved-message organization must be performed by the user agent (client) - very inflexible
  - Session is established by TCP connection to port 110
  - Session proceeds through several states:
    - AUTHORIZATION - entered upon TCP connect, establish client identity as valid mailbox owner (name and password transmitted in plain text)
    - TRANSACTION - entered upon authorization, interact with server to read/delete messages
    - UPDATE - entered upon QUIT command, resource clean-up on server side and TCP disconnect
  - No information state is maintained between sessions
- Internet Mail Access Protocol (IMAP)
  - Significantly more features, complexity, and security than POP; see see RFC 3501
  - Server maintains user-created folders for organizing and search for messages
  - Maintains state, such as folder organizations, between sessions
  - Like other protocols, uses plain-text commands from client and responses from server
  - Session is based on TCP connection and has several states
  - Here 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
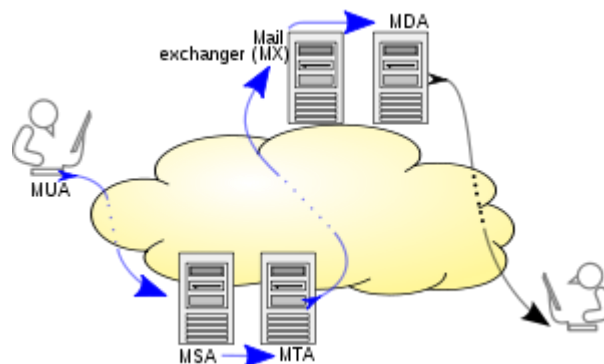–Use TCP to transfer email from client to destination server

The **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.

SMTP (Simple Mail Transfer Protocol) is a TCP/IP protocol used in sending and receiving e-mail.

SMTP (Simple Mail Transfer Protocol) is a TCP/IP protocol used in sending and receiving e-mail. However, since it is limited in its ability to queue messages at the receiving end, it is usually used with one of two other protocols, POP3 or IMAP, that let the user save messages in a server mailbox and download them periodically from the server. In other words, users typically use a program that uses SMTP for sending e-mail and either POP3 or IMAP for receiving e-mail. On Unix-based systems, sendmail is the most widely-used SMTP server for e-mail. A commercial package, Sendmail, includes a POP3 server. Microsoft Exchange includes an SMTP server and can also be set up to include POP3 support.

SMTP usually is implemented to operate over Internet port 25. An alternative to SMTP that is widely used in Europe is X.400. Many mail servers now support Extended Simple Mail Transfer Protocol (ESMTP), which allows multimedia files to be delivered as e-mail.

POP3:

The **Post Office Protocol** (**POP**) is an application-layer Internet standard protocol used by local e-mail clients to retrieve e-mail from a remote server over a TCP/IP connection.[1] POP has been developed through several versions, with version 3 (**POP3**) being the current standard.

Virtually all modern e-mail clients and servers support POP3, and it along with IMAP (Internet Message Access Protocol) are the two most prevalent Internet standard protocols for e-mail retrieval,[2] with many webmail service providers such as Gmail, Outlook.com and Yahoo! Mail also providing support for either IMAP or POP3 to allow mail to be downloaded.

POP supports simple download-and-delete requirements for access to remote mailboxes (termed mail drop in the POP RFC's).[3] Although most POP clients have an option to leave mail on server after download, e-mail clients using POP generally connect, retrieve all messages, store them on the user's PC as new messages, delete them from the server, and then disconnect. Other protocols, notably IMAP, (Internet Message Access Protocol) provide more complete and complex remote access to typical mailbox operations. In the late 1990s and early 2000s, fewer Internet Service Providers (ISPs) supported IMAP due to the storage space that was required on the ISP's hardware. Contemporary e-mail clients supported POP, then over time popular mail client software added IMAP support.

A POP3 server listens on well-known port 110. Encrypted communication for POP3 is either requested after protocol initiation, using the STLS command, if supported, or by POP3S, which connects to the server using Transport Layer Security (TLS) or Secure Sockets Layer (SSL) on well-known TCP port 995.

Available messages to the client are fixed when a POP session opens the maildrop, and are identified by message-number local to that session or, optionally, by a unique identifier assigned to the message by the POP server. This unique identifier is permanent and unique to the mail drop and allows a client to access the same message in different POP sessions. Mail is retrieved and marked for deletion by message-number. When the client exits the session, the mail marked for deletion is removed from the maildrop.

The **POP (Post Office Protocol 3)** protocol provides a simple, standardized way for users to access mailboxes and download messages to their computers.

When using the POP protocol all your eMail messages will be downloaded from the mail server to your local computer. You can choose to leave copies of your eMails on the server as well. The advantage is that once your messages are downloaded you can cut the internet connection and read your eMail at your leisure without incuring further communication costs. On the other hand you might have transferred a lot of message (including spam or viruses) in which you are not at all interested at this point.
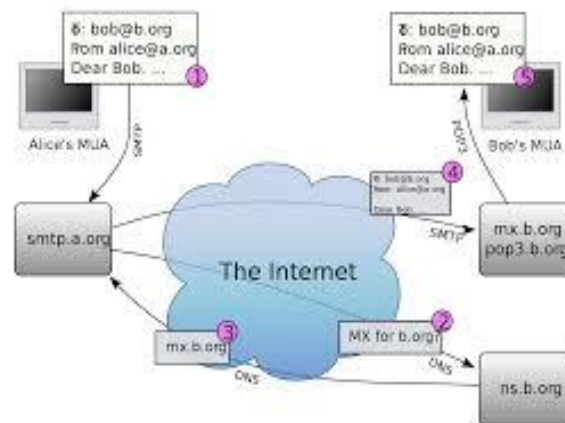
POP3 (Post Office Protocol 3) is the most recent version of a standard protocol for receiving e-mail.

POP3 (Post Office Protocol 3) is the most recent version of a standard protocol for receiving e-mail. POP3 is a client/server protocol in which e-mail is received and held for you by your Internet server. Periodically, you (or your client e-mail receiver) check your mail-box on the server and download any mail, probably using POP3. This standard protocol is built into most popular e-mail products, such as Eudora and Outlook Express. It's also built into the Netscape and Microsoft Internet Explorer browsers.

POP3 is designed to delete mail on the server as soon as the user has downloaded it. However, some implementations allow users or an administrator to specify that mail be saved for some period of time. POP can be thought of as a "store-and-forward" service.

An alternative protocol is Internet Message Access Protocol (IMAP). IMAP provides the user more capabilities for retaining e-mail on the server and for organizing it in folders on the server. IMAP can be thought of as a remote file server.

POP and IMAP deal with the receiving of e-mail and are not to be confused with the Simple Mail Transfer Protocol (SMTP), a protocol for transferring e-mail across the Internet. You send e-mail with SMTP and a mail handler receives it on your recipient's behalf. Then the mail is read using POP or IMAP.

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] Although some 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. It was 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 reclassified RFC1203 "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]
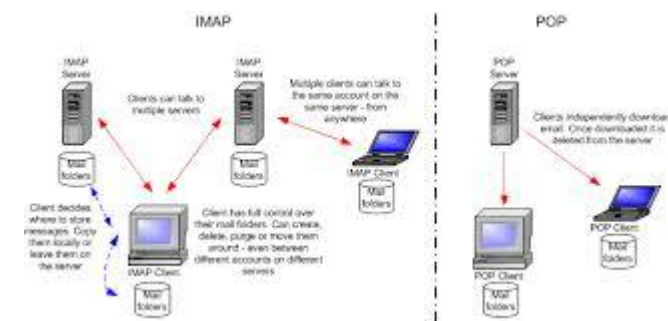
IMAP2bis

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] The IMAP2bis.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 later supports 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

Advantages over POP

Connected and disconnected modes of operation

When using POP, clients typically connect to the e-mail server briefly, only as long as it takes to download new messages. When using IMAP4, clients often stay connected as long as the user interface is active and download message content on demand. For users with many or large messages, this IMAP4 usage pattern can result in faster response times.

Multiple clients simultaneously connected to the same mailbox

The POP protocol requires the currently connected client to be the only client connected to the mailbox. In contrast, the IMAP protocol specifically allows simultaneous access by multiple clients and provides mechanisms for clients to detect changes made to the mailbox by other, concurrently connected, clients. See for example RFC3501 section 5.2 which specifically cites "simultaneous access to the same mailbox by multiple agents" as an example.

Access to MIME message parts and partial fetch

Usually all Internet e-mail is transmitted in MIME format, allowing messages to have a tree structure where the leaf nodes are any of a variety of single part content types and the non-leaf nodes are any of a variety of multipart types. The IMAP4 protocol allows clients to retrieve any of the individual MIME parts separately and also to retrieve portions of either individual parts or the entire message. These mechanisms allow clients to retrieve the text portion of a message without retrieving attached files or to stream content as it is being fetched.

Message state information

Through the use of flags defined in the IMAP4 protocol, clients can keep track of message state: for example, whether or not the message has been read, replied to, or deleted. These flags are stored on the server, so different clients accessing the same mailbox at different times can detect state changes made by other clients. POP provides no mechanism for clients to store such state information on the server so if a single user accesses a mailbox with two different POP clients (at different times), state information—such as whether a message has been accessed—cannot be synchronized between the clients. The IMAP4 protocol supports both predefined system flags and client-defined keywords.

System flags indicate state information such as whether a message has been read. Keywords, which are not supported by all IMAP servers, allow messages to be given one or more tags whose meaning is up to the client. IMAP keywords should not be confused with proprietary labels of web-based e-mail services which are sometimes translated into IMAP folders by the corresponding proprietary servers.

Multiple mailboxes on the server

IMAP4 clients can create, rename, and/or delete mailboxes (usually presented to the user as folders) on the server, and copy messages between mailboxes. Multiple mailbox support also allows servers to provide access to shared and public folders. The IMAP4 Access Control List (ACL) Extension (RFC 4314) may be used to regulate access rights.

Server-side searches

IMAP4 provides a mechanism for a client to ask the server to search for messages meeting a variety of criteria. This mechanism avoids requiring clients to download every message in the mailbox in order to perform these searches.

Built-in extension mechanism

Reflecting the experience of earlier Internet protocols, IMAP4 defines an explicit mechanism by which it may be extended. Many IMAP4 extensions to the base protocol have been proposed and are in common use. IMAP2bis did not have an extension mechanism, and POP now has one defined by RFC 2449.

Disadvantages

While IMAP remedies many of the shortcomings of POP, this inherently introduces additional complexity. Much of this complexity (e.g., multiple clients accessing the same mailbox at the same time) is compensated for by server-side workarounds such as Maildir or database backends.

The IMAP specification has been criticised for being insufficiently strict and allowing behaviours that effectively negate its usefulness. For instance, the specification states that each message stored on the server has a "unique id" to allow the clients to identify the messages they have already seen between sessions. However, the specification also allows these UIDs to be invalidated with no restrictions, practically defeating their purpose.[16]

Unless the mail storage and searching algorithms on the server are carefully implemented, a client can potentially consume large amounts of server resources when searching massive mailboxes.

IMAP4 clients need to maintain a TCP/IP connection to the IMAP server in order to be notified of the arrival of new mail. Notification of mail arrival is done through in-band signaling, which contributes to the complexity of client-side IMAP protocol handling somewhat.[17] A private proposal, push IMAP, would extend IMAP to implement push e-mail by sending the entire message instead of just a notification. However, push IMAP has not been generally accepted and current IETF work has addressed the problem in other ways (see the Lemonade Profile for more information).

Unlike some proprietary protocols which combine sending and retrieval operations, sending a message and saving a copy in a server-side folder with a base-level IMAP client requires transmitting the message content twice, once to SMTP for delivery and a second time to IMAP to store in a sent mail folder. This is remedied by a set of extensions defined by the IETF LEMONADE Working Group for mobile devices: URLAUTH (RFC 4467) and CATENATE (RFC 4469) in IMAP and BURL (RFC 4468) in SMTP-SUBMISSION. POP servers don't support server-side folders so clients have no choice but to store sent items on the client. Many IMAP clients can be configured to store sent mail in a client-side folder, or to BCC oneself and then filter the incoming mail instead of saving a copy in a folder directly. In addition to the LEMONADE "trio", Courier Mail Server offers a non-standard method of sending using IMAP by copying an outgoing message to a dedicated outbox folder.[18]

Lastly, support for message states can also cause problems for shared mailboxes; if a user on an IMAP client downloads and reads new mail from the server, the next user to download the same mail via an IMAP client as well will see the client automatically setting the status of said mail to 'Read' even though he or she has yet to do so. This can be a major issue since most IMAP clients do not provide notifications when read messages are downloaded; POP3 users do not experience such problems as the server does not store message states and as such will always have their messages delivered as new and unread mail.

IMAP (Internet Message Access Protocol) – Is a standard protocol for accessing e-mail from your local server. IMAP is a client/server protocol in which e-mail is received and held for you by your Internet server. As this requires only a small data transfer this works well even over a slow connection such as a modem. Only if you request to read a specific email message will it be downloaded from the server. You can also create and manipulate folders or mailboxes on the server, delete messages etc.
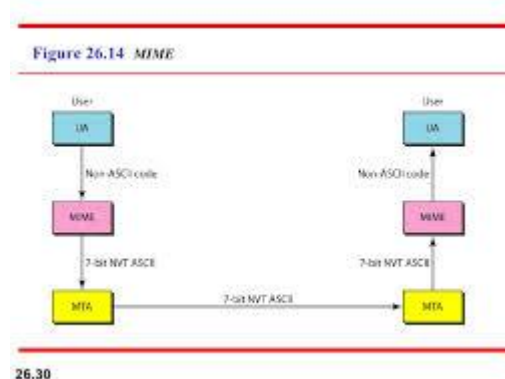
MIME:

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);

Figure 26.14 *MIME*

26.30

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). A MIME message including an attached file generally indicates the file's original name with the "Content-disposition:" header, so the type of file is indicated both by the MIME content-type and the (usually OS-specific) filename extension
- reply with original attached (multipart/mixed with a text/plain part and the original message as a message/rfc822 part)
- alternative content, such as a message sent in both plain text and another format such as HTML (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 when the message is displayed, or
- an attachment content-disposition, in which case it is not displayed automatically and requires 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 the original 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 the presence 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:
  - **7bit** – up to 998 octets per line of the code range 1..127 with CR and LF (codes 13 and 10 respectively) only allowed to appear as part of a CRLF line ending. This is the default value.
  - **quoted-printable** – used to encode arbitrary octet sequences into a form that satisfies the rules of 7bit. Designed to be efficient and mostly human readable when used for text data consisting primarily of US-ASCII characters but also containing a small proportion of bytes with values outside that range.
  - **base64** – used to encode arbitrary octet sequences into a form that satisfies the rules 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 (RFC 6152):
  - **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):
  - **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.

HTTP:

The **Hypertext Transfer Protocol** (**HTTP**) is an application protocol for distributed, collaborative, hypermedia information systems.[1] HTTP is the foundation of data communication for the World Wide Web.

Hypertext is structured text that uses logical links (hyperlinks) between nodes containing text. HTTP is the protocol to exchange or transfer hypertext.

The standards development of HTTP was coordinated by the Internet Engineering Task Force (IETF) and the World Wide Web Consortium (W3C), culminating in the publication of a series of Requests for Comments (RFCs). The first definition of HTTP/1.1, the version of HTTP in common use, occurred in RFC 2068

HTTP functions as a request-response protocol in the client-server computing model. A web browser, for example, may be the client and an application running on a computer hosting a web site may be the server. The client submits an HTTP request message to the server. The server, which provides resources such as HTML files and other content, or performs other functions on behalf of the client, returns a response message to the client. The response contains completion status information about the request and may also contain requested content in its message body.

A web browser is an example of a user agent (UA). Other types of user agent include the indexing software used by search providers (web crawlers), voice browsers, mobile apps, and other software that accesses, consumes, or displays web content.

HTTP is designed to permit intermediate network elements to improve or enable communications between clients and servers. High-traffic websites often benefit from web cache servers that deliver content on behalf of upstream servers to improve response time. Web browsers cache previously accessed web resources and reuse them when possible to reduce network traffic. HTTP proxy servers at private network boundaries can facilitate communication for clients without a globally routable address, by relaying messages with external servers.

HTTP is an application layer protocol designed within the framework of the Internet Protocol Suite. Its definition presumes an underlying and reliable transport layer protocol,[2] and Transmission Control Protocol (TCP) is commonly used. However HTTP can use unreliable protocols such as the User Datagram Protocol (UDP), for example in Simple Service Discovery Protocol (SSDP).

HTTP resources are identified and located on the network by Uniform Resource Locators (URLs), using the URI schemes http and https. URIs and hyperlinks in Hypertext Markup Language (HTML) documents form inter-linked hypertext documents.

HTTP/1.1 is a revision of the original HTTP (HTTP/1.0). In HTTP/1.0 a separate connection to the same server is made for every resource request. HTTP/1.1 can reuse a connection multiple times to download images, scripts, stylesheets, etc after the page has been delivered. HTTP/1.1 communications therefore experience less latency as the establishment of TCP connections presents considerable overhead.

HTTP session

An HTTP session is a sequence of network request-response transactions. An HTTP client initiates a request by establishing a Transmission Control Protocol (TCP) connection to a particular port on a server (typically port 80, occasionally port 8080; see List of TCP and UDP

port numbers). An HTTP server listening on that port waits for a client's request message. Upon receiving the request, the server sends back a status line, such as "HTTP/1.1 200 OK", and a message of its own. The body of this message is typically the requested resource, although an error message or other information may also be returned.[1]

HTTP Authentication

HTTP provides multiple authentication schemes such as Basic access authentication and Digest access authentication which operate via a challenge-response mechanism whereby the server identifies and issues a challenge before serving the requested content.

HTTP provides a general framework for access control and authentication, via an extensible set of challenge-response authentication schemes, which can be used by a server to challenge a client request and by a client to provide authentication information.[10]

Authentication Realms

The HTTP Authentication spec also provides an arbitrary, implementation specific construct for further dividing resources common to a given root URI. The realm value string, if present, is combined with the canonical root URI to form the protection space component of the challenge. This in effect allows the server to define separate authentication scopes under one root URI[10]

Request methods

An HTTP 1.1 request made using telnet. The request message, response header section, and response body are highlighted.

HTTP defines methods (sometimes referred to as verbs) to indicate the desired action to be performed on the identified resource. What this resource represents, whether pre-existing data or data that is generated dynamically, depends on the implementation of the server. Often, the resource corresponds to a file or the output of an executable residing on the server. The HTTP/1.0 specification[11] defined the GET, POST and HEAD methods and the HTTP/1.1 specification[12] added 5 new methods: OPTIONS, PUT, DELETE, TRACE and CONNECT. By being specified in these documents their semantics are well known and can be depended upon.

Any client can use any method and the server can be configured to support any combination of methods. If a method is unknown to an intermediate it will be treated as an unsafe and non-idempotent method. There is no limit to the number of methods that can be defined and this allows for future methods to be specified without breaking existing infrastructure. For example, WebDAV defined 7 new methods and RFC 5789 specified the PATCH method.

GET

The GET method requests a representation of the specified resource. Requests using GET should only retrieve data and should have no other effect. (This is also true of some other HTTP methods.)[1] The W3C has published guidance principles on this distinction, saying, "Web application design should be informed by the above principles, but also by the relevant limitations."[13] See safe methods below.

HEAD

The HEAD method asks for a response identical that of a GET request, but without the response body. This is useful for retrieving meta-information written in response headers, without having to transport the entire content.

POST

The POST method requests that the server accept the entity enclosed in the request as a new subordinate of the web resource identified by the URI. The data POSTed might be, for example, an annotation for existing resources; a message for a bulletin board, newsgroup, mailing list, or comment thread; a block of data that is the result of submitting a web form to a data-handling process; or an item to add to a database.[14]

PUT

The PUT method requests that the enclosed entity be stored under the supplied URI. If the URI refers to an already existing resource, it is modified; if the URI does not point to an existing resource, then the server can create the resource with that URI.[15]

DELETE

The DELETE method deletes the specified resource.

TRACE

The TRACE method echoes the received request so that a client can see what (if any) changes or additions have been made by intermediate servers.

OPTIONS

The OPTIONS method returns the HTTP methods that the server supports for the specified URL. This can be used to check the functionality of a web server by requesting '*' instead of a specific resource.

CONNECT

[16] The CONNECT method converts the request connection to a transparent TCP/IP tunnel, usually to facilitate SSL-encrypted communication (HTTPS) through an unencrypted HTTP proxy.[17][18] See HTTP CONNECT tunneling.

PATCH

The PATCH method applies partial modifications to a resource.[19]

All general-purpose HTTP servers are required to implement at least the GET and HEAD methods,[20] and, whenever possible, also the OPTIONS method.[citation needed]

Safe methods

Some of the methods (for example, HEAD, GET, OPTIONS and TRACE) are, by convention, defined as safe, which means they are intended only for information retrieval and should not change the state of the server. In other words, they should not have side effects, beyond relatively harmless effects such as logging, caching, the serving of banner advertisements or incrementing a web counter. Making arbitrary GET requests without regard to the context of the application's state should therefore be considered safe. However, this is not mandated by the standard, and it is explicitly acknowledged that it cannot be guaranteed.

By contrast, methods such as POST, PUT, DELETE and PATCH are intended for actions that may cause side effects either on the server, or external side effects such as financial transactions or transmission of email. Such methods are therefore not usually used by conforming web robots or web crawlers; some that do not conform tend to make requests without regard to context or consequences.

Despite the prescribed safety of GET requests, in practice their handling by the server is not technically limited in any way. Therefore, careless or deliberate programming can cause non-trivial changes on the server. This is discouraged, because it can cause problems for web caching, search engines and other automated agents, which can make unintended changes on the server.

Idempotent methods and web applications

Methods PUT and DELETE are defined to be idempotent, meaning that multiple identical requests should have the same effect as a single request (note that idempotence refers to the state of the system after the request has completed, so while the action the server takes (e.g. deleting a record) or the response code it returns may be different on subsequent requests, the system state will be the same every time). Methods GET, HEAD, OPTIONS and TRACE, being prescribed as safe, should also be idempotent, as HTTP is a stateless protocol.

In contrast, the POST method is not necessarily idempotent, and therefore sending an identical POST request multiple times may further affect state or cause further side effects (such as financial transactions). In some cases this may be desirable, but in other cases this could be due to an accident, such as when a user does not realize that their action will result in sending another request, or they did not receive adequate feedback that their first request was successful. While web browsers may show alert dialog boxes to warn users in some cases where reloading a page may re-submit a POST request, it is generally up to the web application to handle cases where a POST request should not be submitted more than once. Note that whether a method is idempotent is not enforced by the protocol or web server. It is perfectly possible to write a web application in which (for example) a database insert or other non-idempotent action is triggered by a GET or other request. Ignoring this recommendation, however, may result in undesirable consequences, if a user agent assumes that repeating the same request is safe when it isn't.

Security

Implementing methods such as TRACE, TRACK and DEBUG are considered potentially insecure by some security professionals because attackers can use them to gather information or bypass security controls during attacks. Security software tools such as Tenable Nessus and Microsoft UrlScan Security Tool report on the presence of these methods as being security issues.[21] TRACK and DEBUG are not valid HTTP 1.1 verbs.[22]

Web Services:

A **Web service** is a method of communication between two electronic devices over a network. It is a software function provided at a network address over the Web with the service always on as in the concept of utility computing.

The term Web services describes a standardized way of integrating Web-based applications using the XML, SOAP, WSDL and UDDI open standards over an Internet protocol backbone. XML is used to tag the data, SOAP is used to transfer the data, WSDL is used for describing the services available and UDDI is used for listing what services are available. Used primarily as a means for businesses to communicate with each other and with clients, Web services allow organizations to communicate data without intimate knowledge of each other's IT systems behind the firewall.

Unlike traditional client/server models, such as a Web server/Web page system, Web services do not provide the user with a GUI. Web services instead share business logic, data and processes through a programmatic interface across a network. The applications interface, not the users. Developers can then add the Web service to a GUI (such as a Web page or an executable program) to offer specific functionality to users.

Web services allow different applications from different sources to communicate with each other without time-consuming custom coding, and because all communication is in XML, Web services are not tied to any one operating system or programming language. For example, Java can talk with Perl, Windows applications can talk with UNIX applications.

- A web service is any piece of software that makes itself available over the internet and uses a standardized XML messaging system. XML is used to encode all communications to a web service. For example, a client invokes a web service by sending an XML message, then waits for a corresponding XML response. As all communication is in XML, web services are not tied to any one operating system or programming language-- Java can talk with Perl; Windows applications can talk with Unix applications.
- Web services are self-contained, modular, distributed, dynamic applications that can be described, published, located, or invoked over the network to create products, processes, and supply chains. These applications can be local, distributed, or web-based. Web services are built on top of open standards such as TCP/IP, HTTP, Java, HTML, and XML.
- Web services are XML-based information exchange systems that use the Internet for direct application-to-application interaction. These systems can include programs, objects, messages, or documents.
- A web service is a collection of open protocols and standards used for exchanging data between applications or systems. Software applications written in various programming languages and running on various platforms can use web services to exchange data over computer networks like the Internet in a manner similar to inter-process communication on a single computer. This interoperability (e.g., between Java and Python, or Windows and Linux applications) is due to the use of open standards.

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.
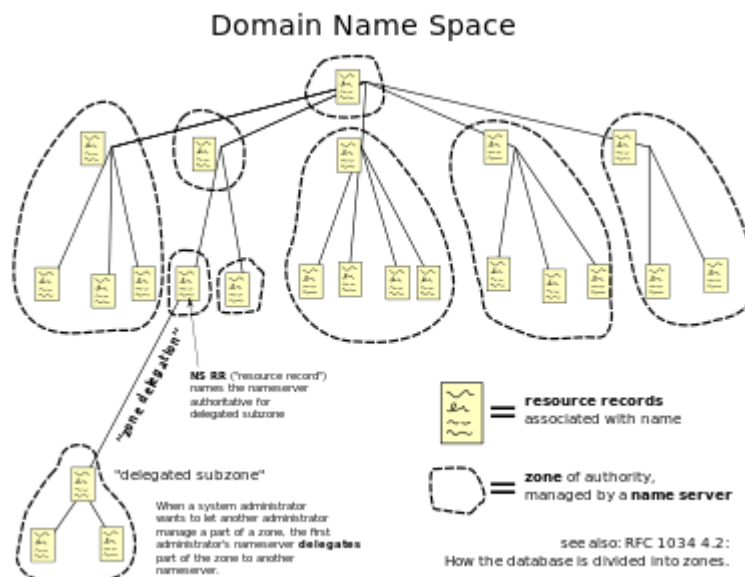
The Domain Name System distributes the responsibility of assigning domain names and mapping those names to IP addresses by designating authoritative name servers for each domain. Authoritative name servers are assigned to be responsible for their supported domains, and may delegate authority over sub-domains 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[1] and the Internet Protocol (IP) address spaces.[2] The Domain Name System maintains the domain name hierarchy 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 name server 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.[1] In the internal binary representation of the DNS the maximum length requires 255 octets of storage, since it also stores the length of the name.[3]

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.[8] Labels may not start or end with a hyphen.[9] An additional rule requires that top-level domain names should not be all-numeric.[9]

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.
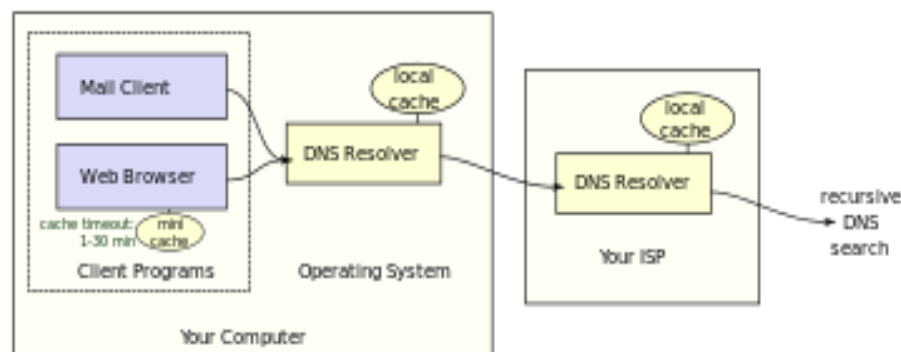
An authoritative name server can either be a master server or a slave server. A master server is a server that stores the original (master) copies of all zone records. A slave server uses an automatic updating mechanism of the DNS protocol in communication with its master to maintain an identical copy 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] The designation 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.
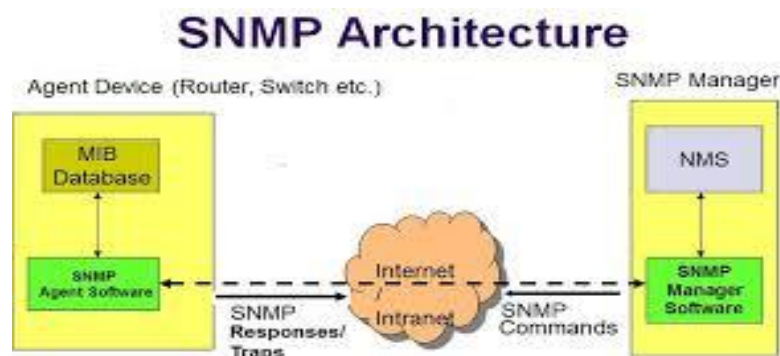
Domain **N**ame **S**ystem (or **S**ervice or **S**erver), an Internet service that translates domain names into IP addresses. Because domain names are alphabetic, they're easier to remember. The Internet however, is really based on IP addresses. Every time you use a domain name, therefore,aDNS service must translate the name into the corresponding IP address.

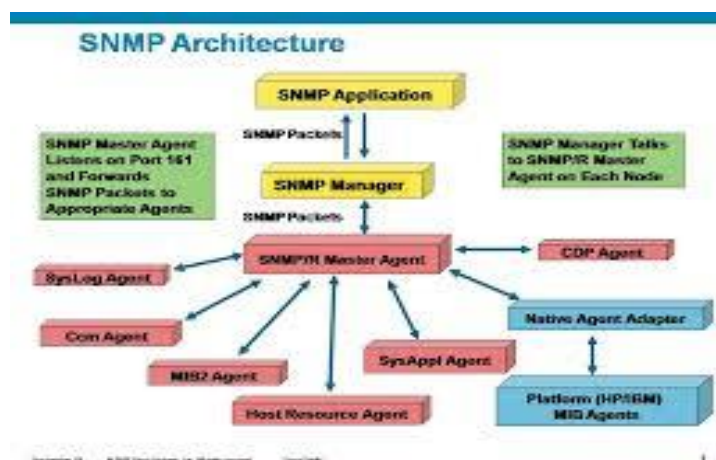For example, the domain name www.example.com might translate to 198.105.232.4.

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

CS 6551 Computer Networks – Unit- V   Page 25

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).

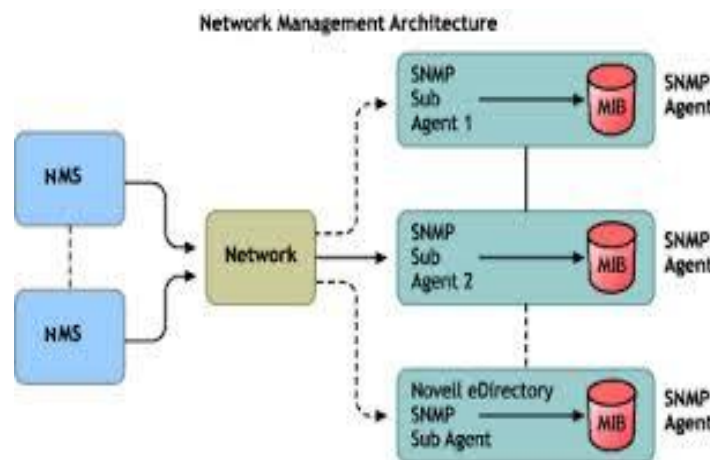An SNMP-managed network consists of three key components:

- 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.

Network Management Architecture

SNMP operates in the Application Layer of the Internet Protocol Suite (Layer 7 of the OSI model). The SNMP agent receives requests on UDP port 161. The manager may send requests from any available source port to port 161 in the agent. The agent response will be sent back to the source port on the manager. The manager receives notifications (Traps and InformRequests) on port 162. The agent may generate notifications from any available port. When used with Transport Layer Security or Datagram Transport Layer Security requests are received on port 10161 and traps are sent to port 10162.

SNMPv1 specifies five core protocol data units (PDUs). Two other PDUs, GetBulkRequest and InformRequest were added in SNMPv2 and the Report PDU was added in SNMPv3.

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 implementations have 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.