Email Protocols

Today, email is delivered using a client/server architecture. An email message is created using a mail client program. This program then sends the message to a server. The server then forwards the message to the recipient's email server, where the message is then supplied to the recipient's email client.

To enable this process, a variety of standard network protocols allow different machines, often running different operating systems and using different email programs, to send and receive email.

The following protocols discussed are the most commonly used in the transfer of email.

11.1.1. Mail Transport Protocols

Mail delivery from a client application to the server, and from an originating server to the destination server, is handled by the *Simple Mail Transfer Protocol* (*SMTP*) .

11.1.1.1. SMTP

The primary purpose of SMTP is to transfer email between mail servers. However, it is critical for email clients as well. To send email, the client sends the message to an outgoing mail server, which in turn contacts the destination mail server for delivery. For this reason, it is necessary to specify an SMTP server when configuring an email client.

Under Red Hat Enterprise Linux, a user can configure an SMTP server on the local machine to handle mail delivery. However, it is also possible to configure remote SMTP servers for outgoing mail.

One important point to make about the SMTP protocol is that it does not require authentication. This allows anyone on the Internet to send email to anyone else or even to large groups of people. It is this characteristic of SMTP that makes junk email or *spam* possible. Modern SMTP servers attempt to minimize this behavior by allowing only known hosts access to the SMTP server. Those servers that do not impose such restrictions are called *open relay* servers.

By default, Sendmail (/usr/sbin/sendmail) is the default SMTP program under Red Hat Enterprise Linux. However, a simpler mail server application called Postfix (/usr/sbin/postfix) is also available.

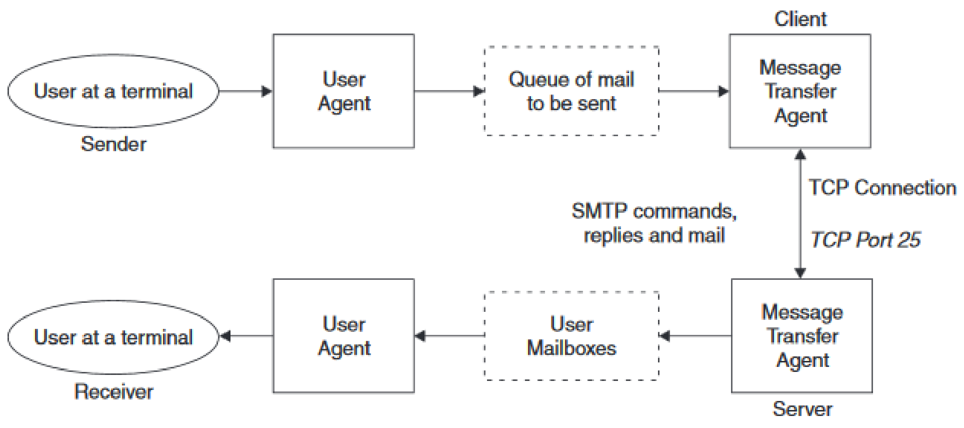

Figure 1: The basic simple mail transfer protocol (SMTP) model.

11.1.2. Mail Access Protocols

There are two primary protocols used by email client applications to retrieve email from mail servers: the *Post Office Protocol* (*POP*) and the *Internet Message Access Protocol* (*IMAP*).

Unlike SMTP, both of these protocols require connecting clients to authenticate using a username and password. By default, passwords for both protocols are passed over the network unencrypted.

A mail access protocol operates in three common modes that differ in where and how a user stores and
processes his or her mail.
Offline mode —e-mail is downloaded from a temporary storage on the mail server to the user's computer. After download, the mail is deleted from the server.
Online mode —user's e-mail, his or her inbox, and all filed mail remains permanently on the mail server. By connecting to the server and establishing an e-mail session, the user can download a temporary copy of his or her e-mail and read it, or send e-mail. Once the connection is finished, the copy is erased from user's computer, and only the original remains on the server.

Disconnected/resynchronization mode —combines both offline and online modes. A copy of the user’s
e-mail is downloaded to his or her computer(s), and the original message remains on the mail server. The
user can change a local copy of his or her e-mail on any computer, then resynchronize all copies, including the original e-mail message on the server and copies on additional computers.

11.1.2.1. POP

The default POP server under Red Hat Enterprise Linux is /usr/sbin/ipop3d and is provided by the imap package. When using a POP server, email messages are downloaded by email client applications. By default, most POP email clients are automatically configured to delete the message on the email server after it has been successfully transferred, however this setting usually can be changed.

POP is fully compatible with important Internet messaging standards, such as *Multipurpose Internet Mail Extensions* (*MIME*), which allow for email attachments.

POP works best for users who have one system on which to read email. It also works well for users who do not have a persistent connection to the Internet or the network containing the mail server. Unfortunately for those with slow network connections, POP requires client programs upon authentication to download the entire content of each message. This can take a long time if any messages have large attachments.

The most current version of the standard POP protocol is POP3.

There are, however a variety of lesser-used POP protocol variants:

- *APOP* — POP3 with MDS authentication. An encoded hash of the user's password is sent from the email client to the server rather then sending an unencrypted password.
- *KPOP* — POP3 with Kerberos authentication. Refer to Chapter 18 *Kerberos* for more information about Kerberos.

- *RPOP* — POP3 with RPOP authentication. This uses a per-user ID, similar to a password, to authenticate POP requests. However, this ID is not encrypted, so RPOP is no more secure than standard POP.

For added security, it is possible to use *Secure Socket Layer* (*SSL*) encryption for client authentication and data transfer sessions. This can be enabled by using the ipop3s service or by using the /usr/sbin/stunnel program.

11.1.2.2. IMAP

The default IMAP server under Red Hat Enterprise Linux is /usr/sbin/imapd and is provided by the imap package. When using an IMAP mail server, email messages remain on the server where users can read or delete them. IMAP also allows client applications to create, rename, or delete mail directories on the server to organize and store email.

IMAP is particularly useful for those who access their email using multiple machines. The protocol is also convenient for users connecting to the mail server via a slow connection, because only the email header information is downloaded for messages until opened, saving bandwidth. The user also has the ability to delete messages without viewing or downloading them.

Email standards

Email structure

These RFCs define the way emails themselves are structured.

- RFC 5322 — Internet Message Format (basic format of an email message), previously RFC 822 and RFC 2822.
- RFC 2045 — Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies (extension to the email message format to support attachments and non-ASCII data).
- RFC 2046 — Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types.
- RFC 2047 — MIME (Multipurpose Internet Mail Extensions) Part Three: Message Header Extensions for Non-ASCII Text.

- RFC 2231 — MIME Parameter Value and Encoded Word Extensions: Character Sets, Languages, and Continuations

Email protocols

These RFCs define how emails are transported between computers, both for sending (SMTP) and receiving (IMAP/POP).

- RFC 5321 — Simple Mail Transfer Protocol (protocol used to send emails between computers), previously RFC 821 and RFC 2821.
- RFC 3501 — INTERNET MESSAGE ACCESS PROTOCOL — VERSION 4rev1 (IMAP protocol, used to read emails).
- RFC 4551 — IMAP Extension for Conditional STORE Operation or Quick Flag Changes Resynchronization (IMAP extension that adds MODSEQ as a way to quickly find changes to a mailbox)
- RFC 1939 — Post Office Protocol, Version 3 (older POP protocol, used to read emails).

Email security

These RFCs define some security standards for email protocols and formats.

- RFC 2595 — Using TLS with IMAP, POP3 and ACAP (protocol used to upgrade a plaintext IMAP/POP connection to an SSL/TLS encrypted one).
- RFC 3207 — SMTP Service Extension for Secure SMTP over Transport Layer Security (protocol used to upgrade a plaintext SMTP connection to an SSL/TLS encrypted one).
- RFC 5246 — The Transport Layer Security (TLS) Protocol Version 1.2 (protocol used to encrypt a connection).
- RFC 6376 — DomainKeys Identified Mail (DKIM) Signatures (allows emails to be signed by a particular domain to ensure they haven't been tampered with, and to say that that domain claims responsibility for the message).

Service discovery

Email software that wants to access a user's email account has to know the server(s) to connect to. This used to be manually configured, but nowadays is often done using the user's email address through a service discovery process.

- Thunderbird/Autoconfiguration --- This is Mozillas custom approach that Thunderbird uses to auto-discover servers.
- RFC 6186 — Use of SRV Records for Locating Email Submission/Access Services (a standard that no-one seems to use yet, pity, it looks reasonable).

Note that some software vendors seem to maintain their own database of email domain → server name definitions to support auto-configuration in their email clients. These seem to be custom databases maintained by each software vendor separately.

Filtering

- RFC 5228 — Sieve: An Email Filtering Language (language used to file/filter/forward emails, what our Rules system ultimately generates. More information about sieve and sieve extensions. Note that our server doesn't support all the extensions.

NNTP (Network News Transfer Protocol)

NNTP (Network News Transfer Protocol) is the predominant protocol used by computer clients and servers for managing the notes posted on Usenet newsgroups.

The Network News Transfer Protocol is an Internet application protocol used for transporting Usenet news articles (Netnews) between news servers and for reading and posting articles by end user client applications.

NNTP replaced the original Usenet protocol, UNIX-to-UNIX Copy Protocol (UUCP) some time ago. NNTP servers manage the global network of collected Usenet newsgroups and include the server at your Internet access provider. An NNTP client is included as part of a Netscape, Internet Explorer, Opera, or other Web browser or you may use a separate client program called a newsreader.

Usenet was originally designed based on the UUCP network, with most article transfers taking place over direct point-to-point telephone links between news servers, which were powerful timesharing systems. Readers and posters logged into these computers reading the articles directly from the local disk.