Hoe werkt het SMTP Protocol? ..

Wanneer je gebruik wil maken van een email adres van een ander dan kan je dat via het smtp protocol doen. Waarom via SMTP ?. Je wilt dit via SMTP doen omdat SMTP voor Simple Mail Transfer Protocol staat.
Simple betekend dat wanneer je een email wilt verzenden, dit zo simpel is dat er geen user verificatie plaats vindt. SMTP gaat er namelijk vanuit dat alleen ISP Mailservers gebruik maken van dit protocol en niet andere gebruikers (Hackers). Hoe werkt het SMTP Protocol ?.

SMTP werkt altijd op TCP Poort 25, wanneer je Telnet opstart en kiest voor verbinden met SMTP.WXS.NL naar Port 25 dan verschijnt het volgende in beeld.

―――――――

.. Een voorbeeld van mailen via telnet en de SMTP-server ..

220 smtp01.wxs.nl ESMTP server (Netscape Messaging Server - Version 3.6)ready Mon, 26 Oct 1998 20:13:40 +0100
Geef nu het commando helo in met als domain naam het te gebruiken account

helo wxs.nl <Enter>

De mailserver antwoordt met:
250 smtp01.wxs.nl

Geef nu in wie je zelf bent (Mag iedere willekeurige naam zijn)

mail from:test@test.nl <Enter>

De mailserver antwoordt met:
250 Sender <test@test.nl> Ok

Geef nu op waarnaar toe

rcpt to:xxx@wxs.nl <Enter>

De mailserver antwoordt met:
250 Recipient <xxx@wxs.nl> Ok

Geef nu in dat je data wil versturen

data <Enter>

De mailserver antwoordt met:
354 Ok Send data ending with <CRLF>.<CRLF> **Dit is een voorbeeld.**
.
250 Message received: 7718A05B53C.AAA36D1

Nu wordt mail verstuurd.
Voor het anoniem versturen doe je alles wat hierboven staat, maar dan via een shellaccount.

Every day, the citizens of the Internet send each other billions of e-mail messages. If you are online a lot, you yourself may send a dozen or more e-mails each day without even thinking about it. Obviously, e-mail has become an extremely popular communication tool.

Have you ever wondered how e-mail gets from your desktop to a friend halfway around the world? What is a POP3 server, and how does it hold your mail? The answers may surprise you, because it turns out that e-mail is an incredibly simple system at its core! In this article, we'll take an in-depth look at e-mail and how it works!

An E-mail Message

According to Darwin Magazine: Prime Movers, the first e-mail message was sent in 1971 by an engineer named Ray Tomlinson. Prior to this, you could only send messages to users on a single machine. Tomlinson's breakthrough was the ability to send messages to other machines on the Internet, using the @ sign to designate the receiving machine.

An e-mail message has always been nothing more than a simple text message -- a piece of text sent to a recipient. In the beginning and even today, e-mail messages tend to be short pieces of text, although the ability to add attachments now makes many e-mail messages quite long. Even with attachments, however, e-mail messages continue to be text messages -- we'll see why when we get to the section on attachments.

E-mail Clients

You have probably already received several e-mail messages today. To look at them, you use some sort of e-mail client. Many people use well-known stand-alone clients like Microsoft Outlook, Outlook Express, Eudora or Pegasus. People who subscribe to free e-mail services like Hotmail or Yahoo use an e-mail client that appears in a Web page. If you are an AOL customer, you use AOL's e-mail reader. No matter which type of client you are using, it generally does four things:

It shows you a list of all of the messages in your mailbox by displaying the message headers. The header shows you who sent the mail, the subject of the mail and may also show the time and date of the message and the message size.

It lets you select a message header and read the body of the e-mail message.

It lets you create new messages and send them. You type in the e-mail address of the recipient and the subject for the message, and then type the body of the message.

Most e-mail clients also let you add attachments to messages you send and save the attachments from messages you receive.

Sophisticated e-mail clients may have all sorts of bells and whistles, but at the core, this is all that an e-mail client does.

A Simple E-mail Server

Given that you have an e-mail client on your machine, you are ready to send and receive e-mail. All that you need is an e-mail server for the client to connect to. Let's imagine what the simplest possible e-mail server would look like in order to get a basic understanding of the process. Then we will look at the real thing.

If you have read How Web Servers Work, then you know that machines on the Internet can run software applications that act as servers. There are Web servers, FTP servers, telnet servers and e-mail servers running on millions of machines on the Internet right now. These applications run all the time on the server machine and they listen to specific ports, waiting for people or programs to attach to the port (see How Web Servers Work for details). The simplest possible e-mail server would work something like this:

It would have a list of e-mail accounts, with one account for each person who can receive e-mail on the server. My account name might be mbrain, John Smith's might be jsmith, and so on.

It would have a text file for each account in the list. So the server would have a text file in its directory named MBRAIN.TXT, another named JSMITH.TXT, and so on.

If someone wanted to send me a message, the person would compose a text message ("Marshall, Can we have lunch Monday? John") in an e-mail client, and indicate that the message should go to mbrain. When the person presses the Send button, the e-mail client would connect to the e-mail server and pass to the server the name of the recipient (mbrain), the name of the sender (jsmith) and the body of the message.

The server would format those pieces of information and append them to the bottom of the MBRAIN.TXT file. The entry in the file might look like this:
From: jsmith
To: mbrain
Marshall,
Can we have lunch Monday?
John

There are several other pieces of information that the server might save into the file, like the time and date of receipt and a subject line; but overall, you can see that this is an extremely simple process.
As other people sent mail to mbrain, the server would simply append those messages to the bottom of the file in the order that they arrived. The text file would accumulate a series of five or 10 messages, and eventually I would log in to read them. When I wanted to look at my e-mail, my e-mail client would connect to the server machine. In the simplest possible system, it would:

Ask the server to send a copy of the MBRAIN.TXT file
Ask the server to erase and reset the MBRAIN.TXT file
Save the MBRAIN.TXT file on my local machine
Parse the file into the separate messages (using the word "From:" as the separator)
Show me all of the message headers in a list
When I double-clicked on a message header, it would find that message in the text file and show me its body.
You have to admit that this is a VERY simple system. Surprisingly, the real e-mail system that you use every day is not much more complicated than this!

The Real E-mail System
For the vast majority of people right now, the real e-mail system consists of two different servers running on a server machine. One is called the SMTP server, where SMTP stands for Simple Mail Transfer Protocol. The SMTP server handles outgoing mail. The other is either a POP3 server or an IMAP server, both of which handle incoming mail. POP stands for Post Office Protocol, and IMAP stands for Internet Mail Access Protocol. A typical e-mail server looks like this:
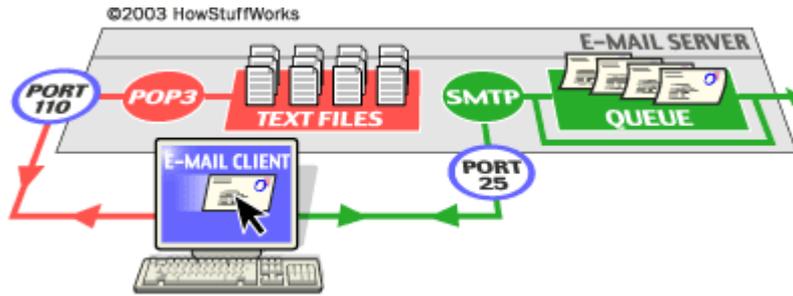
The SMTP server listens on well-known port number 25, POP3 listens on port 110 and IMAP uses port 143 (see How Web Servers Work for details on ports).

The SMTP Server
Whenever you send a piece of e-mail, your e-mail client interacts with the SMTP server to handle the sending. The SMTP server on your host may have conversations with other SMTP servers to actually deliver the e-mail.
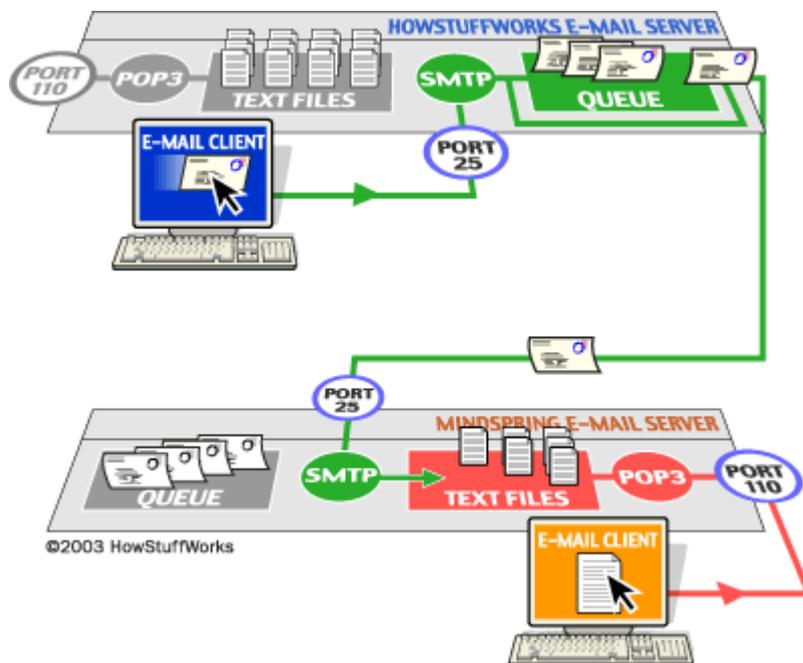
Let's assume that I want to send a piece of e-mail. My e-mail ID is **brain**, and I have my account on **howstuffworks.com**. I want to send e-mail to **jsmith@mindspring.com**. I am using a stand-alone e-mail client like Outlook Express.

When I set up my account at howstuffworks, I told Outlook Express the name of the mail server -- **mail.howstuffworks.com**. When I compose a message and press the Send button, here is what happens:

1. Outlook Express connects to the SMTP server at mail.howstuffworks.com using **port 25**.
2. Outlook Express has a conversation with the SMTP server, telling the SMTP server the address of the sender and the address of the recipient, as well as the body of the message.
3. The SMTP server takes the "to" address (jsmith@mindspring.com) and breaks it into two parts:

4.
- The recipient name (jsmith)
- The domain name (mindspring.com)

If the "to" address had been another user at howstuffworks.com, the SMTP server would simply hand the message to the POP3 server for howstuffworks.com (using a little program called the **delivery agent**). Since the recipient is at another domain, SMTP needs to communicate with that domain.

5. The SMTP server has a conversation with a **Domain Name Server**, or **DNS** (see How Web Servers Work for details). It says, "Can you give me the IP address of the SMTP server for mindspring.com?" The DNS replies with the one or more IP addresses for the SMTP server(s) that Mindspring operates.
6. The SMTP server at howstuffworks.com connects with the SMTP server at Mindspring using port 25. It has the same simple text conversation that my e-mail client had with the SMTP server for HowStuffWorks, and gives the message to the Mindspring server. The Mindspring server recognizes that the domain name for jsmith is at Mindspring, so it hands the message to Mindspring's POP3 server, which puts the message in jsmith's mailbox.

If, for some reason, the SMTP server at HowStuffWorks cannot connect with the SMTP server at Mindspring, then the message goes into a queue. The SMTP server on most machines uses a program called **sendmail** to do the actual sending, so this queue is called the **sendmail queue**. Sendmail will periodically try to resend the messages in its queue. For example, it might retry every 15 minutes. After four hours, it will usually send you a piece of mail that tells you there is some sort of problem. After five days, most sendmail configurations give up and return the mail to you undelivered.

The actual conversation that an e-mail client has with an SMTP server is incredibly simple and human readable. It is specified in public documents called **Requests For Comments** (RFC), and a typical conversation looks something like this:

helo test
250 mx1.mindspring.com Hello abc.sample.com
[220.57.69.37], pleased to meet you
mail from: test@sample.com
250 2.1.0 test@sample.com... Sender ok
rcpt to: jsmith@mindspring.com
250 2.1.5 jsmith... Recipient ok
data
354 Enter mail, end with "." on a line by itself
from: test@sample.com
to:jsmith@mindspring.com
subject: testing
John, I am testing...
.
250 2.0.0 e1NMajH24604 Message accepted
for delivery
quit
221 2.0.0 mx1.mindspring.com closing connection
Connection closed by foreign host.

What the e-mail client says is in blue, and what the SMTP server replies is in green. The e-mail client introduces itself, indicates the "from" and "to" addresses, delivers the body of the message

and then quits. You can, in fact, **telnet** to a mail server machine at port 25 and have one of these dialogs yourself -- this is how people "spoof" e-mail.

You can see that the SMTP server understands very simple text commands like HELO, MAIL, RCPT and DATA. The most common commands are:

The POP3 Server

In the simplest implementations of POP3, the server really does maintain a collection of text files -- one for each e-mail account. When a message arrives, the POP3 server simply appends it to the bottom of the recipient's file!

When you check your e-mail, your e-mail client connects to the POP3 server using **port 110**. The POP3 server requires an **account name** and a **password**. Once you have logged in, the POP3 server opens your text file and allows you to access it. Like the SMTP server, the POP3 server understands a very simple set of text commands. Here are the most common commands:

- **USER** - enter your user ID
- **PASS** - enter your password
- **QUIT** - quit the POP3 server
- **LIST** - list the messages and their size
- **RETR** - retrieve a message, pass it a message number
- **DELE** - delete a message, pass it a message number
- **TOP** - show the top x lines of a message, pass it a message number and the number of lines

Your e-mail client connects to the POP3 server and issues a series of commands to bring copies of your e-mail messages to your local machine. Generally, it will then delete the messages from the server (unless you've told the e-mail client not to).

You can see that the POP3 server simply acts as an interface between the e-mail client and the text file containing your messages. And again, you can see that the POP3 server is extremely simple! You can connect to it through telnet at port 110 and issue the commands yourself if you would like to (see How Web Servers Work for details on telnetting to servers).

The IMAP Server

As you can see, the POP3 protocol is very simple. It allows you to have a collection of messages stored in a text file on the server. Your e-mail client (e.g. Outlook Express) can connect to your POP3 e-mail server and download the messages from the POP3 text file onto your PC. That is about all that you can do with POP3.

Many users want to do far more than that with their e-mail, and they want their e-mail to remain on the server. The main reason for keeping your e-mail on the server is to allow users to connect from a variety of machines. With POP3, once you download your e-mail it is stuck on the machine to which you downloaded it. If you want to read your e-mail both on your desktop machine and your laptop (depending on whether you are working in the office or on the road), POP3 makes life difficult.

IMAP (Internet Mail Access Protocol) is a more advanced protocol that solves these problems. With IMAP, your mail stays on the e-mail server. You can organize your mail into folders, and all the folders live on the server as well. When you search your e-mail, the search occurs on the server machine, rather than on your machine. This approach makes it extremely easy for you to access your e-mail from any machine, and regardless of which machine you use, you have access to all of your mail in all of your folders.

Your e-mail client connects to the IMAP server using **port 143**. The e-mail client then issues a set of text commands that allow it to do things like list all the folders on the server, list all the message headers in a folder, get a specific e-mail message from the server, delete messages on the server or search through all of the e-mails on the server.

One problem that can arise with IMAP involves this simple question: "If all of my e-mail is stored on the server, then how can I read my mail if I am not connected to the Internet?" To solve this problem, most e-mail clients have some way to cache e-mail on the local machine. For example, the client will download all the messages and store their complete contents on the local machine (just like it would if it were talking to a POP3 server). The messages still exist on the IMAP server, but you now have copies on your machine. This allows you to read and reply to e-mail even if you have no connection to the Internet. The next time you establish a connection, you download all the new messages you received while disconnected and send all the mail that you wrote while disconnected.

Attachments

Your e-mail client allows you to add attachments to e-mail messages you send, and also lets you save attachments from messages that you receive. Attachments might include word processing documents, spreadsheets, sound files, snapshots and pieces of software. Usually, an attachment is not text (if it were, you would simply include it in the body of the message). Since e-mail messages can contain only text information, and attachments are not text, there is a problem that needs to be solved.

In the early days of e-mail, you solved this problem by hand, using a program called **uuencode**. The uuencode program assumes that the file contains binary information. It extracts 3 bytes from the binary file and converts them to four text characters (that is, it takes 6 bits at a time, adds 32 to the value of the 6 bits and creates a text character -- see How Bits and Bytes Work to learn more about ASCII characters). What uuencode produces, therefore, is an **encoded** version of the original binary file that contains only text characters. In the early days of e-mail, you would run uuencode yourself and paste the uuencoded file into your e-mail message.

Here is typical output from the uuencode program:

```
begin 644 reports
M9W)E<" B<&P_(B O=F%%R+VQ9O9R]H=''1(39#36V(B]W96(V-V%G96965>+C1F-
BYA8V95>-E<W,N;.;;&>GM('P@8%W5T("H"%UUUF#.+60@C`\B(P@8%W5T
("UF#($H"+60@(B8Q(#X(V6I:5A9RUFM(C``M#U-*]C>
W5N="I59V86=E<II?%\('-O+0<0"0/.B!39&%%T<RTD,4&9=<
" @[M?%B]9Y&)3 71+4+2+$G5I2M67%O5B9H5699YG65N=F,I#L=M;F9871%J5A=
&$@<65G;F5U5=H="M]R>P66@<65G;F5U5=D</;C+-M7#"!!]^$V+%89/-]HN4O:9+R^)(YB@B
]5:&'5R9U5HM86G11+L4V+=*=E>#V,M9^0$$^!8='QW)29P#.B!39%%7$%;A>,(]A!W6EB`M4686?L:W18@96JA)NO
$^?Q]T]U)G1%L;V<P/B%L+5S+#B0&0<P>II?%\('-O+`@T4=W@@.&0$HT^P)#
$^?)1]G:711/5C<P/B%,;5=L+9U+%^K=CT(B<9M=9;'<:G1&D-P0>S[P)11:`W6T"@M
D+/HIA
```

    End

The recipient would then save the uuencoded portion of the message to a file and run **uudecode** on it to translate it back to binary. The word "reports" in the first line tells uudecode what to name the output file.

Modern e-mail clients are doing exactly the same thing, but they run uuencode and uudecode for you automatically. If you look at a raw e-mail file that contains attachments, you'll find that the attachment is represented in the same uuencoded text format shown above!

Considering its tremendous impact on society, having forever changed the way we communicate, today's e-mail system is one of the simplest things ever devised! There are parts of the system,

like the routing rules in sendmail, that get complicated, but the basic system is incredibly straightforward.

The next time you send an e-mail, you'll know exactly how it's getting to its destination!

For more information on e-mail and related topics, check out the links on the next page.

- **HELO** - introduce yourself
- **EHLO** - introduce yourself and request extended mode
- **MAIL FROM:** - specify the sender
- **RCPT TO:** - specify the recipient
- **DATA** - specify the body of the message (To:, From: and Subject: should be the first three lines.)
- **RSET** - reset
- **QUIT** - quit the session
- **HELP** - get help on commands
- **VRFY** - verify an address
- **EXPN** - expand an address
- **VERB** - verbose