A day in the life of an email

We use it every day, but what exactly goes on when we click send?

Email is now so ubiquitous, we no longer even consider how it all works. Billions of emails are sent each day (the majority of which are spam, admittedly), and even with the rise of social networking, we haven't abandoned email yet.

Some good, some bad; some work-related, some personal: it's the communications medium of the 21st century.

But what exactly is an email? How does it get from me to you? What processes and servers have to be running in order to ensure all this magic works to the point where we don't need to worry about it?

Back in the very early days, messages could only be sent from computer to computer on the same network. For this to happen, both computers had to be running and online (that is, both endpoints had to have users logged in) since the originating computer made a direct connection to the destination computer in order to transfer the message. This worked in essentially the same way that phone switches work to route a call: the originating and the destination phone must be connected directly for the length of the phone call. For computers on the same network, this method worked pretty well, but it didn't scale at all once we started to link local networks together.

The birth of email

In 1969, the precursor of the internet, ARPANet, was created by a research team at MIT and at DARPA (Defense Advanced Research Projects Agency). It was the first packet-switched network, so named because all data traffic was split up into packets. The packets were numbered sequentially and put into digital envelopes, with destination addresses encoded into the envelopes. ARPANet was a collection of servers, each able to receive and pass packets onto other servers on the network. This meant that a large message would be split into different packets, and each packet might be routed a different way through the network to the destination. Each node on the network knew only enough to pass on packets that weren't destined for itself, and it was the receiving computer that was responsible for collecting all the packets that made up a message and checking that none were missing. This methodology meant that packets from many different messages from many sources could be interleaved and sent on a link, without the need to tie up the link to send a single message.

A couple of years later in 1971, Ray Tomlinson implemented the first system that we would recognise as email. His system was based on a program that copied files across a network and allowed users of different networks to send messages (as files) to each other. To help with the addressing of the ▸

```
Date: Thu, 25 Nov 2010 09:56:39 -0500 (EST)
From: communications-digital-edition@acm.org
Reply-To: communications-digital-edition@acm.org
To: sample@mydomain.com
Message-ID: <2554975.73111290696999078.
JavaMail.Administrator@acm28-8>
Subject: Communications of the ACM: December
2010 Issue
MIME-Version: 1.0
Content-Type: multipart/mixed;
    boundary="----=_Part_5975_24868004.1290696
999078"
X-Mailer: ColdFusion 8 Application Server
X-Nonspam: Statistical 65%
```

▲ **Figure 1: Header section from a recent email.**

email, he came up with a simple solution: separate the username from the remote network domain name by use of the '@' sign – a convention we still use today.

The earliest emails sent were text files, usually seven-bit ASCII. Although emails are no longer physical files, they remain as text. An email consists of two main parts: the header and the message section, separated by a single null line (that is, a line that that only comprises a carriage return/line feed).

Modern messages

Nowadays, the message section can – and usually does – have a lot more structure associated with it thanks to the MIME (Multipurpose Internet Mail Extensions) standard. This standard extends the original seven-bit ASCII-only messages to incorporate other character sets including Unicode, attachments (usually encoded with something like Uuencode or base64) and multiple parts (where a message is encoded as pure text, HTML and rich text within the same email).

The header section remains resolutely ASCII (although MIME does allow for addressing with other character sets). It consists of various header information about the email, such as the subject, the recipient address(es), who sent it, a unique message ID, where replies should go to, and so on. Email clients usually suppress most of this information when displaying an email, although there's usually a way to show them.

Figure 1 shows an example header section from a recent email from the Association for Computing Machinery (ACM). Reading this you can see who sent it (and where to send the response to if I wanted to reply) and when it was sent. The message itself is in a multipart MIME format (the line that defines the boundary between the parts is shown) – as it happens, the message is represented in both straight text format and in HTML within parts of the email and it's up to the email client as to which is actually displayed to the user.

Routing for emails

What also generally happens when an email is sent across the internet is that intermediary servers add extra routing information to the header section. For simplicity, this information is prepended to the header section, so the server doesn't have to hunt for the end of the section to add it. The routing information generally details which email systems looked after and rerouted the email on its way to the inbox. For example, I've set up my personal email so that all messages are rerouted to Gmail, which means I can access my email easily using a browser or my phone. The routing information included on the example email from Figure 1 shows (reading from the bottom upwards) the originating server name, the receipt by my email server at my personal domain, its sending on of the email to Google, the receipt by Gmail, and the final delivery to my inbox (see Figure 2). By tracing the times shown on the routing information, I can see that the email appeared in my domain's inbox in a matter of seconds, whereas the automated Gmail fetch process took about 30 minutes.

Although legitimate email servers will provide valid information as they prepend

Spotlight on… Email issues

The biggest problems with email are that it's unauthenticated, unverified, unencrypted and plain text. Because of this, it's extremely easy to spoof parts of the email to make it appear as though it's been sent from someone else, somewhere else. SMTP itself doesn't add any authentication – any SMTP server can connect to any other. Indeed, you could argue that the success of email is mostly down to the fact that the system is so simple and open.

The ability to easily spoof emails has led to the rise of spamming and phishing attacks where the real sender is to be hidden at all costs. The From, Reply-To and Return-Path header fields in such a fraudulent email are fake, or are legitimate addresses that have nothing to do with the email. Since the payload of the email is in the message text ('This email is from your bank. Something's wrong. Click here to log in,' or 'Click here to see Anna Kournikova!'), it makes sense to set the header fields to something that looks legitimate (CustomerService@YourBank.com, or the email address of one of your contacts. In many cases, your spam checker will catch these messages, but if you want to check manually, you would have to view the header section of the email, in particular the routing information. Did the email purportedly from joe@example.com start off in the 'example.com' domain? Note that email routing information can be spoofed too but you should be able to trace a complete path from source to target.

It's amazing to consider that, in the old days, before spam became such a profitable commercial enterprise, it was fairly usual for SMTP servers to be open relays. They would pass on email from anyone, spoofing the header fields when necessary to make it look like the user's email had come from their usual server. This hasn't been accepted practise for a long time, though.

Email encryption

From the description of the format of an email, you can see that it's all in plain text. Anyone who can inject themselves between SMTP servers, or between you and your POP3 server can read your emails. There are government agencies and systems that purport to do this already, including the NSA in the USA and the ECHELON system in the UK. What can you do? The answer is to encrypt your emails. Emails are text-based, so this involves an encryption step using a symmetric algorithm like AES, or a public key algorithm like RSA. After that, the encrypted text must be encoded into text using something like base64 or UUENCODE. The PGP (Pretty Good Privacy) email security system wraps this up for you, although Outlook has support for signing, encrypting and decrypting emails once you've purchased a digital certificate for email from a company like Verisign.

```
Delivered-To: myaddress@gmail.com
Received: by 10.227.9.154 with SMTP id l26cs77430wbl;
        Thu, 25 Nov 2010 07:28:00 -0800 (PST)
Received: by 10.227.156.21 with SMTP id u21mr1064770wbw.9.1290698878755;
        Thu, 25 Nov 2010 07:27:58 -0800 (PST)
Received: by 10.241.241.88 with POP3 id 24mf16292wwb.132;
        Thu, 25 Nov 2010 07:27:58 -0800 (PST)
X-Gmail-Fetch-Info: sample@mydomain.com 1 pop.secureserver.net 110 julianb@boyet.com
Received: (qmail 5873 invoked from network); 25 Nov 2010 14:56:40 -0000
Received: from unknown (HELO m1pismtp01-020.prod.mesa1.secureserver.net) ([10.8.12.20])
          (envelope-sender <communications-digital-edition@acm.org>)
          by smtp25-02.prod.mesa1.secureserver.net (qmail-1.03) with SMTP
          for <sample@mydomain.com>; 25 Nov 2010 14:56:40 -0000
Received: from acm26-2.acm.org ([199.222.69.107])
  by m1pismtp01-020.prod.mesa1.secureserver.net with ESMTP; 25 Nov 2010 07:56:40 -0700
Return-Path: <communications-digital-edition@acm.org>
Received: from acm28-8 ([192.168.1.104])
        by acm26-2.acm.org (IceWarp 10.0.7) with SMTP id HZD67139
        for <sample@mydomain.com>; Thu, 25 Nov 2010 09:56:39 -0500
```

▲ **Figure 2: Routing information from the same email as Figure 1.**

routing information, many others won't. Spam emails especially tend to contain fake routing information, so you can't rely on this header information until the point when it reaches your email server.

How emails are sent

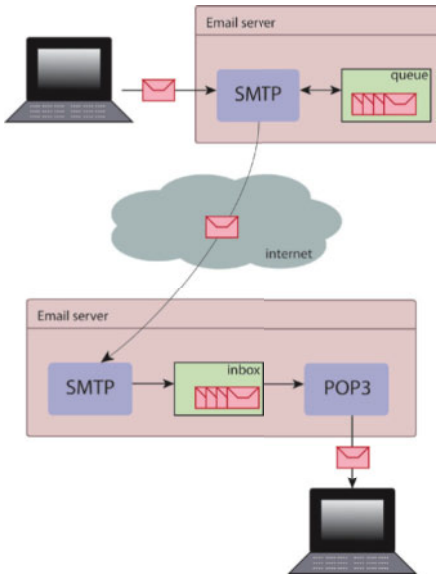
Having touched on routing for emails, we should take a look at what goes on when you hit 'Send' on an email message until the point when the recipient reads it in their email client.

The vast majority of email uses two types of server to send an email from A to B: the outgoing mail server and the incoming mail server. The outgoing email server is almost certainly an SMTP server (Simple Mail Transfer Protocol), while the incoming server can be a POP3 (Post Office Protocol) or IMAP server (Internet Mail Access Protocol).

When you set up your email client (let's say this is Microsoft Outlook, since that's what I use), you specify for it the address of your SMTP server. You also define to it the user ID and password that has been assigned to you to use the server's facilities (without a properly protected SMTP server, your email could be hijacked for spam broadcast purposes).

You write an email in Outlook, specify the recipient and press 'Send'. Outlook formats the message according the email standards (since 2008 this is defined in the RFC5322 document, which superseded RFC2822 from 2001, which in turn superseded RFC822 from 1982). It then connects to the SMTP server on port 25, passing the user id and password for authentication, and sends the email.

Once the SMTP server gets the email (and adds its routing information), it looks for the address to send it to within the header section. It strips off the username and the @ sign, leaving the domain name that the email must be sent to. The SMTP server queries the Domain Name System (DNS) for the MX (Mail eXchange) records for that domain name. The DNS entry for a domain name consists of a set of records defining the addresses of servers that process various types of connection (there are A records, AAAA records, CNAME records, and so on), and the MX record defines the server that can receive emails for the domain.



▲ **Figure 3: The typical route taken by an email message from source to destination.**

Web email security

These days, most people use one of the free email services provided by companies like Microsoft (Hotmail), Google (Gmail) or Yahoo (Yahoo! Mail).

Although the physical systems themselves are as secure as modern corporate systems usually are, because you're viewing your email in a browser, you're vulnerable to various ingenious cross-site scripting attacks. These attacks use cleverly designed, malicious HTML and Javascript code in an email to try to gain access your inbox. Once they have access to your account, they could do something as trivial as using it to send out bogus emails with similar embedded hacks to your contact list (I'm sure you've received that kind of email), to searching through all your messages looking for telltale words such as 'account' and 'password' and then forwarding them to the bad guys, who can use them to access your accounts.

Our recommendation is to avoid saving personal account information in messages on your email system, whether you're using webmail or something else. It's just too easy to lose once the system is compromised, and the larger the system, the more attacks there will be against it from criminals trying to gain control. ■

For example, with my personal domain, the A record currently points to 97.74.144.79. This is the IP address of the server that hosts my domain and my website. My highest priority MX record (you can have several MX records and they are ordered according their priority, the order in which SMTP servers try to connect with them) is pointing to smtp.secureserver.net, the GoDaddy server that deals with my email. And, yes, your SMTP server then has to resolve secureserver.net to an IP address in order to continue.

You've got mail

Your SMTP server then sends your email to the recipient's MTP server using the Simple Mail Transfer Protocol. Of course, it may be that, due to unforeseen circumstances, my SMTP server is offline or down. In this case, your SMTP server will put your email in a queue and try to send it again later. If the server finds that after several tries it can't send the email at all, it wraps the email in a 'cannot deliver' message and posts that to your email inbox.

But let's assume that all goes well and my SMTP server receives your email (and adds its routing information). It in turn reads the recipient email address, works out the user name, and puts the email in my inbox.

By 'inbox', I don't mean the inbox in Outlook or whatever email client you use. I mean the inbox on the email server for my email address. In the old days, the inbox was very simple: it was a set of text files, one per email, in a folder named after my email address (or maybe a single text file and new emails were appended). These days it's more integrated – the inbox is in a database, with the usual failsafe guarantees that provides.

Incoming mail servers

We now come to the opposite end of the email trail: the incoming mail server. Ignoring the heavy duty corporate email systems such as Microsoft Exchange, Lotus Notes or Blackberry Server, there are two main ones in use today: the POP3 server and the IMAP server. POP3 is the older and less sophisticated of the two, but they both have roughly the same features. The main difference between them is what happens to the emails. With POP3, although you can leave emails on the server, there's no provision for marking any as read/unread – the assumption is that emails are downloaded to your client and deleted from the server. Of course, this presents a problem if you want to use a variety of clients to access your email, because you may find that a particular email that you want to read is on a different PC to the one you're currently using.

With IMAP, the assumption is the opposite: emails are left on the server and can be marked as read/unread. This means that you can access your emails through a variety of email clients (desktop, phone, web) and all clients will agree on the current state of the emails. With IMAP you can also do things like set up an inbox folder tree on the server or move emails around the tree, and again the clients will all agree on the current state of the inbox.

> **"The earliest emails were sent in text files – usually seven-bit ASCII"**

Let's assume that I'm using POP3. Again, I will have configured Outlook so that my incoming server is at such and such address and has a particular user ID and password (I can't have all and sundry reading my emails after all). When I ask Outlook to retrieve all my emails, it will log in to the POP3 server with the credentials I gave, ask for a list of emails, and then download and delete them one by one. It will read the header information from each email in order to ascertain how the message is structured, how the constituent parts are encoded, from whom the email came, the delivery date/time, and so on.

Outlook will then decode and display the emails for me to peruse and read, and with that we come to the end of the journey for that email, from your PC to mine. PCP

Julian M Bucknall has worked for companies ranging from TurboPower to Microsoft, and is now CTO for Developer Express.
feedback@pcplus.co.uk