High Level Internet Protocols and Client Applications

Major Internet Protocols/Tools:

- E-mail
- FTP
- Telnet
- WWW

In this lecture we explore the first 3 in detail and then follow the development of these into the WWW.

E-mail, FTP, Telnet *etc.* Packages used in this course

The main packages used to explore the Internet in this course are:

E-mail — Mac Mail, Unix Command line Mail.

FTP — Fetch, Unix Command Line FTP, Web Browser.

Telnet — Mac Terminal, Secure Shell

WWW — Safari, Internet Explorer, Netscape, Opera

E-mail

Electronic mail, or E-mail, is your personal connection to Internet.

- All of the millions of people on the Internet have their own e-mail addresses.
- You should know your E-mail address by now. It will be something like:

  `A.N.Other@cs.cf.ac.uk`

  or

  `santa@santaclaus.com`

E-mail is like regular mail.

- You send mail to people at their particular addresses.
- People write to you at your E-mail address.
- You can subscribe to the electronic equivalent of magazines and newspapers.
- You even get electronic junk mail — SPAM

Why use E-mail

E-mail has three distinct advantages over regular mail.

- Speed — your message can reach the other side of the world in hours, minutes or even seconds (depending on where you drop off your mail and the state of the connections between there and your recipient).
- Cost — for you (at least whilst at University) email is free. Even if you pay for it the cost is minimal (a local phone call).
- Extensible – More than mail. You can use e-mail to access databases and file libraries. You'll see how to do this later, along with learning how to transfer program and data files through e-mail.

Email on the Department's Computers/Macintoshes

Hopefully: You should already be using email

Details of a variety of Email applications may be found at http://www.cs.cf.ac.uk/systems/notes.html

The more useful/relevant docs here include:

401 Configuring and Using Pegasus Mail (Html) (PDF)

402 Configuring and Using Evolution Mail (Html) (PDF)

403 Configuring and using Macintosh Mail.app ((Html) (PDF)

404 Using the Web Mail Service (Html) (PDF)

Understanding Mail Messages

- Mail messages consist of two parts:
  - a header and
  - a body
- Header consists of a number of lines at the beginning of the message
- Body consists of the actual text of the message

Sample Email Message

```
Return-Path: <opt!opt.kpi.kiev.ua!vladimir@gar.kiev.ua>
Delivery-Date: Tue, 23 Sep 1997 14:49:58 +0100 Received: from
logrus.rada.kiev.ua (actually host 195.230.148.1)
          by sentinel.cs.cf.ac.uk with SMTP (PP);
          Tue, 23 Sep 1997 14:46:30 +0100 Received: from polytech.kiev.ua
([194.44.128.5])
          by logrus.rada.kiev.ua (8.6.12/0418) with ESMTP id QAA02474
          for <dave@cs.cf.ac.uk>; Tue, 23 Sep 1997 16:39:41 +0300 Received: from
opt.UUCP by polytech.kiev.ua with UUCP id QAA07291;
          (8.6.12/zad/1.1) Tue, 23 Sep 1997 16:27:44 +0300 Received: by
opt.kpi.kiev.ua (UUPC/@ v6.14e, 07May95);
          id AA30288 Tue, 23 Sep 1997 15:01:32 +0400 To:
Dave.Marshall@cs.cf.ac.uk Message-Id: <AAB6w9qWN9@gar.chik.kiev.ua>
Organization: Russian Department X-UIDL: 875022733.001 From: Vladimir
<vladimir@gar.chik.kiev.ua> Date: Tue, 23 Sep 97 15:01:31 +0400 X-Mailer: BML
[MS/DOS Beauty Mail v1.36h] Return-Receipt-To: vladimir@gar.chik.kiev.ua
Subject: Test Message Lines: 21 MIME-Version: 1.0 Content-Type: text/plain;
charset=us-ascii Content-Transfer-Encoding: 7bit Content-Length: 646


Dear All

This is the body of this test message ......
```

Viewing Mail Headers

Note: Most Mail Applications do not usually show the complete mail header.

- But mail applications usually provide easy access to such information
- (example using the Horde Mail program)

Mail Headers Dissected

- The `Received` statements indicate the path the message took, as well as the times, dates, and what programs were being used each step of the way
- Notice that a long the way the message was received by the following station
- `id AA30288 Tue, 23 Sep 1997 15:01:32 +0400`
  - `AA30288` represents a local identifier associated with the message
  - `Tue, 23 Sep 1997 15:01:32 +0400` represents the time the message was received by this station
- Since the Internet spans the globe, its users are in many different time zones
- The Internet has adopted Greenwich Mean Time (GMT) as the standard
- Sometimes referred to as Universal Time (UT)
- The *+0400* indicates that the message was received 0400 hours later than GMT

Mailing Text and Binary Files as Attachments

- Mail messages may be comprised: simple text messages, HTML or binary coded messages, or a combination of both
- Any file that can not be represented as a series of ASCII characters can be considered binary data
- All pictures, sounds, or programs are examples of binary information
- All Internet mail clients can send and receive text messages
- However, many mail clients cannot handle binary data
  - e.g. programs which use **SMTP** (**Simple Mail Transfer Protocol**) alone cannot transfer binary data
  - In most cases files must be manually encoded prior to transmission
  - **More on SMTP Later**

High Level Internet protocols: Email Protocols

Protocols:

- *Everyone MUST understand the protocols that are spoken on the net.*
- A *protocol* is a set of **standardised** commands and responses.
- **Essentially you (or your client application) enter a dialogue or conversation with a server.**

Brief Recap: High Level and Low Level internet Protocols

There are two basic levels of internet protocols:

Low-level (Mid Layer OSI) — TCP/IP: crucial to the Internet,

- More Complex in structure — ultimately binary bit streams
- Enables a server to *listen* and respond to incoming conversation.
- Conversations arrive at a specific port: incoming packets of information can arrive here
- Each type of service (Email or FTP) has its ownport

High-level (Application Layer OSI) — Email, FTP, Telnet, HTTP *etc.* Protocols.

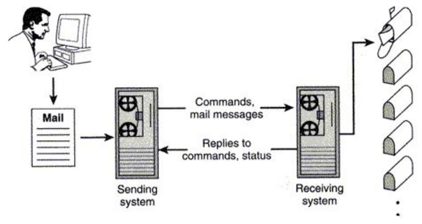
The Basic Protocol Mechanism

The protocols all have the same basic pattern, which is essentially a dialogue of sorts:

- **Begin a Conversation** — Your computer (the client) starts a conversation with another computer (the server).
- **Hold a Conversation** — During the conversation, commands are sent and acknowledged. The server typically acknowledges a command with some sort of code (number) which may be further qualified with simple text commands. The code indicates if the command has been successful or if some error has occured.
- **End a Conversation** — The conversation is terminated.

A Network Protocol Communications Schematic Diagram

Here the example uses email (others similar):



The Mail Protocol Suite Example Protocols

The main Email protocols include:

SENDING MAIL:

SMTP — Simple Mail Transfer Protocol (more below)

MIME — Multipurpose Internet Mail Extensions.

ACCESSING/RETRIEVING MAIL:

POP3 — Post Offce Protocol (more below)

IMAP — Internet Message Access Protocol. Accessing mail over a (possibly shared) mail servers.

When using email (or other clients) you usually do not need to know or worry about the protocols:

- Mail Client application (once set up correctly) deals with this.

Writing Your Own Clients?

Not Just Yet, Maybe?

- **Not too hard**: Any programming language (with basic networking libraries)
- **Not too hard**: Especially when using high level Mail protocols. (FTP Similar)
- Java is a very good choice as is Perl or PHP (**More Later**)
- Need to write client to converse with the mail server in the **PROTOCOL**. This is not difficult to do
- Program (your client) needs to
  - Connect to correct Server and **Port** (TCP port 25).
  - Output Protocol commands in correct format and order
  - Recieve Server resonses and understand them
  - Many packages (Java and Perl) do most of this for you (A little more later on Perl)

Sending Mail (SMTP: SMTP Simple Mail Transfer Protocol)

- Historically, one of the OLDER Email protocols.
- Command-line mail clients use protocol directly in communication e.g. UNIX `mailx`)
- The message can only use alphanumeric characters.

Basic Idea:

- Simple Preparation: **Entire message needs to be composed First**.
- Entire message then sent

SMTP Commands

SMTP uses several commands to communicate with mail servers.

The basic SMTP commands include:

`HELO` — Initiates a conversation with the mail server. When using this command you can specify your domain name so that the mail server knows who you are. For example,
`HELO mailhost.cf.ac.uk`.

`MAIL` — Indicates who is sending the mail. For example,
`MAIL FROM: <dave@cs.cf.ac.uk>`.
The name of the person who is sending the mail message.
Any returned mail will be sent back to this address.

Some More Basic SMTP Commands

`RCPT` — Indicates who is recieving the mail. For example,
`RCPT TO: <user@email.com>`. You can indicate more than one user by issuing multiple `RCPT` commands.

`DATA` — Indicates that you are about to send the text (or body) of the message. The message text must end with the following five letter sequence: "\r\n.\r\n."

`QUIT` — Indicates that the conversation is over.

There are further SMTP commands, such as: EXPN,HELP, NOOP, SEND, SAML, SOML, TURN, VRFY.
You can find out more from text books or the internet if your interested. In this course we wont be going this deep.

SMTP Mail Server Replies

Recall:Most Protocol transacation are a simple dialogue:

- Every command will receive a reply from the mail server
- The reply format is:
  - a three digit number
  - followed by some text describing the reply.
- For example,

```
250 OK
```

or

```
500 Syntax error, command unrecognized.
```

SMTP Mail Server Reply Codes

The complete list of reply codes is shown below: (you'll never see most of them if you program your mail server correctly!!)

211 — A system status or help reply.

214 — Help Message.

220 — The server is ready.

221 — The server is ending the conversation.

250 — The requested action was completed.

251 — The specified user is not local, but the server will forward the mail message.

354 — This is a reply to the `DATA` command. After getting this, start sending the body of the mail message, ending with "\r\n.\r\n."

421 — The mail server will be shut down. Save the mail message and try again later.

450 — The mailbox that you are trying to reach is busy. Wait a little while and try again.

451 — The requested action was not done. Some error occurmiles in the mail server.

452 — The requested action was not done. The mail server ran out of system storage.

500 — The last command contained a syntax error or the command line was too long.

501 — The parameters or arguments in the last command contained a syntax error.

502 — The mail server has not implemented the last command.

503 — The last command was sent out of sequence. For example, you might have sent `DATA` before sending `RECV`.

504 — One of the parameters of the last command has not been implemented by the server.

550 — The mailbox that you are trying to reach can't be found or you don't have access rights.

551 — The specified user is not local; part of the text of the message will contain a forwarding address.

552 — The mailbox that you are trying to reach has run out of space. Store the message and try again tomorrow or in a few days-after the user gets a chance to delete some messages.

553 — The mail address that you specified was not syntactically correct.

554 — The mail transaction has failed for unknown causes.

Yet another Protocol!!! — MIME

- To enable users to send and receive binary data another protocol, named **MIME**, was developed- advantage is that it's automatic
- **MIME** stands for **Multipurpose Internet Mail Extensions**
- MIME is designed for Sending Binary Data
- The binary data is attached to the mail message
- It is not necessary for senders and receivers to use the same mail client or specifically encode the file, only that both support MIME
- Some (older/command line) mail clients do not support MIME

Compressing/encoding attachments

- To reduce the amount of data which is transferred via the Internet, data can be *compressed* before mailing or ftping *etc.*.
- Common file compression formats include:
  - Zip
  - Stuffit
  - Binhex
  - UUencode
  - Unix Compress

  Another use of packages such as Zip is not only compression, but to more easily send a group of files in one go, instead of attaching multiple files.

Receiving Mail (POP: Post Office Protocol)

The POP protocol is used to receive/access email. It is similar to SMTP, but has added security.

Why do you thing added security is need?

POP Session States

A POP3 session progresses through a number of states:

Connection — Once the TCP connection has been opened and the POP3 server has sent the greeting, then

`AUTHORIZATION` **state** — Login on

`TRANSACTION` **state** — The Dialogue

`UPDATE` **state** — When the client has issued the QUIT command:

- The session enters the UPDATE state.
- Now the POP3 server releases any resources acquired during the `TRANSACTION` state and says goodbye.
- The TCP connection is then closed.

Essential POP3 Commands(1): `AUTHORIZATION` state

The following commands are essential and are valid in the `AUTHORIZATION` state:

`USER name` — Login process starts. Only you can read your mail.

`PASS string` — Login proceeds. A valid Password must be supply to match Username.

`QUIT` — Terminate `AUTHORIZATION`.

Essential POP3 Commands(2): `TRANSACTION` state

The following commands are essential and are valid in the `TRANSACTION` state:

`STAT` — The POP3 server issues a positive response with a line containing information for the maildrop if it receives this message.

`LIST msg` — List mail messages (or messages refered to by `msg`) by number and size of messages waiting to be read.
A `msg` message-number (optional), which, if present, may NOT refer to a message marked as deleted.

`RETR msg` — Retrieve a mail message from the server

`DELE msg` — Delete a mail message

`NOOP` — Check if server is still up. The POP3 server does nothing, it merely replies with a positive response.

`RSET` — Reset the server. If any messages have been marked as deleted they are unmarked.

Essential POP3 Commands(3): `UPDATE` state

The following commands are essential and are valid in the `UPDATE` state:

`QUIT` — end the session.
The POP3 server removes all messages marked as deleted from the mail box.

Email Tips and Guidelines

- Do not assume mail is private - like sending a postcard
- Once you send mail there is no way to get it back, even if it is undelivered or unread
- Do not assume any message you receive has only been sent to you
  - By using *Bcc:*, a blind copy may be sent that will not appear in the recipients header
- If your message can be misunderstood, it probably will be

Creating Effective E-mail (1)

It is hard to convey any feeling in a E-mail. However some techniques may help overcome this:

- Provide your audience with adequate context:
  - Use meaningful subject lines
    DO NOT SEND BLANK SUBJECT HEADERS
    - ∗ When replying to Mali:
      Your mailer should preface the subject line with “Re:” or “RE:” (for REgarding).
    - ∗ Use “FYI:” (For Your Information), “URGENT:” , in header. *E.g*:

```
Subject: URGENT: Internet Lecture Cancelled.
```

Creating Effective E-mail (2)

– Quote the E-mail to which you are responding
  If you are referring to previous E-mail, you should explicitly quote that document to provide context.
  Instead of sending E-mail that says:

```
yes
```

  Say:

```
> Are you going to Woodville
> for a drink after lectures?

  yes
```

  Most E-mailers let you include previous postings and insert a >

Creating Effective E-mail (3)

• Find replacements for gestures and intonation:

– Asterisks — for light emphasis.
  Instead of:

```
I said that I was going to go
last Thursday.
```

  Say:

```
I *said* that I was going to go
last Thursday.
```

  Or:

```
I said that I was going to go
last *Thursday*.
```

Creating Effective E-mail (4)

• Capital letters — Stronger Emphasis
  Capital letter perceived to be Harsh — Mix up general text

```
> Should I do my coursework?

yes, you should do your coursework.
```

  is not as striking as:

```
> Should I do my coursework?

YES, you SHOULD do YOUR COURSEWORK.
```

Creating Effective E-mail (5)

• Creative punctuation — Strong emphasis, pauses and other uses:

```
> Should I do my coursework?

Err.... YES, you SHOULD do
YOUR COURSEWORK!!!!!.
```

• Smileys — an ASCII drawing of a facial expression
• Mobile Phone Texting type messaging also now popular.
• E.G. lol, k, l8tr, lmao, np, etc.......lots of these!

Common Smileys

:-)	Your basic smiley — inflects a sarcastic or joking statement.
`;-)`	Winking smiley — inflects a flirtatious and/or sarcastic remark.
`:-(`	Frowning smiley — User did not like that last statement or is upset or depressed about something.
`:-I`	Indifferent smiley.
`:->`	User just made a really biting sarcastic remark.
`:->`	User just made a really devilish remark.
`;->`	Winking and devil combined — A very lewd remark was just made.
`;-P`	Sticking you Tongue Out.

Back
Close