RFC

- as a network manager it is important to understand some of the subtle issues within networking
- within the TCP/IP suite of protocols including:
  - TCP, UDP, IP, SNMP, ICMP, SMTP, FTP, HTTP, etc
- the operation of many of these protocols maybe complex
  - but we can find out about these protocols via RFCs
  - request for comments
- some are complex, others are not
  - they are all free!

RFC

- RFCs are documents to communicate ideas for development in the Internet community
  - some of the RFCs have become Internet standards
  - many codes of practice are stated within RFCs
- can down load RPCs via netscape or explorer

FTP Clients

- standard ftp interface is crude because the interface is so rigid
  - there are many graphical alternatives
- and some pleasant command line clients exist (`ncftp`)

Configuring FTP Server

- *very easy to install*
  - *the standard method of uploading files on the Internet*

Uploading files

```
ftp tsx-11.mit.edu
Connected to tsx-11.mit.edu
Name (nic.ddn.mil:jbloggs): ftp
331 Guest login ok, send "guest" as password
Password:jbloggs@some.org
230 Guest login ok, access restrictions apply.
ftp> cd pub/incoming
250 CWD command successful.
ftp> put mycode.tar.gz
ftp> quit
```


Configuring FTP Server

- installation occurs ‘out of the box’
  - system administrator does not *need* to do anything extra
- however a *good* system administrator will!
- double check that anonymous users cannot access files they are not allowed
  - restrict access to ftp through `/etc/ftpusers`
  - plain ASCII file containing users who *cannot* login

Configuring FTP Server

- should contain non real people: root, uucp, news, bin, nobody etc
- why do this?
  - prevents system administrator from transferring files
  - all passwords are transmitted in plain text
  - sniffer could catch system passwords!

Configuring FTP Server

- ensure that the user ‘ftp’ does not own ~ftp
- why
  - anonymous users can alter the ftp environment
  - delete and add files at will
  - create a file called `.rhosts`

Configuring FTP Server

- ensure that you do not become a ftp software pirate repository
- if you need an incoming directory ensure that it has mode 1733 and is owned by root
  - allows files to be left by users but **not** listed
- put quota on ftp user
- generate a shell script to be run every 30 minutes which moves all `incoming` contents into another directory
- put entry in `crontab` to perform this duty

Testing FTP

- prerequisites:
  - make sure DNS is operating
  - DNS is a name to IP lookup mechanism
    - check this is working by trying: `nslookup`
  - check IP is operating via `ping` and `traceroute`

Restricting FTP access

- an easy but extremely useful method is to restrict the IP addresses which can ftp from this server
- this can be done via: `/etc/hosts.deny`
- ```
  #
  # The PARANOID wild card matches any host whose name
  # does not match its address.
  ALL: PARANOID
  ftpd: ALL EXCEPT 193.63.130.*
  ```
- which disallows all clients except the trusted `193.63.130.*` subnet

FTP operation under GNU/Linux

- note this is normally done the install mechanism
  - no manual intervention is necessary
  - we include it here to aid understanding and to help you fix problems/security holes
- the server process is called `ftpd`
  - it is normally run from a master daemon `inetd` which reads the configuration file `/etc/inetd.conf`
- `inetd` checks all incoming tcp/udp connections starts the relevant program in this case `ftpd`
  ```

/etc/inetd.conf

```
#
# These are standard services.
ftp     stream tcp nowait root /usr/sbin/tcpd \
        /usr/sbin/wu.ftpd
telnet stream tcp nowait root /usr/sbin/tcpd \
        /usr/sbin/in.telnetd
```


/etc/services

```
ftp     21/tcp
telnet  23/tcp
```

- in your lab session login into moppsy.comp.glam.ac.uk and examine the contents of these files
- use `less /etc/services`
  - press ‘q’ to quit

GNU/Linux network administration command line cheat list

- extremely large subject so we give frequently used commands

```
ps -aux      # display all processes
kill <pid>  # kill a process number
find . -name 'foo' -print
             # finds a file called foo
find . -name '*.gif' -exec rm -f {} \; -print
             # finds all *.gif files & deletes them
df
             # displays all file systems that
             # can be seen from this machine
su
             # become superuser
```

- vi - an editor many people loathe
  - interface is lousy but `vi` is on *every* UNIX and GNU/Linux machine

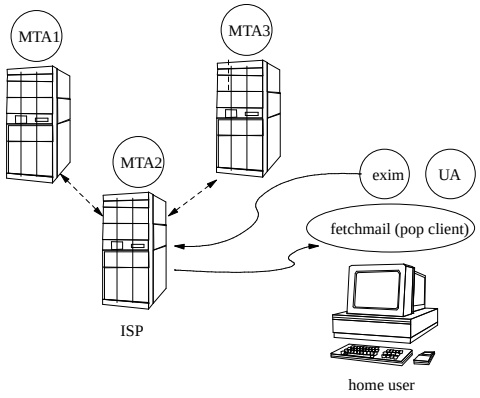
Configuring Email

- could be a very complex subject!
  - sendmail..
  - so we examine `exim` and `fetchmail` which are among the easiest mail programs to configure
- several TCP protocols exists just to handle mail:
  - SMTP see RFC-822
  - POP2, POP3, IMAP

Email architecture

- a mail system consists of three components
  - UA (User Agent) interacts with the end user
  - MTA (Message Transfer Agent) routes and transfers email to the correct destination machine
  - POP client
- there are many MTAs and different protocols (UUCP, SMTP)

Email architecture



Email architecture

- incoming mail messages are held in a **mailbox**
  - normally an ordinary file
  - the UA will then read this file on the users request
- generally a UA runs on the same host where the mail
  - however the POP (**Post Office Protocol**) allows you to read a mailbox on a remote system
  - why is this useful?

Email architecture

- useful as you may prefer to read your email on a PC which is not all ways connected to the network
  - in this case your messages are held on a reliable machine (one that stays on all the time)
  - down loaded to you PC via the POP when you want to read email

Configuring a home machine for email

- two different issues
  - sending email
  - receiving email
- consider configuration of a simple GNU/Linux workstation at home

Exim

- we primarily use it for sending email
- a simple replacement for the complex sendmail program
  - exim is a MTA - the standard MTA on GNU/Linux machines
  - to another MTA `smtp.freeserve.net`

Exim

- however all email sent from our home machine needs to have a falsified return address: `user@glam.ac.uk`
  - why?

Exim

- on installation asks a few questions
- we need to tell the configuration program
  - we do connect to the Internet via dial up
    - not connected all the time
    - not a stand alone machine
- smart host = `smtp.freeserve.net`
  - the MTA at `freeserve` which handles all `freeserves` incoming email
  - our out going email goes to the smart host

Exim

- visible name = glam.ac.uk
  - automatic reply to field is a glam.ac.uk
  - why?

Incoming email

- fetchmail
  - can be configured via a GUI ‘fetchmailconf’
  - or by writing the configuration file yourself!

```
poll pop.freeserve.net protocol pop3
     jbloggs.freeserve.co.uk
     password crackit is joe here
```


Practical observations

- ssh onto mcgreg.comp.glam.ac.uk and see which services are enabled in the file `/etc/inetd.conf`
  - can you see any services which are disallowed?
  - how are disallowed services identified?
  - hint you should google for documentation on /etc/inetd.conf for Linux or read the manual page, via `man 5 inetd.conf` at the command line
- now examine `/etc/hosts.deny` write down what services are available to other machines
- you should also consult the manual page for `/etc/hosts.deny` by using the command line `man 5 hosts.deny`
- now log into `moppsy.comp.glam.ac.uk` and do the same for that machine