Printing and Messaging

Peter Lo

Remote Printing

- Screen -> Hard copy at a remote location
  - Network connected printer
  - Remote fax
- Some issues
  - Type of interface on printer
  - Page Description Language

The two most common uses of networks (at least until the recent rise of the Internet) have been sharing files and sharing printers. Sharing files has been thoroughly covered over the past few weeks - sharing printers, in many ways a simpler issue, will be sketched out in the next section.

Sharing of printers can be characterised as the generation of output on printers that are not directly connected to the computer you are using. This definition is capable of extending to include shared printers on the local network as well as very remote shared printers (more commonly referred to as fax machines!)

There are two general issues that we will cover:

Printer interfaces

How is the actual connection to the printer itself implemented? The overall access to the printer is clearly through the network but we need to focus on the last step that is taken - getting data into the printer itself.

Page Description Languages

The need to transfer a complex image from the workstation to the paper in the printer brings with it the question of how the image is described. Various common approaches to this problem will be discussed.

Printer Interfaces

- Parallel interface
  - Fast single device
  - 8 separate data wires
    - 1 wire per bit - transmit a byte at a time
  - 8 -10m distance maximum
- RS232 serial interface
  - Single device
  - 1 data wire
    - slower - bits sent in series
  - Longer distances
- Universal Serial Bus (USB)
  - Multiple devices
  - Almost a network

The two most common uses of networks (at least until the recent rise of the Internet) have been sharing files and sharing printers. Sharing files has been thoroughly covered over the past few weeks - sharing printers, in many ways a simpler issue, will be sketched out in the next section.

Sharing of printers can be characterised as the generation of output on printers that are not directly connected to the computer you are using. This definition is capable of extending to include shared printers on the local network as well as very remote shared printers (more commonly referred to as fax machines!)

There are two general issues that we will cover:

Printer interfaces

How is the actual connection to the printer itself implemented? The overall access to the printer is clearly through the network but we need to focus on the last step that is taken - getting data into the printer itself.

Page Description Languages

The need to transfer a complex image from the workstation to the paper in the printer brings with it the question of how the image is described. Various common approaches to this problem will be discussed.

Page description languages

- How to tell the printer what to print
  - **Print driver** translates what's on screen into printer "language"
  - Many different mechanisms in printers
    - Need some common techniques
    - Simplify driver design
      - Re-use software
  - Most printers are "raster" devices
    - Page built in memory of printer
    - Page transferred to paper as dots

A **Page Description Language** becomes an issue when the variety of printer mechanisms is considered.

Traditionally printer behaviour has been controlled by including special control codes in the stream of data that is sent to the printer. A certain code might. for example, turn on italic printing. The problem with this technique is that different printers require different codes - hence the need for a **printer driver** which has the job of translating a request for italics from the word processor into the particular control code for the printer being used.

Since the creation of a particular driver for each model of printer is a big overhead there has been a tendency for printers to adopt one or more of the emerging standards for **Page Description Languages** which will be discussed in the next few slides.

Out of three such languages that we cover two recognise the fact that most printers are **raster devices.** This means that the ultimate image that is put onto the paper is generated as a series of dots - much like the image on a monitor. The raster-based languages therefore have the aim of producing. in printer memory, a dot by dot image of a page which can then be transferred to the paper. In the case of a dot-matrix or ink-jet printer you can witness this transfer multiple rows of dots at time whereas in a laser printer an entire page worth of dots is transferred to the drum before the entire page is printed.

Postscript

- Xerox invention
  - As was the GUI, IPX, SmallTalk…
- Complete programming language
  - Connect a terminal to a PS printer, login and start writing!
  - NeXT computer used PS for screen too
- Printer expects PS
  - Will not print "plain text"

The most widely established standard for a Page Description Language is **Postscript.**

Postscript is yet another of the inventions that came out of the Xerox labs in Palo Alto in the late 70's and early 80's. When the general nature of computing today is considered much of what we now take for granted can be traced back to Palo Alto. The use of a mouse and graphical user interface - which spawned the Macintosh and later Windows; the IPX protocol; the Smalltalk object oriented programming language and Postscript all come from the same stables.

Postscript is an entire programming language through which one can describe the complete layout of a page and then direct the Postscript printer to print the page.

To emphasise that Postscript is a complete language it is actually possible to connect a terminal to a Postscript printer, login to the printer, and then start writing and running programs. In days gone by the Postscript printer might have been the most powerful computer in many sites - hence the interest in hooking up and logging in!

In the 80's the NeXT computer even used Postscript to control its screen as well.

The one drawback to a Postscript printer is that expects print jobs to arrive in the form of Postscript "programs". This is not an issue if you are using a PC with Postscript printer drivers or a Macintosh which creates all print output in Postscript but if you want more basic print out - like **DIR > LPT1**

PCL

- Printer Control Language
  - Hewlett Packard development
    - Now more widely used by other manufacturers
  - Printer will print plain text
    - eg DIR > LPT1

PCL (Printer Control Language) was originally developed by Hewlett Packard to control their laser printers.

The development of a Page Description Language is an admission that simple sending a sequence of bytes with codes embedded in it may not be flexible enough to describe the more complex features of a printer and also that printers are becoming smarter and smarter and can be given more of the work involved in printing.

A good example is the way that the lab sheets are printed out - two pages per sheet. The ability to shrink whole pages of output is now built into the printer and a Page Description Language is clearly needed to control such a feature. Basically we can say to the printer - “here is a description of some pages - now render them two to a sheet” (or four, or eight…)

The HP PCL has become a defacto standard which many other manufacturers are adopting for their printers. This language has the advantage of understanding “plain text” output such as **DIR > LPT1.**

HPGL

- Hewlett Packard Graphics Language
- Language to driver plotters
  - Plotters are “vector”` devices
    - They “draw” the image onto the paper
      - PU, PD
  - Plotters expect serial connection
    - Can be a problem in a network

HPGL (Hewlett Packard Graphics Language) stands all on its own as being a language for controlling plotters which are **vector devices.**

Plotters produce output much as we humans do. They draw a line, lift up their pen, change colour, put their pen down and draw another line until the whole page is complete. Each line is considered a **vector.**

Because of the fact that a page consists of a whole series of commands (such as **PU - Pen Up** and **PD - Pen Down**) which must be executed in sequence plotters have tended to have serial interfaces and also expect to be able to interact with the program doing the plotting.

This can make it hard to share plotters through a network.

Network vs Local Printing

- Share printers
- Multiple access
  - Type of connection to printer
    - Local - serial or parallel cable
      - Need computer to make network connection
    - Network direct to printer
- Queue jobs
  - Spooling
    - File storage involved

When a printer is shared through a network there are a new set of issues to consider

In some way there needs to be a way for multiple users to access the printer. This shared access has two aspects:

Getting connected

Given that printers in many cases are equipped for point-to-point connection how can multiple users send their jobs to a printer? The answer is that the device at the other end of the printer serial or parallel cable must be on the network. This "device" might be a client PC, a server PC or even a specialised modem-like box.

There is a growing tendency for printers to have their own NICs built in in which case they are fundamentally accessible through the network.

Time sharing

If my job is currently being printed what happens to your job? This underscores the need for some kind of queue of jobs to be associated with the printer in order that clients to not have wait to complete the process of sending the job.

Printing becomes broken down into two steps - sending the job - known as **spooling** and the actual transfer to paper which we will still refer to as printing!

Spooling requires storage space for the print job - this may be provided on an intermediate computer or the printer itself may have a hard disk.

Making the Connection

- Redirection
  - Like drive mapping
  - Local resource points to remote printer/queue
    - Win95 - object in Printers folder
    - DOS - Capture
      - Local resource is LPT port (LPT1, LPT2 etc)

The client that needs to use a remote printer has a choice of strategies for requesting the service.

In the DOS/Win95 environment the same concept of **redirection** is applied to printing as was used in drive mapping - local printer requests that would have gone to a printer port like LPT1 are redirected in software to a remote printer.

In actual fact the redirection sends the spooled output to a **print queue** where the job is stored until the printer is ready.

Making the connection….

- Explicit remote request
  - Like FTP
  - UNIX
    - LPR sends job to remote printer
    - LP**D** is the daemon that receives the job & prints
  - Appletalk
    - Find printers on the wire when they are needed
      - Original MAC & LaserWriter had network connection

An alternative way to access a shared printer is to make an explicit request for remote printing. This is the technique used in Unix and Appletalk.

The significance of the "explicit request" can be understood if we consider FTP where it is built right into the program that something is being accessed through the network.

Unix Printing

The client uses the **LPR (Line Printer)** protocol to send the print job to a waiting server. The server is running a program called **LPD (Line Printer Daemon)** that waits for LPR requests, spools jobs and then sends them to a printer.

Appletalk Printing

The Chooser program which we saw being used to select remote printers sets up an explicit connection to a remote printer. If queuing is required than a computer (a **print server**) with storage available needs to respond to NBP requests for a printer - under these circumstances the printer needs to be told to keep quiet because we want print jobs to be intercepted by the print server.

Spooling for speed

- Quick “printing”
  - Quicker to create file than print the job
  - Intermediate file format still quicker
    - W95/NT - Printer Metafile
      - Generic description of output
      - Needs further translation for specified printer
        - Where to do this?

Spooling has already been justified as a means to implement the queue of jobs that is to be sent to a printer but spooling also has performance advantages even for a single user of a printer. This is because it is generally quicker to write to a file than to wait for ink to be transferred to a page!

Even the process of creating a printer-ready file can be considered too slow so Microsoft, in recognition of this, have defined an intermediate file format called the **Printer Metafile.** Printer Metafiles are quicker to write and are device independent - meaning that they are not tied to a particular type of printer.

The client can therefore finish even sooner with spooling but the print job requires further “translation” work before it can be sent to a printer. This work would be undertaken by the computer that we have described as the **print server**.

Spooling storage

- Large but temporary storage
  - Hard disk somewhere
    - On a shared computer
      - server or peer
    - On the printer itself
- Security
  - Who can do what to which print jobs

Spooling storage, especially where highly graphical output is concerned, can imply that quite large files need to be created on a temporary basis.

The storage for these files might be on a server, on a peer computer or even in the printer itself.

As soon as storage is set up there are also security issues relating to that storage - who can do what, and to which files in that storage?

What is a “printer”?

- NT refers to the physical thing as a “print device”
- Need a name for the software that provides spooling and possibly translation
  - NT
    - Printer
  - Novell
    - Printer Agent

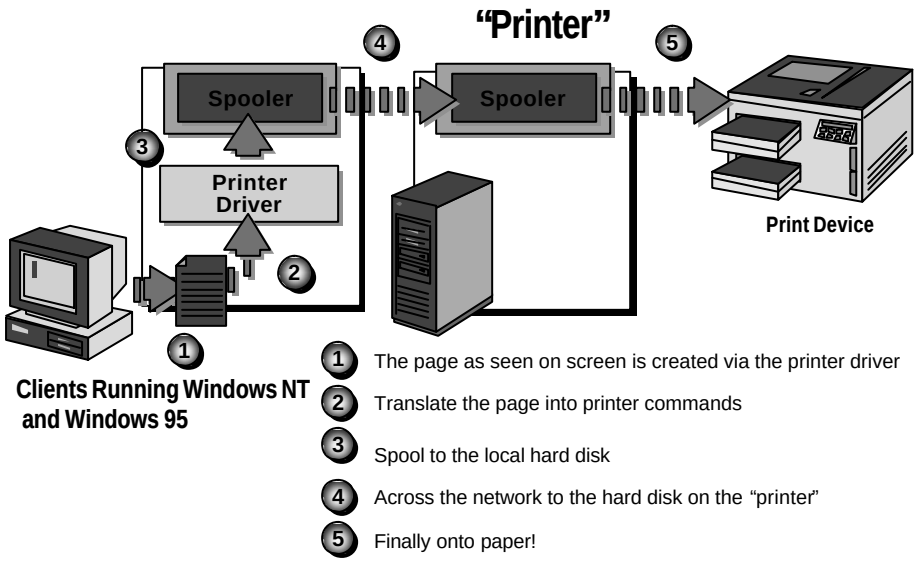
There is some substantial “terminology drift” in the field of network printing and once again Microsoft is at the forefront in telling us that the old way we used these words is no longer good enough.

The Microsoft printing architecture uses the word **printer** to describe the device that receives print jobs from clients. In other words, in Microsoft jargon, when we spool we spool to a **printer.** With the word “printer” hijacked in this way Microsoft has to refer to the things with paper and ink in them as **print devices.**

In the Novell printing architecture the Microsoft “printer” has the slightly less confusing name of **Printer Agent.**

These two architectures are spelt out in the next two slides.

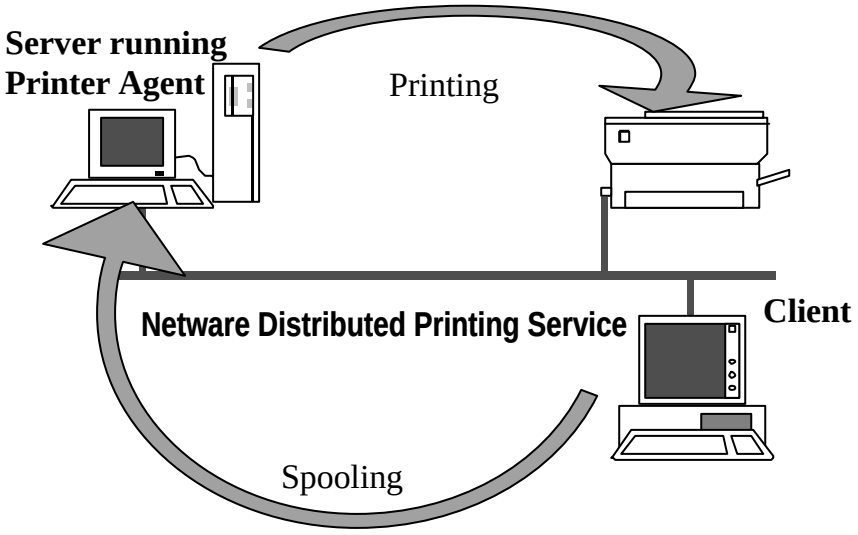
Windows Printing Architecture



1. The page as seen on screen is created via the printer driver
2. Translate the page into printer commands
3. Spool to the local hard disk
4. Across the network to the hard disk on the "printer"
5. Finally onto paper!

This animation shows the progress of a print job from a client via what Microsoft call the "printer" and finally to the actual print device.

Novell Printing Architecture



In the Novell printing architecture, **NDPS (Netware Distributed Printing Service)** the spooling service is provided by the Printer Agent which is a piece of software that either runs on a server or is sometimes implemented within the printer.

Other issues

- Managing print jobs
  - Scheduling
  - Removing
- Printer control
  - Bi-directional communication
    - "I'm out of toner"
- Providing up to date drivers
  - Novell RMS (Resource Management Service)
  - Microsoft Print Server

Most of the focus so far has been on the actual act of printing but in a distributed printing environment there are several other kinds of task that need to be attended to:

Managing Print Jobs

Jobs that are waiting in the queue to be printed can be administered in various ways. Unwanted jobs may need to be deleted and the priority of a job can be changed so that it gets printed sooner (or later)

The privilege to perform these kind of tasks is generally quite separately managed from privilege in the file system.

Printer Control

When printing on a locally attached printer the potential is there for interaction between the user and the printer during the printing of the job. The printer might, for example, signal the user that it is out of toner.

In a shared environment this kind of communication is more difficult to organise because the job is being sent to the printer by the server at some point later in time due to the queue. The user that sent the job may even have gone home!

The NDPS Printer Agent attempts to implement two way communication between the printer and the user by relaying messages to and fro.

Updating Drivers

When many users share access to a number of printers installing new or updated printer drivers on workstations can be a major maintenance task.

UNIX Printing

- lpr
  - Sends a job
  - Usage:
    - lpr -P<name of printer> <filename>
- lpd
  - Receives the job
    - Spools
    - Prints

Unix printing relies on two components:

lpr

Sends a print job to a named printer

lpd

Runs at the server and receives (spools) print requests.

These two components communicate via the LPR protocol.

Messaging

- Summarise messaging services and provide examples
  - a broad view of *messaging*
- Focus on email
  - More detailed view of a specific messaging service
  - largely UNIX-based

Messaging is a broad subject and broadly used - it is probably one of the main things that the Internet gets used for today. I think I read that 3 **trillion** emails were sent last year.

We will take a general look at the rich variety of mechanisms that can be classed as messaging and then take a closer look at the most widespread messaging - email.

Many faces of Messaging

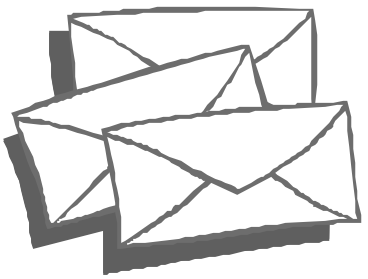
- Messaging services
  - *Delivering Data*
    - *text,*
    - *binary*
    - *graphic*
    - *digitized video*
    - *audio*
  - *Between human users / computer systems*

In the widest sense messaging refers to the delivery of data from one point to another. The sender and the receiver may be humans or computer systems - the mechanisms of messaging that have evolved to service humans are finding widespread application to enable distributed computer systems to operate.

The data can take many forms from the simplest text only message (still the most widespread) through to digitised video.

Email: first look

- Email
  - Initially confined to text
  - Today, more multimedia potential
  - Involves
    - Transportation
    - Presentation



In considering email it s important to realise that simply delivering the mail may be only part of the problem.

This component, the transportation of the mail, involves specialised protocols that sit on top of the Transport layer protocols that we have already looked at.

However there may well be some other issues involved in relaying mail between different computer systems. We had a small glimpse of this in NFS last week.

Email and Voice-mail

- Email and Voice-mail integration
  - The generalised nature of computers as tools is evident in voice-mail services
    - Voicemail systems are computers
  - There is interest in developing voice-mail systems that will work on networks, and integrate with email
    - Audio could be converted to text
  - Alternatively, email could be accessed via phone systems
    - Voice synthesisers

Email is beginning to be integrated with voice mail.

Both systems use computers to manage the storage of messages but the relay is via networks in the case of email and telephone in the case of voice mail.

If there is a point of contact between these two systems - a single computer that can make and receive both kinds of mail then some interesting possibilities arise:

Receive voice mail as email

At the very least a user can be alerted to incoming voice mail via email. The sophisticated forefront of voice activation and input for computers is evolving the technology that could actually deliver the content of the voice mail as email.

Receive email as voice mail

If no computer is to hand then a user might want to read their email over the phone. This technology is more straightforward - we just need a voice synthesiser.

Object-Orientation

- Object-Oriented applications
  - Typically many small objects with cohesive functionality, communicating with each other as needed
  - Objects on a network may use messaging services to facilitate this communication
  - An agent from the local messaging service assumes responsibility for conveying the message to the target object

As the range of message types and delivery mechanisms multiplies, and the number of messages keeps increasing it becomes more and more time consuming for humans to sift through their electronic in-trays.

The "Object" concept is hinting that messages may flow back and forth between intelligent agents that act on our behalf and filter the flood of messages in various ways.

A good example of this is to consider the needs of the user that is having their email electronically read to them over the phone. At the very least the agent handling the mail should skip file transfers and digitised video messages. What is really needed is a means to specify filtering rules in advance that will be selectively applied when mail is read through the phone.

Workgroup Applications

- Workflow Management
  - Intelligently routes forms and documents
  - Forms and documents are customised to individual needs
  - Replaces cumbersome manual methods of workflow practice
    - Forms to fill in
    - Verification procedures by managers
    - Processing and checking
  - Messaging services enable the workflow routing and delivery decisions

Workgroup Applications which enable organisations to electronically reproduce procedures and sequences of actions that are currently based on paper and forms clearly rely on many kinds of messaging,

The intelligent agents of the previous slide may well have a large part to play here.

Linked Object documents

- The linked-object model
  - Documents consisting of multiple components
  - Resembles the approach taken by OO applications
  - Independent data items are brought together within the document
    - May be video, text, graphs, voice, spreadsheets, etc.
  - Each object uses messaging services agents to communicate

Documents that previously consisted of text with some formatting may now contain a variety of digital objects from various sources.

These objects may well have their own independent needs for messaging resulting in a compound document that has sophisticated messaging requirements in its own right.

MHS-Message Handling Service

- MHS is part of the ISO protocol stack
  - TCP/IP is the protocol stack we have looked at most
  - ISO is another protocol stack
    - you will encounter this again later in the unit
- It is variously known as
  - MHS
  - X.400 MHS
  - MOTIS (Message-Oriented Text Interchange System)
  - ISO/IEC 10021

MHS - Message Handling Service is a complete set of protocols that implements messaging. MHS is part of a family of protocols that we have not looked at yet - the ISO (International Standards Organisation) Protocol Stack.

ISO defined a 7 layer model of networking that we will look at more closely in the last lecture. The Data Link Layer mentioned in Week 7 to refer to Local Delivery was an ISO concept. The layers are **not** actual protocols, just principals according to which real protocols can be designed.

However the involvement of ISO with networking did not end with the design of the 7 layer model. The ISO has moved on to design actual protocols to implement each layer and in the Application layer to solve practical networking problems.

In this unit we will consider a couple of ISO Application layer solutions that have very widespread acceptance in relation to Messaging and later in relation to Directory Services.

The lower layer protocols from ISO are not nearly as widely used as TCPIP - a set of de-facto protocols that has become almost ubiquitous. The large computer company Digital set up one of the earliest global networks to interconnect its main frame and midi computers and the DNA (Digital Network Architecture) now makes very extensive use of ISO protocols.

MHS Components

- User Agents (UAs)
  - Compose messages, send them, read them, notify arrival, local storage and management.
- Message Transfer Agents (MTAs)
  - Forward messages between UAs.
  - Where immediate delivery is impossible, temporary storage may be used.
  - Failures reported when they occur.
- Message Transfer System (MTS)
  - Combined MTAs

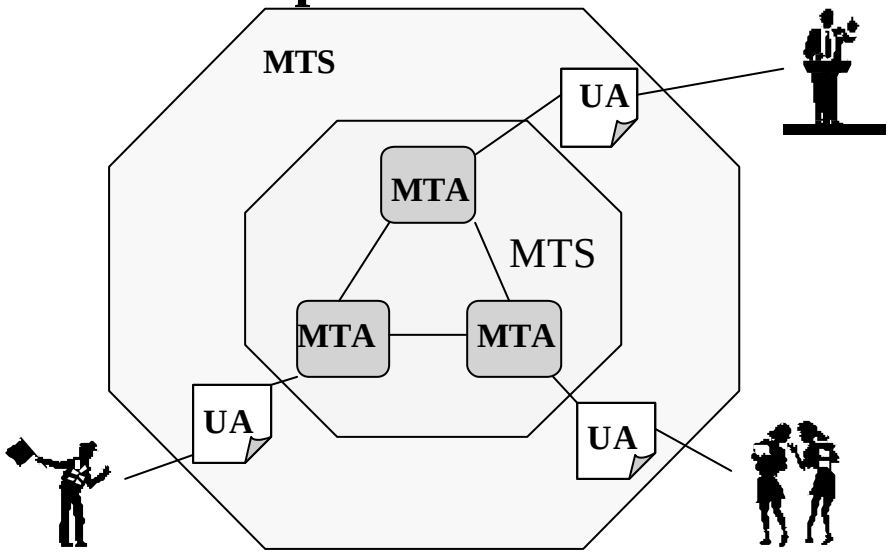
The **architecture** of MHS is based three concepts.

In terms of messages it is **User Agents (UA)** that originate and receive messages. Obviously there will often be an actual human user involved where there is a UA but with the kind of automated agents that we discussed earlier this is not necessarily the case.

The UA, when it dispatches a message, does so by communicating with a **Message Transfer Agent (MTA).** MTA's interact with each other to relay the message to its eventual UA destination.

The combined set of MTA's is referred to as a **Message Transfer System (MTS)**

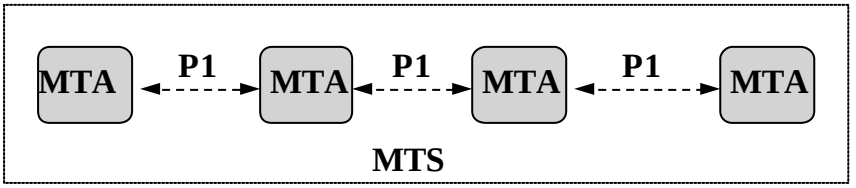
MHS Components



This slide shows the interaction of the MHS components.

MHS Protocols - P1

- MTA's use a *P1* protocol to control the transfer of messages around the Message Transfer System.
- Of course, ultimately this rests on other available protocols.
  - TCP
  - SPX etc

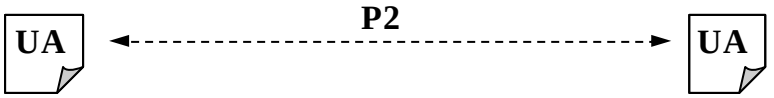


MHS defines protocols for the various steps involved in the relay of the message.

A **P1 protocol** is one that is used in communication between MTA's. Beneath P1 protocols there are, of course, lower layer protocols like TCP or SPX.

MHS Protocols - P2

- P2 protocols are used to ensure that one UA can "talk" to another UA so that the message has the same meaning at both terminals
- This is a "Presentation" layer issue
  - An extra layer that we have not yet talked about
- Message structures differ
  - The one used for simple person-to-person messaging is the *interpersonal messaging (IPM) protocol*



The "end-to-end" relay of the message may involve many issues to do with differences between the sending and receiving systems.

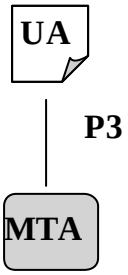
These types of issues are what are generally handled by what is termed the "Presentation" Layer of the 7 layer model. The concern in this layer is no longer the transfer of data between computers but the interpretation of that data once it has reached its destination.

In terms of MHS the management of these kinds of issues is managed by **P2 protocols.**

Different types of message require different types of structure to manage them - hence there are a variety of P2 protocols - simple text email uses IPM (Inter-Personal Messaging) protocol.

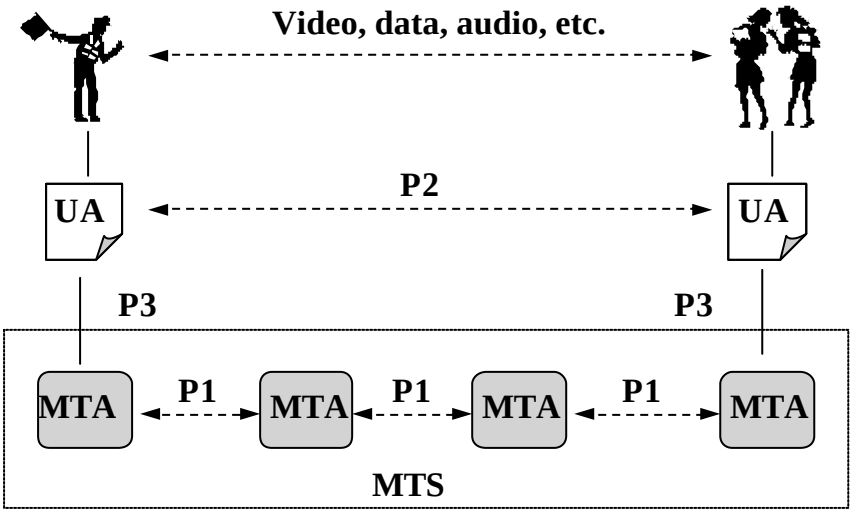
MHS Protocols - P3

- A *submission and delivery protocol* is used to interface the UA and the MTA.
  - Handles submit/deliver/charge functions.



The missing link once P1 and P2 have been defined is from the UA to the MTA. This step is managed by **P3 protocols.**

MHS protocols

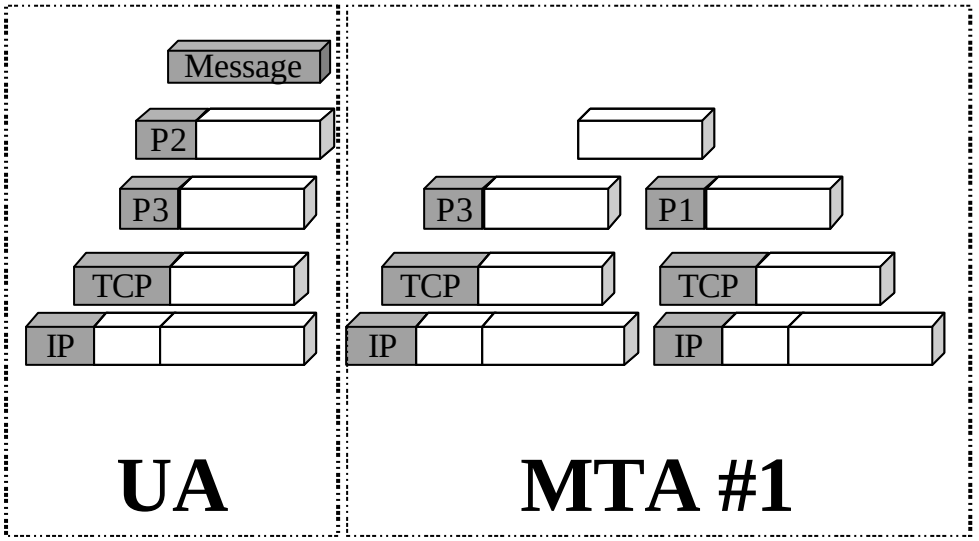


This slide shows the way that P1, P2 & P3 protocols fit together in the end-to-end relay of a message.

P1 and P3 are shown exactly where we would expect them but the representation of P2 needs some explanation.

P2 is used to relay presentation related information between **peer** Presentation layers at the two ends of the conversation and hence is shown in dotted lines between the sender and the receiver.

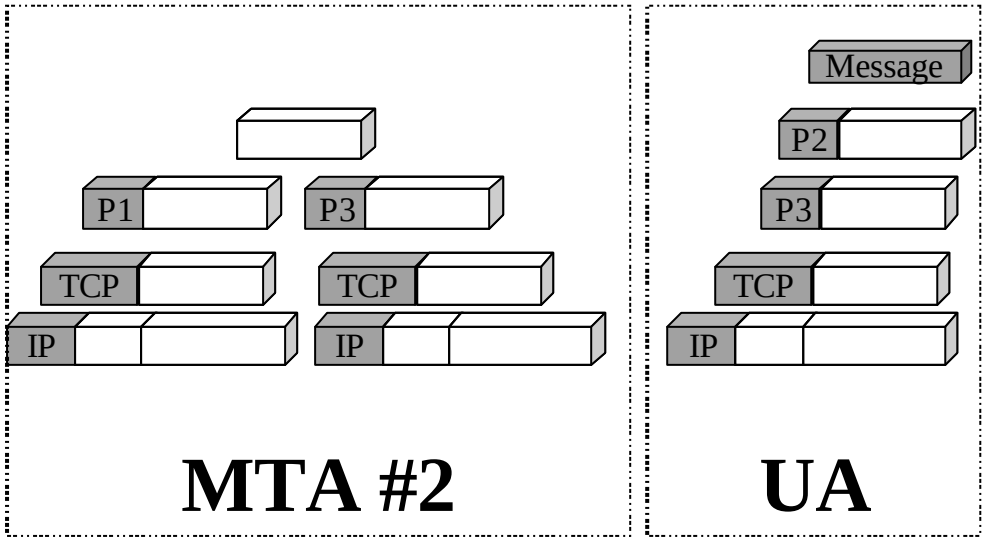
MHS in layers



The interaction of these protocols becomes clearer if we look at the interaction of the layers as we did in earlier topics.

This slide - and the next one - shows the message being encapsulated within the various headers needed by the MHS, and other protocols.

MHS in layers

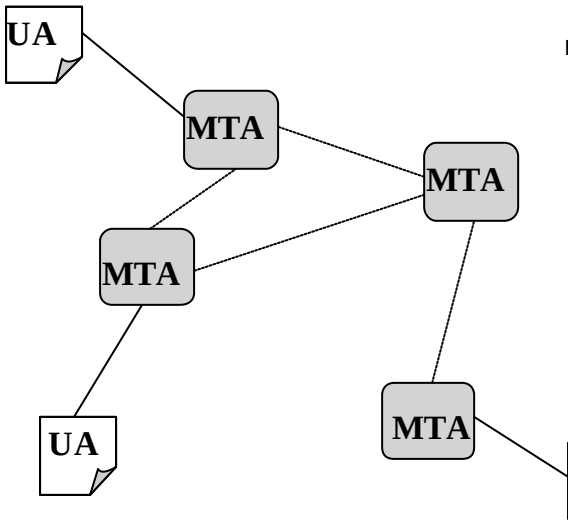


The most important thing to keep in mind is the way that **peer** layers interact and this is most clearly shown with the P2 protocol.

MHS Summary

- What messaging services are
  - Some examples
  - The typically modular nature of a networked services such as messaging, as shown via MHS
  - A reinforcement of the concepts of layering and peer-to-peer communication via the MHS example

Review: MTAs, UAs, etc.



- MTAs
  - must support the same mail transfer protocol in order to communicate
    - P1
    - SMTP
    - etc.
  - they may support *multiple* protocols

The MHS concepts of UA and MTA can be applied in general to messaging systems - the basic requirement is that MTA's must support the same "P1" protocol in order to communicate.

The most widespread protocol at this level is **SMTP (Simple Mail Transfer Protocol)** which is the protocol used by Internet email.

It is possible for MTA's to support multiple protocols and therefore to act as gateways between different messaging systems.

MTA/UA Division of Labour

- MTA
  - routing
  - address validation
- UA
  - user interface
    - ensure that composing, sending, reading, accessing mail is easy.

This enables mail that adheres to a standard format (*such as RFC822*) to traverse many different networks, each of which may use different mail transfer protocols, yet still arrive at the user agent in a coherent form.

MTA's and UA's have distinct tasks in the messaging architecture.

The UA takes care of all user interface issues and sets out to make the processes of composing, sending. reading and accessing mail strightforward.

MTA's take complete responsibility for routing mail and validating addresses.

In order for a UA to send or receive mail the UA needs to be in touch with one MTA that becomes their entry point into the network of MTA's. This is why (until they invented **hotmail**) you had to have an account with an ISP in order to participate in email. The ISP account is basically permission to access an MTA.

hotmail (and the other free, web-based email solutions) are worth thinking about in terms of UA/MTA's. **hotmail** is an enormous MTA with 40 million accounts. The cluster of **hotmail** servers implements the MTA side of things but it is also the UA - since the Web forms that customers use to access email are generated by and submitted to the **hotmail** computers.

Internet Mail

- Simple Mail Transfer Protocol (SMTP)
- Based originally on Unix
- Now very widely implemented

RFC-822

- The mail data consists of a header and a message body, separated by a blank line.
- the header includes colon terminated fields such as:
  - *To:*
  - *From:*
  - *Subject:*
  - *Date:*
  - *Received:*
    - *added to by intermediary MTAs. It is a form of "postmark"*
  - *Cc:*

RFC (Request For Comment) 822 defines the format of a standard email.

A word first on **RFC's.** These documents are the design basis of most of the protocols and services that we now use on the Internet - as many of you will be realising by now as you proceed with your research.

To begin with an RFC is a proposal for some new functionality in the Internet - whereupon the Internet community (as indeed it was in the early days) evaluates the idea and makes comments. Through this peer-review system a refined version of the proposal would evolve that would them be implemented.

RFC 822 has a basic definition of a message as having two parts - a body and a header.

The header has a variety of fields - each preceded by the field name with a colon.

The **Received:** field is added to by each MTA along the way (turn on "Show All Headers" in Pegasus Mail to see the steps that the message has taken in reaching you - more recent steps are shown first)

Mail Delivery and Routing

- Local MTA runs as *root* on UNIX, allowing it to append incoming mail to files owned by mail recipients.
  - /usr/mail/
  - /usr/spool/mail/
- Mailboxes are files with the same login name as the owner
- permissions are set to allow read and write to the owner (*only*).
- The UA copies mail to a temporary file for reading.
- This enables file locks to be released and further delivery by the MTA.
- When sending mail, MTAs typically check the *userid* of the sender, and put it in a *"From:"* line.
- Also adds a *"Received:"* line (with datestamp).

UA/MTA communication is based on reading and writing shared files on the MTA. This is true of most popular email systems - Unix SMTP based mail, Microsoft Mail and Novell-based Pegasus mail. The consequence of this basic design decision is that email is often a point of attack for hackers trying to break into a system - here are directories or files to which a large amount of extra privilege has been granted - what better place to start the attack?

The steps taken in causing an MTA to initially send of an email message are considered one by one in this slide.

SMTP

- uses the reliable delivery service provided by TCP
- It uses a "well-known" port number of 25 on the server
  - this is *well-known* so that remote services know what transport-level addressing to use
  - SMTP uses ASCII
  - by *telnetting* to port 25 you can use SMTP in a "raw" form.

telnet krause 25

- Some commands:
  - helo
  - help
  - mail
  - rcpt
  - data
  - quit
  - vrfy
  - expn

The MTA in Unix is accessed via the **Simple Mail Transfer Protocol (SMTP)** which is a simple text-based protocol that has a series of four letter words as its commands.

To make SMTP work all that needs to happen is that the commands and the responses have to pass back and forth between the UA and the MTA. The Telnet protocol is ideal for sending and receiving such command and responses. Hence the MTA listens on port 25 and then does what it is told.

You can talk directly to the Unix MTA by using:

telnet krause 25

and the messaging lab will be based on learning how to use this access.

SMTP - sending email

```
$ telnet krause 25
Trying 141.132.64.13...
Connected to krause.ballarat.edu.au.
Escape character is '^]'.
220 krause.ballarat.edu.au ESMTP Sendmail 8.8.8/8.8.8; Thu, 24 Sep 1998 14:33:19
 +1000 (EST)
helo mycomp.thing.com
250 krause.ballarat.edu.au Hello krause.ballarat.EDU.AU [141.132.64.13], pleased
 to meet you
mail from:<dave@thing.com>
250 <dave@thing.com>... Sender ok
rcpt to:<d.stratton@krause.ballarat.edu.au>
250 <d.stratton@krause.ballarat.edu.au>... Recipient ok
data
354 Enter mail, end with "." on a line by itself
Some exciting news...
.
250 OAA05956 Message accepted for delivery
quit
221 krause.ballarat.edu.au closing connection
Connection closed by foreign host.
```

Note that krause *could* (but doesn't) verify this. Most systems do. What information is available to check?

This slide shows the transcript of a **telnet** session to the SMPTP port on **krause**

In this sequence the commands entered by the user are in bold type - they need to be recognised SMTP commands and arguments to those commands.

Notice that the SMTP server accepts anything that you say to it - with no authentication or evaluation of who is sending the message. This explains why there is so many problems with junk email and why so much of the junk email that you get cannot be replied to.

In the end, when the message has been accepted for delivery it is given a unique ID by the SMTP server which means that it can be traced through the MTA's that it passes through on the way to its destination.

Email received

From dave@thing.com Thu Sep 24 14:35 EST 1998
Received: from mycomp.thing.com (krause.ballarat.EDU.AU [141.132.64.13])
by krause.ballarat.edu.au (8.8.8/8.8.8) with SMTP id OAA05956
for <d.stratton@krause.ballarat.edu.au>; Thu, 24 Sep 1998 14:34:11 +1000 (
EST)
Date: Thu, 24 Sep 1998 14:34:11 +1000 (EST)
From: tony@thing.com
Message-Id: <199809240434.OAA05956@krause.ballarat.edu.au>
Content-Type: text
Content-Length: 35

Some exciting news...

$

Here is the message as received - using the Unix mail program on **krause** (although it could have been viewed in Pegasus mail just as easily)

vrfy, expn

- vrfy <login>
  - used to *verify* user details
  - if “login” is the local name of the user, then basic account information is returned.

For security reasons, both of these SMTP commands are commonly disabled.

- expn <alias>
  - used to *expand* aliases
  - aliases may be lists of users which need expansion, or single users which have multiple mailing names
  - as lists may consist of other lists, there is a recursive nature to this which halts after a certain depth.