# File Sharing

Peter Lo

# File Sharing

- What is it?
- How is it different from File Transfer
- How it it done?



This lecture we move away from the topic of **transferring** files to the notion of **sharing** them.

To begin with the distinction between these two approaches to file access will be made clear before we move on to look at two models of file sharing - the one used by Novell and the one used by Microsoft.

We will pay special attention to the Novell model of file sharing because a clear understanding of this helps students in next semester's Network Operating Systems unit.

# Analysis of File Transfer

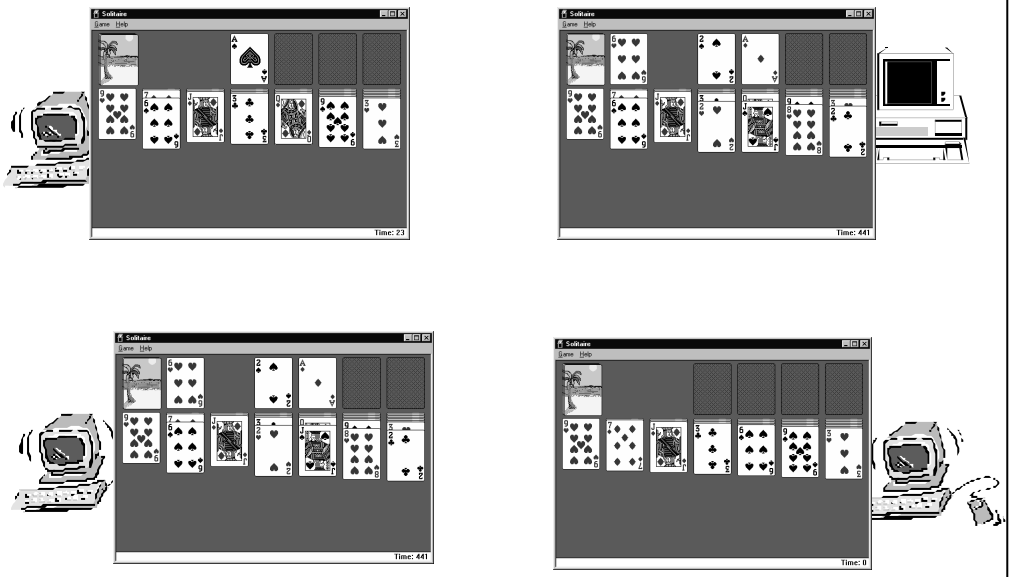
- Copy files from one node to another
  - How many versions exist after one transfer?
  - 2, 3, 4 transfers…..
- Protocols we saw in TCP/IP file transfer
  - FTP
  - TCP
  - IP
  - Ethernet



The File Transfer we have looked at has been the TCPIP FTP protocol. Each time you FTP **get** a file from the server there is a new copy of the file in existence and it you were to follow this with a **put** you would overwrite whatever the previous use had **put**. There is no concept in the way that FTP looks at files of two users having concurrent (i.e. simultaneous) access to a file.

How did FTP do its stuff? Well you should get used to sentences like “FTP over TCP over IP over Ethernet” because that is a concise definition of what we were looking at last week.

# Transfer may not be OK...





This animation sets out to show why sharing and transfer are quite different things.

Consider a network card game in which several users take turns in a game of solitaire.

This implementation uses **transfer** to send the state of the game between users and, as can be seen, a lot of the users do not see the more recent changes.

# Function of Protocols

- FTP
  - establish session
  - transfer files
- TCP
  - Reliability
- IP
  - Routing
- Ethernet
  - Local delivery

- Application Layer
- Transport Layer
- Network Layer
- Data Link Layer



This slide is just some revision of last weeks material and reminds us of the different tasks performed by the different protocols.

These tasks fall into the definitions of what needs to be done by each of the four layers of networking that were being used. This is the view of networking that we are moving towards using and the terminology that we will come to rely on more and more.

Remember what **encapsulation** means and how it responsible for the practical implementation of this layering - and extra layer in the networking model simply means an extra layer of encapsulation in the packet.

# File Sharing

- Common files are made available to all users
  - Single copy of database
  - One network copy of an Application
- Require Network protocols to facilitate



The contrasting approach to remote file access - using files that are actually stored on someone else's hard disk - is to make these files simultaneously to multiple users.

This is clearly the only way that we could setup database access for example. If there is a database (say of customers) that an entire organisation needs to share it clearly makes no sense at all to have individual users **copy** the data to their machines before operating on it. When the file was copied back by one user they would overwrite the changes made by the previous user.

There needs to be a means to allow multiple users to have **concurrent access** to a file - we need protocols that facilitate **file sharing.**

# File Sharing Environments

- Microsoft NT/2000
  - Client/Server or Peer-to-Peer
- Microsoft Win 95
  - Peer to Peer
- Novell Netware
  - Client/Server
- AppleShare
  - Peer to Peer
- Unix NFS
  - Network File System



File Sharing is such an important concept that all the major networking systems provide a file sharing environment.

This slide lists four major environments and shows whether they are peer-to-peer or client/server.

In deciding on whether a scheme is client/server or peer-to-peer you need to think of the big picture. Clearly, during any one particular file access on machine will be the server and the other the client (one machine has the file on hard disk and the other does not). The point to consider is whether the client can change roles for the next file access and begin to share one of its files with machine that was being the server.

It is this **changing of roles** that defines peer-to-peer.

# Novell Netware File Sharing

- Aim
  - Files located on Netware server behave like local files
  - Cases to consider
    - DOS (Pre Netware)
    - Windows



If you are a Mt Helen student the major focus here will be on Novell's approach to file sharing - this rather one-sided look at the issue is designed to make your life easier in the second year **Network Operating Systems** unit where you will be learning to configure access to a Novell (and also Microsoft) file server.

The aim of the file sharing that both Novell and Microsoft provide is make files on a server behave as if they were local files.

This requires a different approach for older DOS based clients as compared to a modern Windows client.

# Microsoft File Sharing

- Aim
  - Files located on Microsoft server or peer computer behave like local files
  - Cases to consider
    - DOS (Pre Windows)
    - Windows



For other students the major focus will be on Microsoft's approach to file sharing.

The aim of the file sharing that both Microsoft and Novell provide is to make files on a server behave as if they were local files. The approach that is used is influenced by the way that the modern operating system has developed from DOS.

# Local File Access

- Typical file operations include
  - Open a file
  - Close a file
  - Write to a file
  - Read a file
- Call to OS kernel
  - Request operation on file

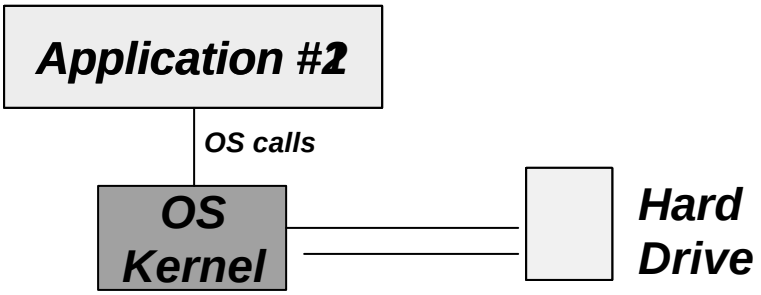


To understand the issues involved in remote file access we first of all need to get clearer about how computers access the files on their local drives.

Typically a program might need to perform one of the operations shown on a local file and the way that the program makes one of these things happen is by calling a routine in the kernel of the operating system.

# Universal access...

- Any application
  - Written in any language
- On any hardware
  - That runs the OS






The goal - the whole reason for having an **Operating System** - is to make it possible for any program, whatever language it is written in, and whatever computer it was developed on, to access files on our machine.

The OS produces a standard interface to files on the hard disk that is completely and utterly the same on hundreds of millions of computers around the world.

In the animation we see different applications accessing files on different hardware but the OS remains the constant element.

# How does the OS access files?

- Uses drive letters
  - A: Floppy
  - C: Hard Drive
  - D: CDROM
  - E: What more hard drives?
    - May be a partition



This slide reminds us that the OS on our computer at home uses **Drive Letters** as the first step in describing a file that we want to access.

Each letter is used to refer to a different major storage region.

# What about remote files?

- **Must** make sense to the OS
  - Early OS's could only use drive letters
- Use drive letters for remote access
  - A: Floppy
  - C: Hard Drive (local)
  - F: to Z: pointers to remote file system



When we consider the task of accessing a file that is stored remotely (e.g. on a server) we are constrained by the abilities of the operating system. Until fairly recently the operating system on PC's (DOS) could only understand drive letters as the means to begin the description of where a file is stored.

Consequently remote file access needed to managed in terms of drive letters. Fortunately there are quite a few letters in the alphabet.

These letters (in general anything that is not being used locally so in most cases F: through to Z:) are **pointers** to places on the server. Local drive letters are being used as pointers to the remote file system.

# Typical DOS Commands

- *TYPE C:\FILE1.TXT*
  - Access file on hard drive with OS calls
- *TYPE K:\FILE2.TXT*
  - Accesses file on server but must somehow generate some packets!



We can sue these pointers to access files **exactly as if they were stored locally.**

The two command shown have exactly the same effect - except that one of the files comes from the server. Somehow we have to arrange for some packets to flow on the network!

# Microsoft Solution

- Workstation requires:
  - NIC
  - Appropriate software
    - Application
    - Transport
    - Network
    - Data Link



To access files (using drive letters like "K:") on another computer the workstation needs a NIC and the appropriate software to give it the layers through which it can access the server.

Remember that for Microsoft the other computer might be a server (running NT or Windows 2000) or another workstation – running W95/98/ME etc.

The OS needs to provide all the various layers of networking software:

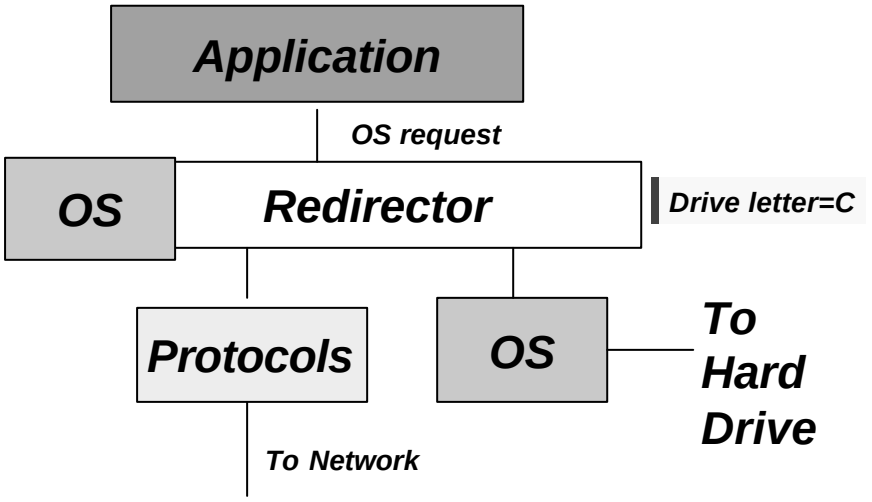
•Application Layer – so that programs can treat remote files as local

•Transport – so that the connection with the remote machine can be reliable

•Network – so that the remote computer can be anywhere in the world

•Data Link – everything depend on this

# Redirector

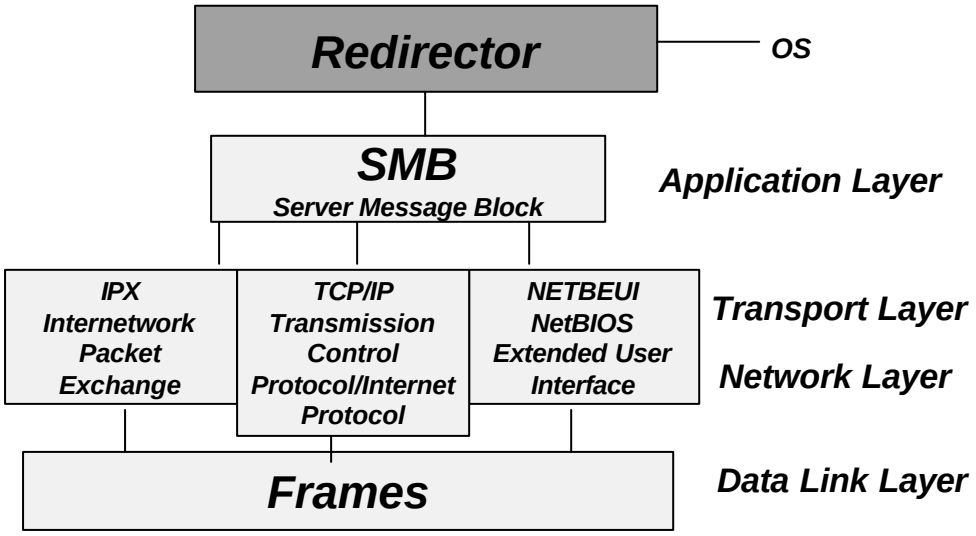





Assuming the software is installed at the workstation the procedure for getting a file from the server (rather that the local drive) is based on a trick played in software by a component known as the **redirector.**

The redirector inserts itself into the chain of command at the workstation so that it sees file access requests before the kernel of the OS. Any requests which are seen to be local (by inspecting the drive letter) are passed on to the OS whereas requests for files on remote drive letters are passed down through the appropriate protocols and across the network.

# Microsoft protocols






Microsoft sets out to fit into any networking environment and this is demonstrated by the amount of choice that exists below the Application layer.

At the Application layer all requests use the SMB protocol.

At the Transport and Network layer it really depends on how you have configured your networking in the Network Control Panel.

The three basic options are:

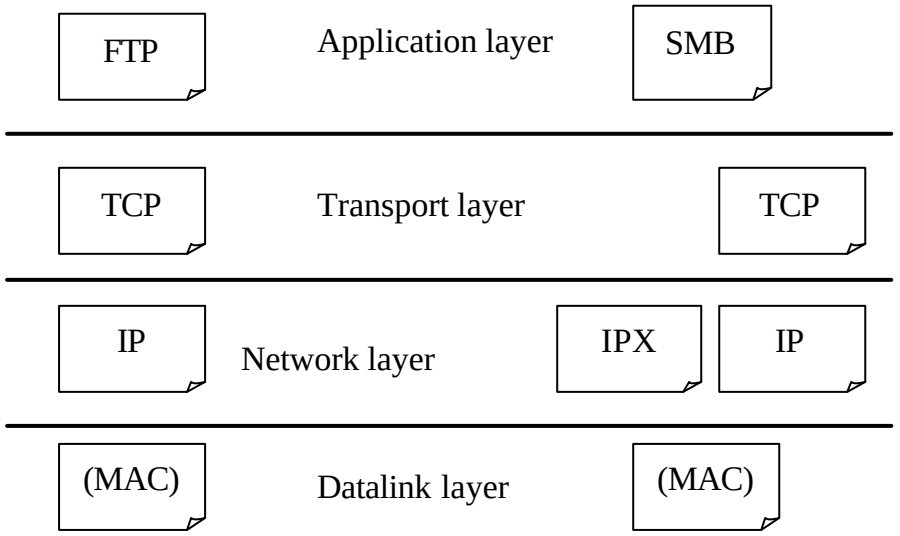
•NETBEUI – Self-configuring but **will not pass through routers.** Only useful for small networks.

•IPX – Self configuring, routable, compatible with Novell

•TCPIP – Requires more administration but is compatible with the Internet

Windows will try to use each of the protocols that have been configured and this can create a lot of (unwanted) network traffic.

# Compare with TCPIP - for file access



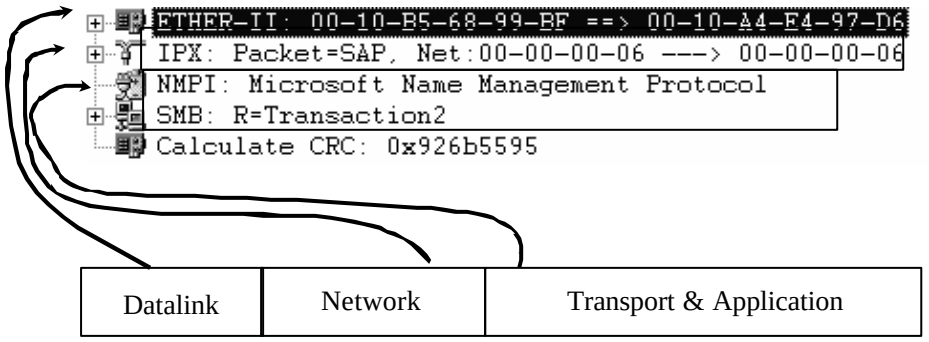



Here you can see the two **Protocol Stacks** compared - in regards to file transfer at least.

We could draw different versions of this chart for other jobs that needed doing on the network.

You might well ask what happened to the Transport Layer in the IPX case? Well, since IOX is **not a reliable protocol** (in the formal sense) the reliability needs to handled by SMB.

# NetXRay's point of view

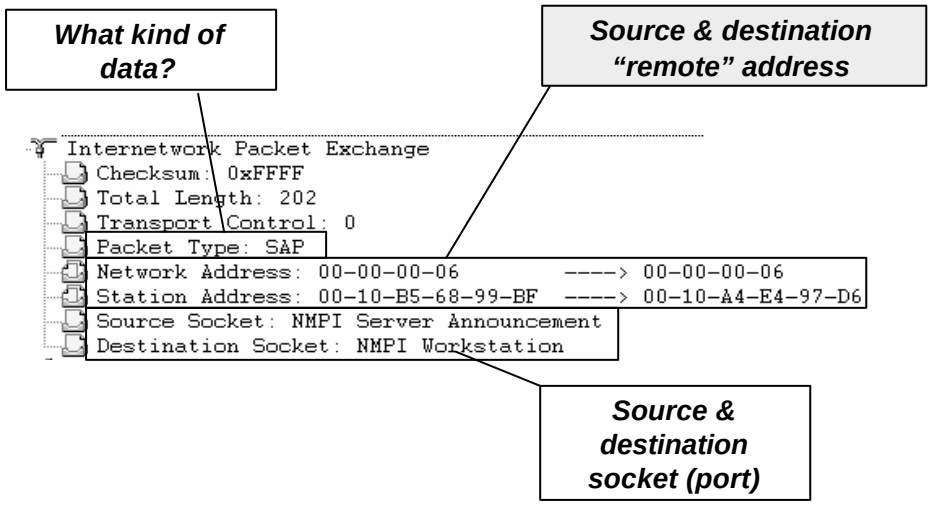





Here we see the various layers from within NetXRay.

The one confusing thing is the extra layer – the Microsoft Name Management Protocol (NMPI). You can ignore this – there is no data associated with this protocol.

# IPX Header

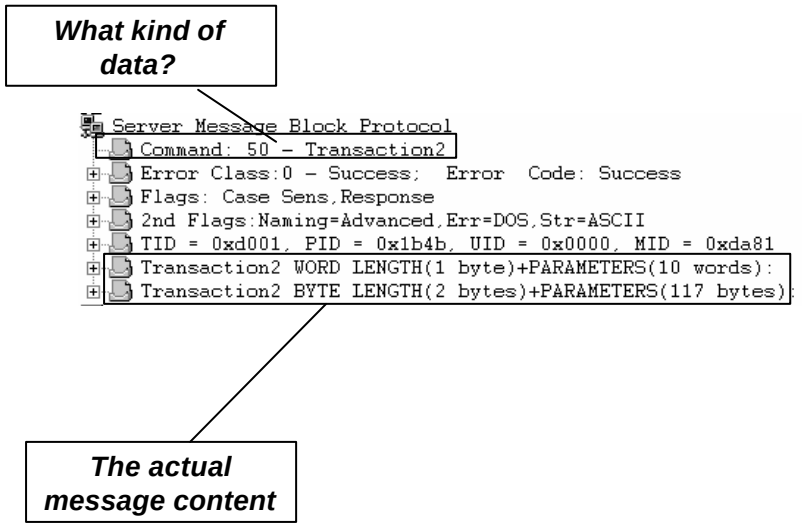





The IPX header contains fields that control the remote delivery of the packet.

You will notice that we see a **Socket (aka Port)** address in the IPX header which controls what destination the packet will have within the machine that it is addressed to.

# SMB Header

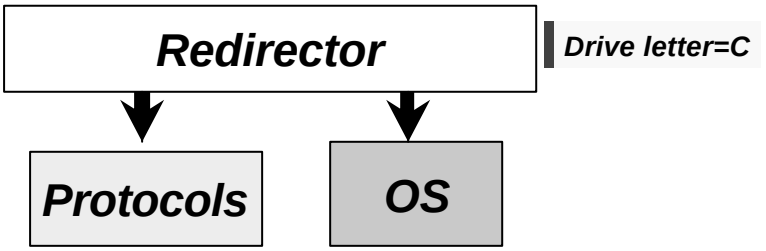





When we look at the SMB header we have to remember that Microsoft networking has developed piece by piece over many years (it actually began its life as a non-Microsoft product called LAN Manager)

The clearest sign of this is the use of "Transaction2" in the packet. An older pair of computers, with an earlier OS, would have used "Transaction" but the protocol has been added to in order to support newer OS versions.

# Redirector switching




- ***How does the redirector decide?***
  - C=local
  - K=remote ( eg \\DHS\DOCS )
  - The redirector needs a **MAP**



Returning to the redirector component at the workstation which is busy switching between local and remote file access. The decisions are based on the drive letters that are used in file names but the redirector requires a list of what letters are local and what are remote.

For remote drive letters the name of the server and volume also need to be stored.

This **MAP** is set up and queried by a command of the same name.

# Mapping Drives

- 3 methods
  - login scripts
  - DOS prompt
  - Windows



Drives need to be **mapped** automatically when you login - this is done by placing drive mapping commands in a **login script.**

You can also map drives from the DOS prompt and from Windows explorer.

# What can you Map to?

- *The Remote computer must have file sharing enabled*
  - *Default for NT or 2000*
  - *Enable in Network Control Panel for W95/98*
- *A directory must be made accessible*
  - *Default for NT or 2000*
  - *Right click and "Share" for W95/98*



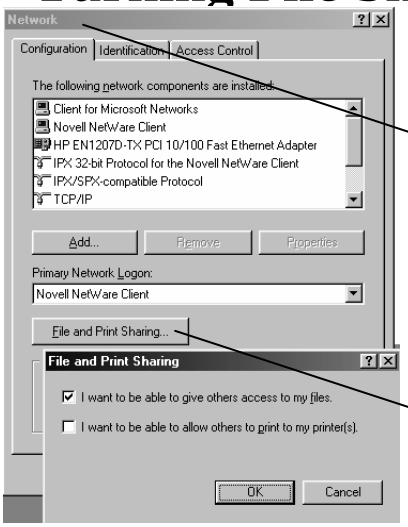
If we are to access files and directories on a remote computer the remote computer must have some say in the matter!

There are two levels to permitting access:

•Enabling file sharing
In other words the computer is prepared to talk SMB at all. The Microsoft server OS's (NT/2000) are in this state by default but the desktop OS's (Win 95/98) need file sharing to be turned on in the Network Control Panel.

•Some region(s) of the local file system must be made available.
The act of making a region available is called "sharing". Sharing for a directory needs to be turned on via the right-click menu for the directory.

# Turning File Sharing on




***Network Control Panel***

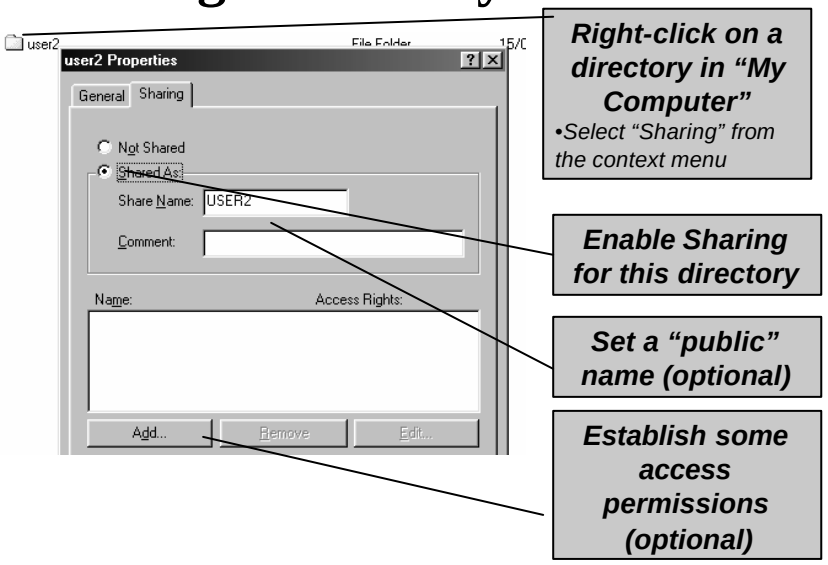
*•Left click on "Network Neighbourhood" and select "Properties"*

***or***

*•Start/Settings/Control Panel/Network*

***Display "File and Print Sharing" dialog***

# Sharing a directory



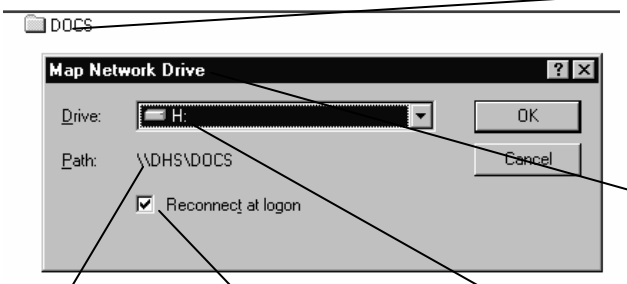



Given a directory that you want to share the right-click menu should now contain a "Sharing" entry. If it does not then you have missed the previous step!

When a directory is shared you can give it a name for public access (this will show up in the Network Neighborhood of other computers in the network)

Another question to be answered is "Who can access these files?" By default anyone can do anything to your files so you may wish to set up some access control (users and passwords)

# Mapping from Explorer




***Double-click on the computer to see what is being shared***
*( "DOCS" in this example )*

***Right-click on the folder to see the "Map Network Drive" dialog***

***Use "Network Neighbourhood" to find the remote computer***
*( "DHS" in this example )*

***Makes <u>your computer</u> remember this setting***

***Select a drive letter that you will "MAP" to the remote resource***



To map a drive you must start by browsing the Network Neighborhood icon on your desktop.

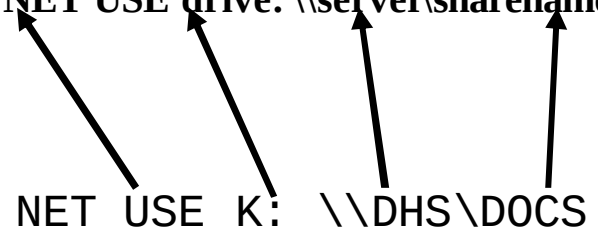
You need to locate the computer that you are trying to connect to – if you cannot find it you might try Start/Find/Computer – entering the name of the computer you are looking for. (Remember in the Remote Address Resolution topic that the Microsoft Browser service, and its shortcomings, were described)

When you see the computer icon double-click to see what it is sharing.

If you right click on one of the shared folders you should see a "Map Network Drive" option.

# Mapping at DOS prompt

- Syntax
  - **NET USE drive: \\server\sharename**



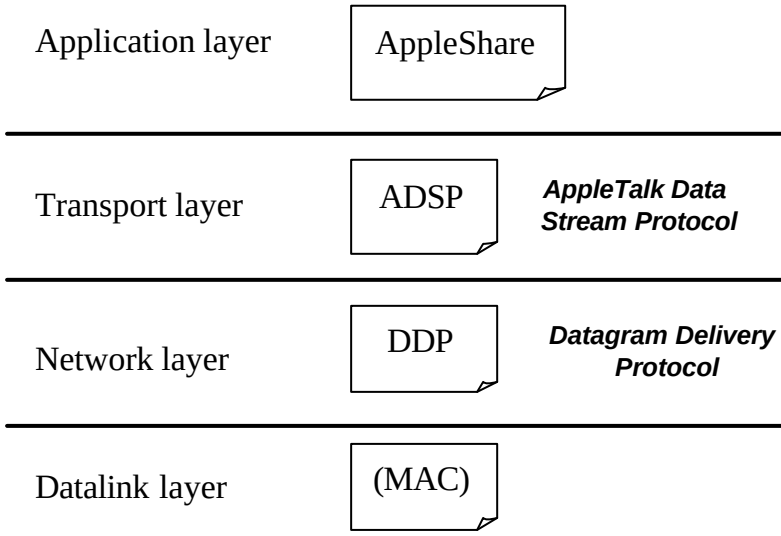



The syntax of the NET USE command is simply what is needed to describe remote locations and assign them to local drive letters.

The remote location needs a server and share name to be specified

The thing that most often gets forgotten is the colon after the drive letter.

# AppleTalk file sharing protocols

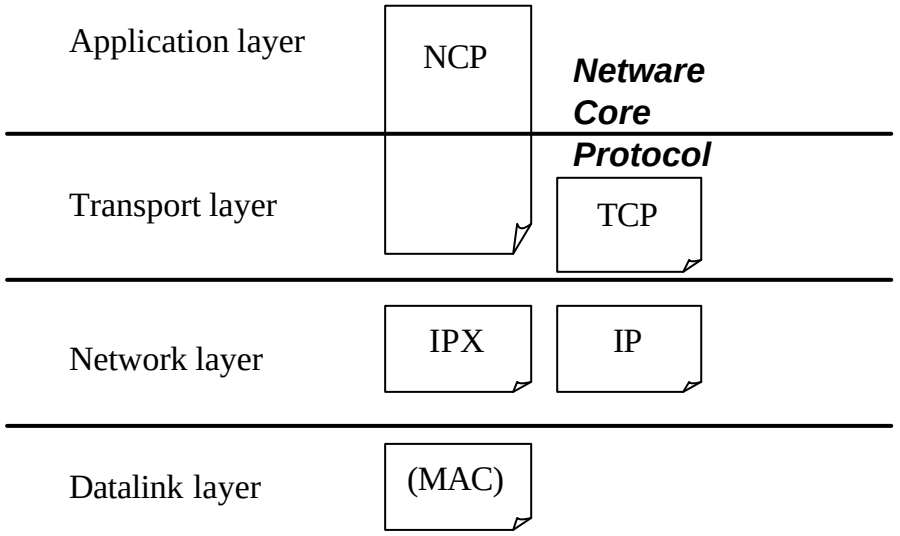





This is the way Apple does file sharing.

There is an Apple protocol to occupy each slot in the layered hierarchy. This is the first real look at the Apple protocol stack.

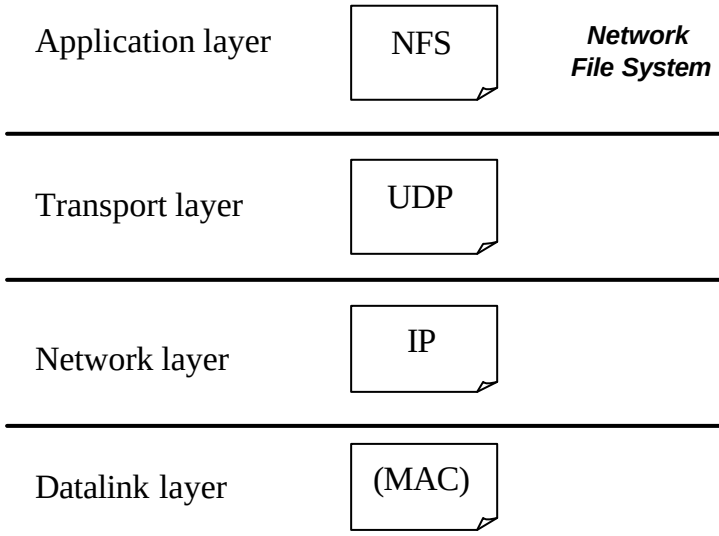
# Novell Networking






Novell uses **Netware Core Protocol** at the Application layer (so it does the same job as SMB) but in the lower layers it will use whatever transport is available.

# Unix NFS






The way that files are shared in Unix (as opposed to copied with FTP) is the **Network File System** protocol. We will study this in more depth next week.