In Figures 5.3 to 5.6, the results for up to 10 PCs are shown to suggest the performance of the VM system during extreme loading on the VM database server. The maximum expected N in normal conditions is, however, only two. The ethernet becomes more congested as N increases.

The time-out values used in the tests were chosen for the following reasons: To = 0.1 s is so small that a user would not notice the delay due to three failures (approximately 0.9 s with two unsuccessful retries to a failure) while To = 1.5 s is so large that even if only one retry occurred, the user would notice the delay. To = 0.2 s is a value in between the two extremes (a user would probably not notice one failure here).

The queue sizes used in the test were chosen for the following reasons: q = 1 is the minimum queue size possible (this is also the normal N) while q = 25 is a queue size that is always larger than N. The maximum N in normal usage is expected to be two and thus q = 2 is also included in the test.

From Figures 5.3 and 5.5:

If $q < N$, a better performance (i.e. a smaller T and R) is achieved with a larger queue size. The VM database server stops 'listening' for requests when the queue is full, and this increases R and thus also T.

Performance decreases as N increases. T increases because of the increased loading on the database server and because R increases. R increases because of the increased congestion on the network (and because of full server queues if $q < N$).

The performance of the case with two VM database servers (q = 1) is slightly worse than that of the single database server case, with q = 1 at half the N (within a Standard Deviation of 20% of the average). As N increases, each database server (in the TWO DB case) will handle approximately half the load, so performance is expected to be similar to that of the single database server with the same load. (The performance is slightly worse because the network congestion in the TWO DB case is higher.)

Having two VM database servers is better than having a single database server when the sum of the queue sizes of the two servers equals the queue size of the single database server and the VM database servers are under high load and operating in periods of high network congestion. In both cases the network congestion is intuitively expected to be similar, but each server in the TWO DB case only handles half the load, so a transmitting PC will have a smaller R and therefore a smaller T.

The best performance is obtained simply by ensuring that $q > N$. If the normal maximum expected N was greater than two, there might have been some performance advantage in having two database servers each with $q > N$.

From Figure 5.4, the smaller the time-out value set the smaller the T. At small time-out values, requesting PCs are quicker to send a retry to the server resulting in a smaller T. When an 8 MHz PC is used as the VM database server, the worst T is obtained because the server takes a longer time to process requests from the transmitting PCs.

From Figure 5.6, R increases as the time-out value is reduced (Standard Deviation is approximately 20% of R shown for the case in which $N = 2$, 3, 4, 5, or 6 and 10% of R shown for $N \geq 7$). At small time-out values, requesting PCs are quicker to repeat requests to the server and hence add to the congestion on the network thereby increasing the possibility of collisions thus resulting in more retries. When an 8 MHz PC is used as the VM database server, network congestion is at its lowest and hence the smallest R is obtained whatever the value of N.

From the results the following general statements can be made:

The performance of the VM database server is best when $q \geq N$. The time-out value should be small enough to prevent the user from seeing a delay in receiving the menu and large enough to avoid failing to receive information from the VM database server. A good time-out value would be about 0.2 s. With this value, even if one server fails the user will be reconnected to another one (if there is another one) within 0.8 s.

5.3.3 VM Database Server is on Another LAN

The purpose of this test is to see what effect placing the VM database server on another LAN has on the response time as seen from the user's PC.

The maximum queue size on the VM database server was set to 25 and the time-out value on the TEST800 program was set to 1.5 s. The TEST800 program was run on a 16 MHz Olivetti PC. The VM database server program was run separately on:

- the bridge; a 16 MHz Olivetti.
- 20 MHz Olivetti on LAN B which was not operating as a file server.
- 20 MHz Olivetti on LAN B which was operating as a non-dedicated file server.

The results are presented in Table 5.4 below.

VM Database Server	**Response Time for 800 Successful Requests [s]**
Non-Dedicated Bridge	5.87
20 MHz PC	6.10
Non-Dedicated File Server	7.36

Table 5.4: Response Time When the VM Database Server is on Another LAN

The computer used as the non-dedicated bridge should be similar in performance to the other 16 MHz computers on LAN A. Comparing the response time for 800 requests from a 16 MHz computer on LAN A when the VM database server runs on the bridge (5.9 s) to when it runs on an ordinary 16 MHz PC on LAN A (2.6 s as shown in figure 5.2), the poorer performance of the server on the bridge can be attributed to it being run on a non-dedicated bridge. The response time is even poorer when the server runs on an ordinary 20 MHz PC on LAN B. The VM database server

on the bridge was at least on the same LAN as the requesting PC. This was not the case with the 20 MHz PC. When the 20 MHz PC operated as both a non-dedicated file server and as a VM database server, the poorest performance was recorded. This again is attributed to the file server being non-dedicated. Generally, response time seems to worsen the further away the VM database server is from the requesting PC in terms of hops. (The user will probably not notice that the menu is being received from a remote source as long as the response time is less than 0.8 s.)

5.4 Failure of a VM Database Server

The purpose of this test is to see what effect the failure of a VM database server has on the user.

One VM database server was set up on LAN A. As long as it was operational, all users could access applications through the VM menu program. When the server was terminated, the VM menu program could no longer locate a VM database server and hence the menu program could no longer be used. It was observed that the menu program took three times the time-out value specified within the VM menu program to determine that no VM database servers could be located.

Two VM database servers were set up on LAN A. One VM database server was then terminated. It was observed that if the failure occurred on a server that the VM menu program was not accessing, then absolutely no difference could be detected when the failure occurred. If the server that was terminated was the one that the VM menu program was communicating with, then after a period of three times the time-out value specified within the program elapsed, a menu from the other VM database server was received. This situation remained the same even if the failure occurred on a VM database server that was located on another LAN.

These observations were verified using the TEST program as well.

The user will transparently be connected to another VM database server if the preferred server fails, as long as there is more than one VM database server in the

network. The importance of the time-out being set to a suitable value is again emphasised by this test. The VM_MENU program keeps a list of up to four VM database servers. If all the servers fail, then the user will get a "no VM database server" error message after 12 times the time-out value (two retries for each server). If the time-out value is 0.2 s then this will be after 2.4 s in the worst case.

5.5 VM Database Update

The purpose of this test is to determine what effect a VM database update will have on a user who is communicating with the VM database server.

The VM database server and a PC running the TEST program (both 16 MHz PCs) were started. From another PC the LOADDB program that would instruct the VM database server to load a new copy of the database was run. It was observed that the TEST program sometimes registered a retry attempt at that point and sometimes proceeded without registering any retries at all. It was very difficult to observe what a VM menu program user would notice in terms of time delay, during the database update because of the speed at which the VM database was updated.

If a user was using the menu program during the VM database update, depending on the changes made to the database the following two conditions could occur:

1. If the user chooses a menu from his screen that does not appear to users who run the VM menu program after the VM database has been updated, the following can be observed:

    a. If the menuID of the menu he has selected exists in the MENU.DBF file but none of the other menus from the main menu onwards refers to it, the user will still be able to step through the menus from that point onwards without any problem.

    b. If the menuID of the menu he has selected exists in the MENU.DBF file but it now refers to the 'wrong' menu (probably because that entry was deleted from the MENU.DBF file), the wrong menu will be presented to him. Any

application he selects from the wrong menu will be started as normal without any other problem.

c. If the menuID of the menu he has selected does not exist in the MENU.DBF file at all, he will not be presented with a new menu on that selection.

2. If the user chooses an application from his screen to which changes in the VM database have been made, the following can be observed:

   a. If the record in the AP_ACCES.DBF file corresponding to the appID of the application he intends to select is reset, but no other changes are made to the other files, the user will not be able to select that application from the menu.

   b. If the record in the AP_ACCES.DBF file corresponding to the appID of the application he intends to select is deleted, so that the appID now points to another record, but no other changes are made to the other files, the user may or may not be able to start the application from the menu. This all depends on the access category of the application that has moved up to occupy the slot in the AP_ACCES.DBF file caused by the deletion of a record.

   c. If the record in the AP_BATCH.DBF file corresponding to the appID of the application he intends to select is deleted, the application cannot be started from the menu.

   d. If the reference to the appID the user intends to select from his current menu has been deleted from the MENU.DBF file, but it exists in the other files, then the application is started as would have been the normal pre-update case.

   e. If the reference to the appID the user intends to select from his current menu has been deleted from the MENU.DBF file and from the AP_BATCH.DBF file and has been reset in the AP_ACCES.DBF file, the application cannot be started or selected. This is the situation that is expected if no mistakes are made while editing the VM database files.

If the proper procedure for modification of the VM database given in the VM database administrator's manual in Appendix B1 has been followed, then the information served out by the VM database servers can be updated without any problem, even while users are accessing the server. If the proper procedure has not been followed and mistakes occur, then it is possible to diagnose the problem according to the response the user sees when using the VM_MENU program.

5.6 File Server Administrator's Reaction

The file server administrator of the School of Engineering network has found the VM to be a convenient and labour-saving method of making applications located in multiple file servers available to all the users. It is easy to increase the number of users in the system and give them access to all the applications they are entitled to, and it is just as easy to change the VM group that the user belongs to (this determines the user's access rights). Putting users into a VM group is the same as putting users into any other group defined on a Novell file server. The VM group is a convenient way of dividing the user population because users tend to fall into certain natural categories. Having a standard format for the application batch files also helps in ensuring that applications can be successfully started even if the user is attached to many file servers. The menu system and the batch files substantially reduce training time for the users because the users do not have to know the NetWare commands. Problems with users forgetting their passwords are restricted to one password and one file server. Considering all these benefits, the additional steps required to set up the file server and users defined there to be part of the VM is worthwhile. The association with other file server administrators and the VM database administrator, to determine the VM groups and set access rights is only a small increase in the labour requirements from a single file server environment.

The file server administrator was not very happy about:

1. *Accounting*. Individual accounting for the use of system resources on file servers other than the user's home file server is not possible.

2. *Work groups*. The situation might arise where users belonging to different groups and on different file servers need to work together on a project. As a result, all the members of this work group might need access rights to data and applications required for the project. The current VM does not enable a user on one file server to have access rights to an application on another file server that is different from the general access rights given to his group. All users belong to only one VM group. This restriction makes it impossible to create cross-category work groups.

   Consider the following case:

   A policy decision is passed that no one belonging to the category of user called STUDENT is allowed to access an application called UNZATEST. Consequently, all server administrators place this restriction on all members of the group STUDENT. The user VM_STUDENT belongs to the VM group STUDENT on all the file servers and is the name by which users of the category STUDENT created on one file server can log into another file server. Now, suppose a particular user of category STUDENT is a member of a work group that uses UNZATEST for a project. Also suppose that this application does not exist on that user's home server. There is currently no way of enabling this user to access UNZATEST without creating a user account for this user on the server that has this application. This method does not scale well and will not be transparent to the user because he will have to explicitly login to the other file server.

3. *Individual access rights*. At present, users who want to access an application on another file server have either got access, or no access to that application. More detailed access rights such as the ability to read but not to write cannot be issued at the VM level. By common agreement between the file server administrators, various VM groups can be given additional rights.

For example:

The group LECTURER and the group STUDENT might both be given access to a particular application and its data files at the VM level. The group LECTURER might however be given the ability to write to the data files as well while the STUDENT can not. The setting of these rights is done by the file server administrators and not the database administrator. A particular member of the group LECTURER can not be given this detailed access rights.

4. *Printing*. Printing is another important facility that is greatly used on a network. The current VM does not allow printers connected to other file servers to be made transparently available to users on a different home file server. If printing is restricted to using a print queue on the home file server, then for as long as the desired type of printer is connected to the correct print queue transparent printing is not a problem. If the printer however changes location then the user will face some difficulties e.g. the user might think that the print job will appear on a laser printer but instead it appears on a dot-matrix printer with garbled output.

   Currently, access to printers on other file servers can be done in this way:

   a. Define print queues on the local file server that will be serviced by printers located elsewhere. The advantage of this scheme is that it provides transparent printing facilities to the user. Another advantage is that accounting facilities for the use of printers can be enabled. The disadvantage of this scheme is that it is not scalable. Whenever a printer's location is changed or a new printer is added, all the users have to be notified. Every user's printer configuration file will also have to be changed as well.

   b. Use the VM login name to access other printers. The advantage of this scheme is that it is scalable. It is very easy to add many users to new print queues simply by assigning the VM login name to a print queue. One disadvantage of this scheme is that it is not transparent. The formatting for the printer will have

to be done by the user. Here again the user has to know what type of printer it is and a lot more details. If the printer's location is changed, then the user will be really lost. Another disadvantage is that it is not possible to have detailed accounting. The usage of a printer for all the users of a particular VM login name can be determined, but the usage by a particular user of that VM group can not be ascertained. This is only good for statistical purposes.

The file server administrator's reaction confirms that the VM reduces labour requirements in installation of users and time taken to train users to use the network. This meets the scalability requirements of the VM. The concerns raised by the file server administrator are discussed below.

Application access from multiple file servers for a large user population, operating under a decentralised management, creates problems that this research deems of sufficient importance to be addressed. Hence, the special focus of this research has been on application access and not accounting. In a PC LAN for a university-type environment, accounting for things such as processor time, disk space, use of applications and network connection time is not usually done. Accounting for the use of expensive peripherals however, is something that could indeed prove beneficial. Since a complete VM provides a user with transparent access to all the peripherals he is entitled to use, this is a valid area for future research into the NetWare based VM. In NetWare, accounting under the technique this research reports on is difficult. The difficulty arises because user accounts are created on file servers and are only valid for the file servers in which they have been created. Creating work groups and giving individual users detailed access rights is also made difficult by this limitation.

Printing falls within the area of "transparent access to all the peripherals" stated in the paragraph above. When considering printing problems, the following should also be noted: currently, if a user has access to more than one printer, it often occurs that a user sends a print job to the 'wrong' printer. This, of course, will be further aggravated when the user has access to many printers located in various places on a campus.

Having mentioned this, it is still good to provide users with access to many printers located in various places. One way of doing this is by developing a VM printing application consisting of the following two components:

1. A database of information on printers. This database will contain information about the type of printers, their location and commands to set up the printer as well as VM access rights to it.
2. A program to run on the user's PC. This will present the user with menus. Using this program the user can specify:
   - the location to which his printing should be sent.
   - the type of printer.
   - the formatting on the printer.

   After the user has made his selections, all printing will then onwards be directed to the specified printer. The program will attach the user to the file server which has the queue serviced by the printer of choice using the VM group login name of the user. The program will also set the capturing of the printer port so that printing can be transparently redirected to the appropriate printer.

This scheme bears some similarity to the second option mentioned by the file server administrator but has the added advantage of being transparent to the user. The disadvantage of detailed accounting mentioned by the file server administrator unfortunately remains.

5.7 VM Database Administrator's Reaction

The file server administrator for the School of Engineering network performed the role of the VM database server administrator as well. Hence, it is not possible to obtain a reaction to the difficulties involved in co-ordinating various file server administrators as to the assignment of access rights to the VM groups and such types of administrative duties.

However, on the creation and maintenance of the existing database, the administrator says that the job is very simple and there is really very little to do.

5.8 VM Application Developer's Reaction

This is another area for which it would be good to provide a reaction to. Unfortunately, only the author of this dissertation has any experience in developing applications for the VM. I would say that developing applications for the VM is only as easy (or as difficult) as developing applications for NetWare. The communication with the VM database server is made easy by way of the library of routines that have been provided. Only rarely would an application deal exclusively with the VM database server only. Knowledge of the NetWare interface is definitely required to develop applications for the VM.

Note:

The TEST, TEST20K and TEST800 programs used the VM library. Some of the programs developed for the operation of the VM also use the VM library.

5.9 The VM Database and Server

5.9.1 An Easier Way to Create the Database.

The VM database was created by a 'C' program and is the largest program in the set of programs to create the VM as can be seen from Appendix A1. This program served its function well. To extend the information in the database, however, a new program will have to be written that will provide a user interface and the necessary data entry screens. An alternative way of achieving this is to use a traditional Database Management System (DBMS) to create and modify the database, and then write a 'C' program to change the contents of the database into a desired format. In this way, the user interface and other facilities are provided by the DBMS. This will reduce development time in creating a new database.

5.9.2 Placing the VM Database Server on a Non-Dedicated PC.

If the VM database server is placed on one of the file servers (or bridge), an extra computer to act as a database server will not be required. Below are some considerations:

1. In NetWare, processes are not interrupted by other processes (i.e. not pre-emptive multitasking)[25].
2. The memory requirements of the database might not make it possible for it to be loaded into the file server's memory. This will slow down the performance of the database server if it has to access the disk.
3. If the database server is written as a Value Added Process (VAP) to run on a NetWare v 2.15 file server, then updating the database will be very difficult. To restart the database server, the entire file server will have to be brought down and restarted so that the VAP can be loaded.
4. If the database server is written as a NetWare Loadable Module (NLM) to run on a NetWare v 3.10 (or more recent) file server, a 32 bit 'C' compiler is required[26].
5. If the database server is either an NLM or a VAP and is resource intensive (such as during periods of heavy demand for data), the performance of the file server is affected seriously[25].
6. The performance of the database server will be affected if it is placed on a non-dedicated file server.

From the above considerations and section 5.1 we can see that the VM database server is a good candidate to be an NLM. It is not resource intensive because the maximum number of items in the queue is usually only one. The VM database is not very large (about 9 KB, from section 5.1) so it can easily be loaded into the file server's memory. Even if the VM database is not loaded into memory and the VM database server has to access information from the hard disk, the users will not be noticeably inconvenienced (if the file server has a fast hard-disk) because of the low demand for data.

Consideration number one and number four above will still have to be taken into account when creating the NLM.

5.10 The PC

VM_MENU is not a TSR, but there have not been any noted disadvantages with this. Some advantages are; VM_MENU does not occupy any memory space after the user has selected an application and it does not clash with any TSR programs that are resident.

The current VM has been developed exclusively for the IBM PC and the Novell environment. In a Novell network that also consists of computers other than IBM PC compatibles or clones, the following will have to be done if these computers are also to benefit from the VM:

- The VM database has to be altered to include information on operating systems and hardware required to start an application.
- A NetWare C Interface for the particular hardware and operating system used on the other computers will have to be obtained.
- The hardware and operating system dependent code for all the programs will have to be changed to work on the other type of computer.

5.11 Failure of a File Server or VM Database Server

Since all users are created on only one file server, if that file server fails then all access to VM facilities for all users that have been created on that file server is terminated. Since this is a very serious issue, protection schemes on every file server has to be employed that will either prevent this failure or else enable quick recovery from a failure.

5.12 Broadcast Nature of the Service Advertising Protocol

The VM database servers advertise their presence on the network every 60 seconds using the Service Advertising Protocol (SAP). To do this, they broadcast their network address, name and type to every station on the network. This scheme increases network traffic and makes it unsuitable[27] for use in WANs. Fortunately, this is not as big a problem in LANs (such as a campus wide network), because generally the bandwidth on a LAN is under-utilised. The SAP provides a convenient method for file servers to update their list of available services on the network. If a VM database server is going down, it uses SAP to cancel the advert of its services and thus enable file servers to delete it from their list of servers.

5.13 A Better VM

The VM system based on the NetWare operating system (versions 2.15 and 3.10) developed in this research has proved to be useful though it is incomplete in the sense that the software for transparent peripheral connections has not been written. Despite it not being complete, the developed VM is still being used in the School of Engineering network. To address some of the concerns raised by the file server administrator of the Engineering Network however, some of the limitations imposed by the current base network operating system will have to be overcome.

A better network operating system for the VM would have all the advantages of NetWare version 3.10 as well as enable users who would have access to all the file servers (individually, not as a group) to be created once only. This would allow accounting, work groups and detailed access rights for individuals. The VM groups would still be useful to work out general access to applications though this can be overruled by the individual access rights (which is a feature that will also permit work groups to be created).

One area that could be improved on is the selection of the preferred VM database server that the user's PC will communicate with. Currently, the first database server on

the list will become the preferred server. A better response time is obtained from a server that is on the same segment of the LAN as the user's PC. The VM_MENU program should discriminate on this basis. The same is true for the selection of the file server containing the application the user has selected. If the application is on more than two file servers the VM_MENU program should determine which one to connect to in the following order of priority:

- Is the application present on the home file server? This is the preferred server.
- Or else, is the application present on any other attached file server? Connect to the nearest (in terms of the number of hops).
- Else, connect to the nearest active file server.

CHAPTER SIX

CONCLUSION AND FUTURE WORK

6.1 Conclusion

The particular focus of this research has been on solving the problems of application access from multiple file servers in a Novell network that caters for a large user community. The application access technique discussed in this dissertation is a step towards creating a computer network system, called a Virtual Machine (VM), with the following characteristics: it is simple to implement, scalable, easy to manage, transparent to the user and suitable for a university-type environment.

The VM has some of the features of a distributed system, but is not as complex and is built around a very popular commercial network operating system called NetWare (from Novell, Inc.). The primary type of computer used in the VM is another popular product: the PC. The trend is that PCs are becoming both cheaper and more powerful. As a result, it is now possible to have a network consisting entirely of PCs. Such a system can take advantage of the benefits of a PC network as well as the benefits of NetWare. As the two products (PCs and NetWare) evolve, the usefulness of the VM is expected to increase.

The value of the application access technique presented in this dissertation becomes apparent when there are many users and many file servers. The benefits of this research to the user are:

PC, so that the number of components in the VM would be the same as the number of components in a normal Novell network.

Four main problems with the developed VM have been noted. These are: accounting, work groups, detailed access rights for individual users and printing. To solve the first three of these problems, a network operating system with the benefits of NetWare version 3.10, and a facility to enable each user to be created only once and yet have access to all file servers using his own user name, is required.

The aspect of the VM that involves presenting a user with access to all the peripherals he is authorised to use, is a feature that this research did not attempt to solve in a scalable manner. At present, only the normal methods employed in Novell networks to allow users access to peripherals is used. However, the missing components can easily be added in future.

All the objectives of the research have been realised.

6.2 Future Work

Some areas for future work are presented below.

6.2.1 Heterogeneous Networks

Novell permits Macintosh computers and Macintosh networks to connect to a NetWare file server. The NetWare file server is presented as an Apple file server to the Macintosh. However, the Macintosh computers will not benefit from the VM because the current software for the VM environment has been designed exclusively for the IBM PC and uses the IPX communication protocol. The same holds true for networks that use different communication protocols (e.g. TCP/IP[28,29]), have different operating systems (e.g. UNIX), and have computers that are not compatible with the IBM PC. To broaden the VM to incorporate other systems, the following could be done:

- VM Station Software: Different station software that will present the application menus for the different types of computers should be developed. Furthermore, the software should be rewritten so that it does not specifically use the IPX protocol.

There are tools available that allow a Novell file server to access other environments. These tools are based on the STREAMS[15] interface (from AT&T) which is a method of providing a standard interface to protocol stacks on a server. The Transport Layer Interface[15] (TLI) supplements the STREAMS tools to allow different combinations of protocols to be stacked together. The TLI is a part of UNIX system V (from AT&T) and therefore enjoys industry-wide support. Software that is TLI compliant will provide service to any program that uses the TLI interface. By using the TLI library calls, a programmer could write an application that does not directly use the NetWare protocols. In this way, at least the networking portion of the application developed would be very portable. At runtime, the application could use the OSI, TCP/IP or NetWare protocol stacks transparently. The underlying transport mechanism can be changed without rewriting the application.

- VM Database Server Software: The VM database server software should be rewritten so that it does not directly use the IPX protocol. The use of Structured Query Language (SQL) should also be investigated to create a more flexible system.
- VM Database: The existing VM database on applications would have to be changed. Additional information such as operating system and hardware requirements to start an application would have to be included in it.

6.2.2 UNIXWARE

An interesting new product offered by Novell is UNIXWARE — a version of UNIX for INTEL-based machines that includes NetWare services[30]. UNIXWARE replaces the core NetWare operating system with UNIX. It comes in two editions; Personal Edition and Application Server Edition[31]. The Application Server Edition enables a PC to share files and can also run programs remotely. MS-DOS users, on a PC running a NetWare shell, can also access the application server to run non graphical UNIX applications through a Novell Virtual Terminal session. The Personal Edition users can

use the TCP/IP protocol instead of the IPX protocol and can transparently access files from file servers using the Network File System (NFS). Applications launched on the application server will present their interfaces on the user's PC through the X Window System.

Considering these features of UNIXWARE, it would be good to investigate how UNIXWARE can be incorporated into the VM.

6.2.3 Operation

The VM database server should be converted to an NLM to operate from a file server. The VM_MENU program should also determine the number of hops to file servers and VM database servers and thus establish the best server to communicate with.

6.2.4 Services

The following services can be added to the VM to make it even more useful.

1. *Printing*. A scalable and transparent print service has to be developed for the VM. Currently, the file server administrator on the user's home file server sets the user's access rights to particular print queues found on the home file server. These print queues are usually serviced by printers attached to the user's home file server. The user has limited control over his printer configurations.

2. *Directory service about users*. A VM directory service with information about users containing login and logout times can be created. This service should consist of two software parts; one that will create a database of user names and user's home file servers from the binderies of all the file servers on the network, and the other to obtain a complete user name from supplied clues and then acquire more information about that specific user from that user's home file server.

3. *Creation of job servers*. It is possible to have a degree of distributed computing by creating a few job servers in the VM. An example of a good choice for a job server is a compiler. Source files that have to be compiled can be sent to a job

server that will perform the actual compiling and return the object files to the requesting station. The compile job server would normally be a more powerful computer than the station requesting service.

4. *E-mail.* The VM is just a collection of application packages and is therefore still a traditional Novell Network. Most applications developed for NetWare should work without any modification in this network. This includes electronic mail packages. The mail packages that are available should be investigated to find the most suitable in the VM environment. The choice should not detract from the benefits of the VM especially in the area of transparency and scalability.

REFERENCES

1. Bennett, M., *Proposal on the Expansion of Computer Facilities*, University of Zambia, Computer Centre, Lusaka, Zambia: 1990. (Internal Report).

2. Hawe, B., Kirby, A. and Stewart, B., "Transparent Interconnection of Local Area Networks with Bridges", *Journal of Telecommunication Networks*, Vol. 3, No. 2, Summer 1984, pp 116-130.

3. Udell, J. and Mitchell, R., "Networks of Peers", *BYTE*, Vol. 15, No. 6, June 1990, pp142-162.

4. Rash, W. and Stephenson, P., *The Novell Connection*, New York, NY: Brady, 1990.

5. Sloman, M. and Kramer, J., *Distributed Systems and Computer Networks*, Englewood Cliffs, NJ: Prentice-Hall, 1987.

6. Champine, G.A. et al., "Project Athena as a Distributed Computer System", *IEEE Computer*, Vol. 23, No. 9, September 1990, pp40-50.

7. Tanenbaum, A.S., *Computer Networks*, Englewood Cliffs, NJ: Prentice-Hall, 1989.

8. Coulouris, G.F. and Dollimore, J., *Distributed Systems: Concepts and Design*, Wokingham, England: Addison-Wesley, 1989.

9. Halsall, F., *Introduction to Data Communications and Computer Networks*, Wokingham, England: Addison-Wesley, 1985.

10. Tanenbaum, A.S. et al., "Amoeba: A Distributed Operating System for the 1990s", *IEEE Computer*, Vol. 23, No. 5, May 1990, pp44-52.

11. Satyanarayanan, M., "Scalable, Secure, and Highly Available Distributed File Access", *IEEE Computer*, Vol. 23, No. 5, May 1990, pp9-20.

12. Black, D.L., "Scheduling Support for Concurrency and Parallelism in the Mach Operating System", *IEEE Computer*, Vol. 23, No. 5, May 1990, pp35-43.

13. Novell, Incorporated, *NetWare C Interface —DOS*, Volumes I and II, Austin, TX: Novell, Inc., 1990.

14. Walker, B.J. and Popek, G.J., "A Transparent Environment", *BYTE*, Vol. 14, No. 7, July 1989, pp225-233.

15. Malamud, C., *Analyzing Novell Networks*, New York, NY: Van Nostrand Reinhold, 1990.

16. Microsoft Corp., *Microsoft MS-DOS version 5.0: User's Guide and Reference*, Redmond, Washington: Microsoft Corp., 1991.

17. Novell, Incorporated, *Supervisor Reference Set for SFT/Advanced NetWare*, Provo, Utah: Novell, Inc., 1990.

18. Novell, Incorporated, *Maintenance/Upgrade Set for SFT/Advanced NetWare*, Provo, Utah: Novell, Inc., 1990.

19. Novell, Incorporated, *Installation Set for SFT/Advanced NetWare*, Provo, Utah: Novell, Inc., 1990.

20. Novell, Incorporated, *User's Manual for SFT/Advanced NetWare*, Provo, Utah: Novell, Inc., 1990.

21. Novell, Incorporated, *Supplements*, Provo, Utah: Novell, Inc., 1990.

22. Novell, Incorporated, *NetWare System Interface Technical Overview*, Austin, TX: Novell, Inc., 1990.

23. Hogan, T., *The Programmer's PC Sourcebook*, Redmond, Washington: Microsoft Press, 1988.

24. Stallings, W., *Data and Computer Communications*, New York, NY: Macmillan, 1988.

25. Rose, C.G., *Programmers Guide to NetWare*, New York, NY: McGraw-Hill, 1990.

26. Davis, R., *NetWare 386 Programmer's Guide*, Reading, Massachusetts: Addison-Wesley, 1991.

27. Miller K., "LAN Managers: Take a Walk on the Wide Side", *Data Communications International*, Vol. 22, No. 4, March 1993, pp 23-24.

28. Cerf, V.G. and Kahn, R.E., "A Protocol for Packet Network Intercommunication", *IEEE Transactions on Communications*, Vol. COM-22, No. 5, May 1974, pp637-648.

29. Leiner B.M., Cole, R., Postel, J. and Mills, D., "The DARPA Internet Protocol Suite", *IEEE Communications Magazine*, Vol. 23, No. 3, March 1985, pp29-34.

30. Hurwitz, J.S., "January Surprise: Novell Heats Up Client-Server Competition", *Data Communications International*, Vol. 22, No. 5, March 1993, pp 25-26.

31. Yager, T., "UnixWare: New Hope For Unix", *BYTE*, Vol. 18, No. 1, January 1993, pp 51-52.

GLOSSARY

80x86 Family of microprocessors made by INTEL and used in the PC line. Examples are the 80286, 80386 and 80486 chips. The PC/AT uses the 80286 chip.

Accounting A service within Novell NetWare that permits the system administrator to determine usage of a particular resource on the network by the user. Normally this is used to charge for network expenses, but it may also be used for security.

Accounts The information that NetWare maintains about a user. This includes the user's name, password, rights and trusteeships.

Address An object is located there.

Amoeba A distributed system developed at the Free University in Amsterdam.

API See Application Programming Interface

AppID See Application ID

Apple Manufacturer of the Macintosh computer.

Application	A program that performs functions for a user. Spreadsheet and word processing programs are examples of applications.
Application ID	The record number starting from one in the AP_ACCES.DBF file in the VM Database. This number uniquely identifies the application.
Application Programming Interface	A routine in a computer program that is designed to allow information transfer with another program. Generally, these are a set of services that are available to a programmer. NetWare APIs are the programmers interface to NetWare.
Athena	A network system bearing some similarity to the VM. Athena was developed at the Massachusetts Institute of Technology.
Authentication	Determining the identity of the user
Authorisation	Determining that the user has legitimate access to the requested resource
Backbone	A network used to connect other networks.
Bandwidth	The total capacity for information transfer in a communications medium.

Bindery	The files in the NetWare System directory that contains information about users, passwords, connection information, etc. The NetWare operating system uses the bindery for management purposes, but this is also available to the users of the network to store information.
Bridge	Bridges connect two LANs, or two different portions of the same LAN. It is a term used by Novell to denote a computer that accepts packets at the network layer of the OSI model and forwards them to another network. A bridge will isolate traffic to the portion of the LAN for which it is intended, and will only forward those packets to the other connecting LAN if it really has to be forwarded.
Broadcast	Send information to all users on a LAN.
C	A programming language.
Centralised Database	The database is just in one location.
Client	A user (or a program acting for a user)
Collision	Events on an ethernet in which two stations attempt to transmit information at the same time.
Command.com	The MS-DOS command interpreter program. This is a TSR.
Computer Network	An interconnected collection of autonomous computers.

Connection	An attachment to a file server. The file server identifies the PC making requests for data by means of a unique connection number that it assigns for the PC.
Database	An organised collection of information.
Distributed Database	Appears to the user as a single database, but is in fact a collection of several different databases.
Distributed Processing	The processing load is split between co-operating computers.
Distributed System	A very transparent computer network that includes distributed processing.
Driver	A program that acts as an interface between an application or operating system and an external device.
Electronic Mail	A collection of programs that enables users to send messages to each other.
E-mail	See Electronic Mail
Ethernet	A widely used LAN protocol developed jointly by Xerox, INTEL and DEC and subsequently adopted by the IEEE as a standard.
Executable	A program that is ready to run on an operating system.

File Server Administrator	A person who creates and manages user accounts and sets access rights and trustee rights on a File server.
File System	The part of the operating system that is responsible for storing and retrieving data from a disk.
Gateway	A computer connected to two different network architectures that enables users to be provided with services in the remote environment.
Heterogeneous	Different.
Heterogeneous Network	A network consisting of different network protocols or kinds of computers.
Hop	A single data link. A mechanism that will move data for the network layer of the OSI model.
IBM	International Business Machines. A computer manufacturer.
INTEL	Manufacturer of the most widely used microprocessor chips (80x86) in the microcomputer industry.
Interface	The connection area between dissimilar systems.
Interrupt	A process that stops the operation of the microprocessor and causes it to take some action.
Invoke	To begin operation.

Internetwork Packet Exchange	The Network Layer protocol in Novell NetWare.
IPX	See Internetwork Packet Exchange.
LAN	See Local Area Network.
LAN Manager	A network product from Microsoft that competes with Novell NetWare.
Loading	The demand for services.
Local Area Network	A computer network the covers a small geographical area.
Login	The process of authentication that is required to attach a user to a server. Only after a succesful login will the user be allowed to use facilities or services provided by the server.
Login Name	The name by which the user (or any object) is known to the server.
Login Script	A file that contains preset commands to be run whenever a user logs into a NetWare file server.
Logout	The operation to remove a user's connection to a file server. This operation is normally performed when the user ends a session.
MAP	The process of assigning areas on the file server as disk drive equivalents.
MCB	See Memory Control Block

Memory Control Block	This is used to control a block of memory, is 16 bytes long, and is located immediately before the controlled memory area.
MS-DOS	Microsoft Disk Operating System.
Name	An object is referenced by this.
NetWare	A LAN operating system and the networking components sold by Novell. The NetWare operating system, PC software, transport protocol stack and a collection of data link drivers is commonly called NetWare.
NetWare Loadable Module	Program in NetWare version 3.1x that when loaded becomes a part of the operating system.
Network Address	The number of the network that the user is on. Each sub-network in a network has a number assigned to it. The full address of a station is the network address plus the local address of the node on the network.
NLM	See NetWare Loadable Module.
Node	An individual item in a set. An ethernet node is a device attached to the cable with a transceiver, including a repeater, bridge or computer. A file system node is a directory or an individual file.
Novell	The manufacturer of NetWare.

OSI	*Open Systems Interconnect.* A standard for LAN organisation developed by the ISO that allow many different vendor's equipment to communicate. The standard consists of seven theoretical layers, each with individual functions to perform within a LAN.
Parallel Processing	A feature of systems containing many processors that enables a task to be split among a subset of the processors for simultaneous processing.
Password	A group of characters used to prove identity.
PATH	An MS-DOS environmental variable that indicated directories to be search to locate an executable file.
Peer-to-Peer	A LAN design where all nodes communicate to each other on the same level with respect to the OSI model.
Peer-to-Peer LAN	A LAN model where all stations can act as both a server and workstation.
Program Segment Prefix	This is a 256 byte long reserved area that is set up by MS-DOS at the base of a memory block allocated to a transient program. The PSP contains some linkages to MS-DOS that can be used by the transient program, some information that MS-DOS saves for its own purposes and some information that MS-DOS passes to the transient program.
Protocol	A common pattern for communication between computers.

PSP	See Program Segment Prefix
Queue	A list of requests, processes or jobs waiting for a server or a peripheral to handle them.
RAM	See Random Access Memory
Random Access Memory	Dynamic memory which to which information can be written to or read from.
Read-only	A device or storage medium that can only be read from and not written to.
Record	A line of data in a database file.
SAC	See Server Attachment Count
Scalability	The ability of a system to cope with the addition of stations and users with a minimum of management and disruption.
School of Engineering	The only school in UNZA that had a Novell LAN during the development of the VM. This school provided the facilities for the research and the developed VM was tested and implemented here.
Server	A program on a computer that provides services or resources to workstations. File, database, print and communications servers are just a few examples.
Server Attachment Count	This is a count of the number of applications that require a connection to a particular server. The information is stored in a file called a SAC file.

Serverstation	Same as peer-to-peer LAN.
Service	A set of actions to be performed
SAP	See Service Advertisement Protocol.
Service Advertisement Protocol	The current network address of services are made known by using this NetWare protocol.
SQL	See Structured Query Language.
Station Mobility	The feature of a network whereby stations can be moved around on a network with the minimum of management.
STREAMS	An AT&T mechanism developed for the UNIX operating system and developed by Novell for NetWare. STREAMS is a way of connecting a series of software modules, letting them send messages to each other.
Structured Query Language	International standard language for communicating with databases.
Sub-network	A section of a LAN that is connected to the rest of the LAN by means of a bridge.
TCP/IP	Transmission Control Protocol / Internet Protocol. A non-proprietary network protocol used for interconnecting dissimilar network systems (such as NetWare to UNIX) by interconnecting at the transport layer of the OSI model.

System Manager	The one who manages a network system. In the VM, this could be a file server administrator or a VM database administrator.
Terminate and Stay Resident	A program that is loaded into the computer's memory and is invoked without having to be loaded again.
TLI	See Transport Layer Interface.
Transparent	The concealment of separation from the user and the application programmer, so that the system is perceived as a whole rather than as a collection of independent components.
Transport Layer Interface	AT&T developed specification for the interface between the transport layer and the upper-layer users.
Trustee	NetWare security concept. A user or group that has limited control over a file or a group of files on the file server. Trustees usually have different rights from the general public over the files that they are trustees.
TSR	See Terminate and Stay Resident.
University of Zambia	A university located in Lusaka, Zambia. The creation of a LAN for this university was the driving force behind this research.

UNIX	An operating system developed by AT&T which is now available for nearly every computer sold.
UNZA	See University of Zambia.
User	Person who uses a PC, program or services
User Mobility	The feature of a network whereby any user can use any station.
VAP	See Value Added Process.
Value Added Process	A program that runs in a NetWare version 2.1x file server, and adds additional capabilities. Replaced by the NLM in NetWare version 3.1x.
Virtual Machine	A computer network system similar to a distributed system but less complicated and without any distributed processing ability.
VM	See Virtual Machine
VM Database	The information files that are necessary for the operation of the VM.
VM Database Administrator	The person who manages the VM database and the VM database servers.
VM Database Server	This is a server that serves information out of the VM database to requesting PCs.
VM Group	A sub-section of the user population that are given common access rights to applications. A user can only belong to one VM group.

VM Menu Program	The program that presents a menu of applications to users. This program runs on the user's PC and requests information from the VM database server.
WAN	Wide Area Network.
Work Group	A term for people who work together on a common project.
Workstation/server	Classification of computers on the network as being in two mutually exclusive categories known as servers and workstations.

APPENDIX A1

INFORMATION ON THE VIRTUAL MACHINE

A1.1 Introduction

This appendix contains the following information on the Virtual Machine (VM): hardware specifications, maximum limits due to software, size of the VM programs, structure of the VM database, format of the request and response buffers, and services provided by the VM database server.

A1.1.1 The Virtual Machine

The Virtual Machine (VM) is a computer network system built around Novell NetWare that enables users to transparently access applications located on different file servers and allows for a scalable and easy to manage system. The users are provided access to the applications in the following way:

- Every user in the network is divided into one of several VM groups according to access rights to applications.
- Each VM group has a VM group login name to other file servers.
- PC software running on the users PC will present a menu of applications to the user and depending on his choice will use the VM group login name (of the VM group in which he belongs) to login to the file server where the chosen application is located.

- The PC software receives all required information from a VM database server.

A1.1.2 Categories of Administrators

The VM has the following categories of administrators:

- *VM database administrator* who maintain the VM database and oversee the operation of the VM.
- *File server administrators* who maintain user accounts on the file servers they are in charge of and enable the users to have access to the VM programs.

A1.1.3 Benefits of the VM

The benefits of the VM to the user are:

- The user only has to deliberately login to one file server to be provided network wide services. Most of the users will be unaware of the file servers and the VM Database servers.
- The user can select the application of choice through a menu system. He does not have to know where the applications are.
- The users who already know how to use a PC undergo only a little further training on the use of the VM.

The benefits of the VM to the file server administrators are:

- User accounts do not have to be created on every file server on the network for each user.
- A degree of autonomy is provided to the file server administrators on each sub-network to create and delete users and to restrict access to applications and other services on their sub-network as they see fit.
- New applications and updated versions can be provided to all the users easily. Through the menu system, the users are shielded from most of the changes because the environment for each application is set for them. Much of the labour

requirements in making users aware of the applications (and how to start them) are thus removed from the file server administrators.

A1.2 Hardware Specifications

A1.2.1 VM Database Server PC

- No Hard disk is necessary
- A 10 MHz, INTEL 80286 based motherboard will suffice.
- 4 MB of RAM should take care of even future requirements
- No floppy drive is required.
- A network card for network communications
- No serial or parallel ports are needed for the operation of the database
- During operation, no keyboard or screen is necessary as well

If the chosen PC does not have a floppy disk drive or a hard disk, it should be configured to boot from a file server.

A1.2.2 User's PC

Even an IBM XT class PC can run the VM software, as long as all pertinent hardware required to be part of a Novell network is installed in it.

A1.3 Maximum Limits Due to Software

The maximum sizes by category is presented below in Table A1.1.

	CATEGORY	MAX. SIZE
1	Applications	1000
2	Menus	100
3	Items in a Menu	20
4	VM Groups	9

Table A1.1: Maximum Size by Category

A1.4 Size of the VM Programs

The size of the VM Programs is shown in Table A1.2.

	NAME	SOURCE (C extension) [BYTES]	EXECUTABLE (EXE extension) [BYTES]
1	CDB	36510	244388
2	BATCH	5832	61296
3	DB	20367	60350
4	LOADDB	8981	36323
5	VM_CREAT	16996	178601
6	MP	6331	34068
7	VMSET	5691	24788
8	REGVM	2960	21663
9	GETPARAM	1607	20141
10	VM_MENU	24134	215665
11	LG	6221	28363
12	LGO	2455	20980

Table A1.2: Size of the VM Programs

A1.5 Structure of the VM Database

The VM database files at present are:

- MENU.DBF
- AP_ACCES.DBF
- AP_BATCH.DBF
- AP_BATCH.NDX
- USER.DBF: This file is not used at present. It is only provided for future use in the creation of a network wide user directory.
- USER.NDX: This file is an index to the USER.DBF file. It is not used at present.
- PASSWORD.DBF

A description of the record structure in these files follows. The 'C' programming language notation is used.

A1.5.1 MENU.DBF

This file contains all the information on the menus that are to be presented to the requesting PC. Before describing the structure of one record in the file MENU.DBF, the description of one menu choice in the menu is given.

```
struct g_menuChoice {
        BYTE type;
        char name[20];
        WORD ID;
        };
```

- g_menuChoice.type indicates whether this choice is another menu or whether it is an application.
- g_menuChoice.name is the name of this choice.
- g_menuChoice.ID is the unique identification number of this menu or application.

```
struct menu {
        char name[20];
```

```
struct menuChoice choice[MENU_ITEMS];
}g_menu;
```

Here g_menu.name is the title of the menu. The next item in the menu structure is an array of structures representing all the possible choices that can be displayed in that menu. MENU_ITEMS is the maximum number of choices possible in the menu (where menuChoice[0] is the first item on the menu, menuChoice [1] is the next item and so on). The record number represents the menuID and this starts from zero. MenuID zero is always the first menu that is to be displayed to the user. All other menus depend on the choice made by the user.

A1.5.2 AP_ACCES.DBF

This file contains all the information as to what category of user can access a particular application and the type of operating system and hardware needed to run the application. Only the access category is being used at the moment.

```
struct apAccess {
    char appName[20];
    BYTE access;
    BYTE OS;
    WORD hardware;
    } g_apAccess;
```

The record number represents the application ID starting from one.

- g_apAccess.appName is the name of the application.
- g_apAccess.access contains information as to the categories of users that can access this application.
- g_apAccess.OS contains information as to the types of operating systems that can start this application. This is not used at present.
- g_apAccess.hardware indicates the hardware requirements of the machine that will start the application.

A1.5.3 AP_BATCH.DBF

This file contains information about which servers have which applications and the batch file that has to be run to start the application.

```
struct apBatch {
        char server[SERVER_NAME_LENGTH];
        WORD appID;
        char batchFile[13];
        } g_apBatch;
```

- g_apBatch.server is the name of the server where the application may be found.
- g_apBatch.appID is the unique ID number of the application.
- g_apBatch.batchFile is the name of the batch file that will start the application.

A1.5.4 AP_BATCH.NDX

This file is created by a program run by the database administrator (BATCH.EXE) that sorts the AP_BATCH.DBF file. The AP_BATCH.DBF is sorted firstly according to appID (in ascending order) and then according to server name (also in ascending order). AP_BATCH.NDX is an index file that keeps an index of the position at which a particular application occurs in the sorted AP_BATCH.DBF file, and the number of consecutive records after this one that has information about that application.

```
struct apBatchNdx {
        int recPos;
        int numRecs;
        } g_apBatchNdx;
```

The record number represents the application ID starting from one.

- g_apBatchNdx.recPos shows the position of the first record in AP_BATCH.DBF that contains information on the application given by the appID specified by the record number in file AP_BATCH.NDX.

- g_apBatchNdx.numRecs is the number of consecutive records in AP_BATCH.NDX that has this information.

A1.5.5 PASSWORD.DBF

```
struct pass{
        char group[VM_GROUP_NAME_LENGTH];
        char user[VM_USER_NAME_LENGTH];
        char password[VM_PASSWORD_LENGTH];
        } pass;
```

The position of each record in the password.dbf file indicates the access category. The category is determined from the formula 2^x (where x is the record number of the PASSWORD.DBF file, and $0 \leq x <$ total number of access groups). The last record in the file is always the access category of the superuser. This is the category of user to whom no restrictions as far as access to applications and files is placed.

- pass.group is the access category.
- pass.user is the name by which this category of user logs into foreign server.
- pass.password is the password used to log into another server

A1.6 Format of the Request and Response Buffers

All communication by PCs with the database server is through two buffers. The first buffer contains PC requests to the database server. The second buffer contains the response from the database server.

A1.6.1 Request Buffer

```
struct {
        BYTE requestType;
        WORD requestNumber;
        BYTE retry;
        char data[REQUEST_SIZE];
        } request;
```

- request.requestType represents the nature of the request.
- request.requestNumber is the number of the request made to the database by the PC. Each time the PC requests a particular information, this number is incremented. If the PC has to retry a request, only the retry count is incremented. In this way it is possible for the PC to be able to determine whether a response to a particular request has already been received or not.
- request.retry contains the retry number if the PC has not been able to receive a response from the database server within a specified time-out value. This number is incremented until the maximum number of retries has been reached. At this time the PC will begin looking for another server to service it's requests. The database server does not however use this information at present.
- request.data contains the actual specifics of the request and this buffer has to be filled in a prescribed format for the particular request type.

A1.6.2 Response Buffer

```
struct {
        BYTE responseType;
        WORD requestNumber;
        char data[RESPONSE_SIZE];
        } response;
```

- response.responseType contains the error information from the server in servicing a particular request.
- response.requestNumber is simply a copy of request.requestNumber. This information is to allow the PC to determine which one of its requests to the database server is being responded to.
- response.data contains the actual specifics of the response and this buffer has to be filled in a prescribed format for the particular request type.

A1.7 Services Provided by the VM Database Server

A1.7.1 Get Password

This will return all the groups, user names and passwords that will fit in the response.data buffer starting from the requested record number in the PASSWORD.DBF file.

```
request.requestType = PASSWORD_RECORD;
```

request.data contains:

```
int recordNumber;
```

response.data contains:

```
struct pass{
      char group[VM_GROUP_NAME_LENGTH];
      char user[VM_USER_NAME_LENGTH];
      char password[VM_PASSWORD_LENGTH];
      } pass[numberOfRecords];
```

where

```
numberOfRecords = RESPONSE_SIZE / sizeof (struct pass);
```

A1.7.2 Get Menu

This will return the menu specified by the requested menu ID (equals record number in the MENU.DBF file starting from zero).

```
request.requestType = MENU_RECORD;
```

request.data contains:

```
WORD ID;
```

response.data contains:

```
struct menu {
      char name[20];
```

```
struct menuChoice choice[MENU_ITEMS];
}g_menu;
```

A1.7.3 Get Application

This will return information about all the servers that contain the application and the batch files that have to be run to start the application.

```
request.requestType = APPLICATION_RECORD;
```

request.data contains:

```
WORD ID;
```

response.data contains:

```
struct application {
    int numberOfServers;
    BYTE accessCategory;
    struct apBatch {
        char server[SERVER_NAME_LENGTH];
        WORD appID;
        char batchFile[13];
        } g_apBatch[numberOfServers];
    };
```

where

numberOfServers is the number of servers that contain the application.

accessCategory specifies the categories of users that can access this application.

A1.7.4 Load VM Database

This will load the VM database from the specified file server. The requesting PC will be notified whether this operation has been successful or not. Reasons for failure could be: none of the files were loaded, or only a partial loading was achieved. If no files

were loaded, the server resumes with the old database. If a partial loading was made, the server terminates.

```
request.requestType = LOAD_DATABASE;
```

request.data contains:

```
struct fileServerWithDatabaseFiles{
      char name[SERVER_NAME_LENGTH];
      char password[9];
      } fileServer;
```

where

name is the name of the file server where the VM database files to be loaded are located.

password is the password to login to that file server (the user name is VM_DATABASE).

response.data does not contain anything.

response.responseType has the following values:

0	Database has been successfully updated.
1	Update FAILED... Database Server TERMINATED!
2	Supplied the wrong file server name and/or password.
0xFF	Database server could not process the request

APPENDIX A2

SUPPLEMENT TO CHAPTER FIVE

This appendix contains information that was too bulky to be included in chapter five (Testing and Discussion).

A2.1 Components Used in the VM Test

The following components were used for the test:

A2.1.1 Cables

Thin Ethernet (10BASE2); LAN A had approximately 180 m of cable and LAN B had approximately 1 m of cable.

A2.1.2 PCs on LAN A

There were 30 PCs on LAN A including the file server and the bridge. All the PCs had the following installed:

MS-DOS 5.0

NET5.COM version 3.10

IPX.COM version 3.01 Rev. A

Other relevant details are given below.

- Eleven IBM PS/2 model 30s:

  INTEL 80286 microprocessor at 10 MHz

  1 MB RAM

3Com EtherLink II network card /MCA

- Seven Olivetti M290S (IBM PC/AT compatible):

  INTEL 80286 microprocessor at 16 MHz

  1 MB RAM

  3Com EtherLink II network card /ISA

- Eight computers:

  INTEL 80386SX microprocessor at 25 MHz

  4 MB RAM

  3Com EtherLink II / 16 network card /ISA

- One Tulip computer:

  INTEL 80386SX microprocessor at 25 MHz

  1 MB RAM

  3Com EtherLink II / 16 network card /ISA

- One Olivetti M24 computer:

  INTEL 8086 microprocessor at 8 MHz

  640 kB RAM

  3Com EtherLink II network card /ISA

- Two Victor computers

  INTEL 80486DX microprocessor at 50 MHz

  16 MB RAM

  Eagle NE2000 network cards

- Bridge

  This is described under its own category

A2.1.3 File Server on LAN A

This had NetWare 3.10 as the network operating system running on a Switch computer in dedicated mode. Other details are:

IBM PC/AT compatible computer with an INTEL 80486DX microprocessor at 33 MHz

4 MB RAM

MS-DOS 5.0

3Com EtherLink II network card /ISA

A2.1.4 File Server on LAN B

This had Advanced NetWare 2.15 as the network operating system running on an Olivetti M380 computer in non-dedicated mode. Other details are:

IBM PC/AT compatible computer with an INTEL 80386 microprocessor at 20 MHz

4 MB RAM

MS-DOS 5.0

NET5.COM version 3.10

IPX.COM version 3.01 Rev. A

3Com EtherLink II network card /ISA

A2.1.5 Bridge

This was an Olivetti M290S computer which was running the bridge software as a real mode bridge in non-dedicated mode. Other details of this computer are:

INTEL 80286 microprocessor at 16 MHz

1 MB RAM

MS-DOS 5.0

NET5.COM version 3.10

IPX.COM version 3.01 Rev. A

two 3Com EtherLink II network cards /ISA

A2.2 Test Programs

A2.2.1 TEST.C

```
/*
```

```
                                File: TEST.C
```

```
Project File should include the following files:
    test.c
    cnit.lib
    lib.vm
```

```
    This is a program to load a database server with requests for the
    main menu. On successful receipt of the menu, the program prints a
    dot in the first window. If the program times out (after 3 attempts),
    it will print 'X' in the first window. On retry, program prints a
    '*' in the lower window.

    By: S. J. Isaac

    date: 18 May 1993
```

```
*/


#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <stdlib.h>
#include <nit.h>
#include <nxt.h>


#include "h:\c\r\h\uvm.h"


//_____________________________________________________________________________
//                    Global variable declaration

SENDDB_struct g_sendDb;
RECEIVEDB_struct g_receiveDb;
DBTABLE_struct g_dbTable[MAX_DB_TABLE_POSITIONS];

ECB  g_ecbReceiveDb;
ECB  g_ecbSendDb;
IPXHeader g_ipxheaderSendDb, g_ipxheaderReceiveDb;

BYTE g_return_from_db_flag = 0;
WORD g_dbServerType;
BYTE g_numberDBservers = 0;
int  g_dbTablePosition = 0;
int  g_dbSocket;
WORD g_primaryConnectionID;



//_____________________________________________________________________________
//                    Declaration of Routines

void Initialise (int *dbTablePos);
BYTE GetIDRec (WORD ID, BYTE request);
BYTE GetMenu (WORD ID);
void far Esr_ReceiveDb (void);
```

```
//_____________________________________________________________________
//                    Main

void main (void)
{
  int i;

    // Set up windows

    clrscr();
    gotoxy (1,24);
    for (i=1; i<9; i++){
        textattr(i);
        cputs("1234567890");
    }
    textattr(0x07);
    window(1,1,80,23);


    // initialise

    Initialise (&g_dbTablePosition);


    // Repeatedly ask for main menu

    while (1){

        if (kbhit()) {
            cputs("\r\n\rUSER BREAK\r\n");
            break;
        }

        GetMenu (0);
    }

}


//_____________________________________________________________________
//                    Initialise

void Initialise (int *dbTablePos)
{
  int ccode;


    // initialize IPX

    ccode = IPXInitialize();
    if (ccode != 0){
        puts ("IPX is not installed !!");
        exit (1);
    }


    // initialise communication with the database server

    ccode = DBinit (&g_dbSocket,
                    &g_dbTable,
                    &g_numberDBservers,
                    &g_sendDb);
    if (ccode == 1){
        puts ("No Database servers could be located");
        exit (1);
    }

    *dbTablePos = 0;
}
```

```
//_____________________________________________________________________
//                    GetIDRec

BYTE GetIDRec (WORD ID, BYTE request)
{
  BYTE c;
  int flag=0;

    memcpy(g_sendDb.data, &ID, sizeof(WORD));
    g_sendDb.requestType = request;
    do {

        c = DBRequest (&g_receiveDb,
                       &g_sendDb,
                       &g_return_from_db_flag,
                       &g_dbTable,
                       g_dbSocket,
                       g_dbTablePosition,
                       Esr_ReceiveDb,
                       &g_ipxheaderSendDb,
                       &g_ipxheaderReceiveDb,
                       &g_ecbSendDb,
                       &g_ecbReceiveDb);

        if ((c==0xC1) || (c==0xCC)){


            // find another database server

            g_dbTablePosition++;

            if (g_numberDBservers <= g_dbTablePosition){


                // locate database servers one more time

                flag++;
                if (flag > 1) break;
                g_numberDBservers = LocateDBServer (&g_dbTable);
                if (g_numberDBservers == 0) {
                    puts ("No DB servers!");
                    exit (1);
                }
                g_dbTablePosition = 0;
            }

        }
        else break;

    } while (flag < 2);


    return c;
}


//_____________________________________________________________________
//                    GetMenu

BYTE GetMenu (WORD ID)
{
  BYTE ccode;
  int i,x,y;
  static int r,c;

    ccode = GetIDRec (ID, MENU_RECORD);

    if (ccode == 0){  // successful
        cprintf (".");

        if (g_sendDb.retry != 0){


            // there were a few retries

            x=wherex();
            y=wherey();
```

```
            // print retry in the retry count window

            window(1,25,80,25);
            gotoxy (r,c);
            for (i=0; i<g_sendDb.retry; i++) cprintf("*");
            r=wherex();
            c=wherey();


            // go back to main window

            window(1,1,80,23);
            gotoxy (x,y);
        }
    }
    else cprintf ("X");  // failed to receive anything from server

    return ccode;
}


//_____________________________________________________________________________
//                    Esr_ReceiveDb

void far Esr_ReceiveDb (void)
{
    _AX = _ES;
    _DS = _AX;
    if (g_sendDb.requestNumber == g_receiveDb.requestNumber)
        g_return_from_db_flag = 1;
    else IPXListenForPacket( &g_ecbReceiveDb );
}
```

A2.2.2 TEST20K.C

```
/*
```

File: TEST20K.C

```
Project File should include the following files:
    test20k.c
    cnit.lib
    lib.vm
```

```
    This is a program to load a database server with requests for the
    main menu. The program prints statistics after every set of results
    for COUNT sets.

    Each set consists of RESPONSES number of readings


    By: S. J. Isaac

    date: 23 May 1993

*/


#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <stdlib.h>
#include <dos.h>
#include <nit.h>
#include <nxt.h>


#include "h:\c\r\h\uvm.h"



#define COUNT     10
#define RESPONSES 20000
#define TIME_OUT  1.5



//_____________________________________________________________________________
//                    Global variable declaration

SENDDB_struct g_sendDb;
RECEIVEDB_struct g_receiveDb;
DBTABLE_struct g_dbTable[MAX_DB_TABLE_POSITIONS];

ECB  g_ecbReceiveDb;
ECB  g_ecbSendDb;
IPXHeader g_ipxheaderSendDb, g_ipxheaderReceiveDb;

BYTE g_return_from_db_flag = 0;
WORD g_dbServerType;
BYTE g_numberDBservers = 0;
int  g_dbTablePosition = 0;
int  g_dbSocket;
WORD g_primaryConnectionID;



//_____________________________________________________________________________
//                    Declaration of Routines

void  Initialise (int *dbTablePos);
BYTE  GetIDRec (WORD ID, BYTE request);
BYTE  GetMenu (WORD ID, int *retry);
void  far Esr_ReceiveDb (void);
```

```
//_____________________________________________________________________
//                    Main

void main (void)
{
  int i, retry[COUNT], fail[COUNT], count;
  FILE *fp;
  BYTE ccode;
  struct time start, end;
  float diff[COUNT], igRetry[COUNT];
  float t_diff=0, t_igRetry=0, t_retry=0, t_fail=0;
  float a_diff=0, a_igRetry=0, a_retry=0, a_fail=0;
  float one_diff=0, one_igRetry=0;


    clrscr();


    // initialise

    Initialise (&g_dbTablePosition);

    if ((fp=fopen("C:\\TEMP\\TEST.DAT","w+"))==NULL){
        perror("could not open test.dat file");
        exit(1);
    }


    // print titles

    printf("Number of responses = %i\n", RESPONSES);
    printf("Time-out value = %4.2f seconds\n\n", TIME_OUT);

    printf("COUNT\tTIME [s]  \tT-Retry [s]  \tRETRIES      \tFAILS\n");
    puts("-----------------------------------------------------------------");


    // Ask for and receive 20000 main menus TEN times

    for (count=0; count<COUNT; count++){
        i = 0;
        retry[count] = 0;
        fail[count] = 0;

        gettime (&start);

        while (i < RESPONSES){

            if (kbhit()) {
                puts("\nUSER BREAK");
                exit (1);
            }

            ccode = GetMenu (0, &retry[count]);
            if (ccode == 0) i++;
            else fail[count]++;
        }

        gettime (&end);

        diff[count] = DiffFloatTime (&end, &start);
        igRetry[count] = diff[count] - (retry[count] * TIME_OUT);

        t_diff += diff[count];
        t_igRetry += igRetry[count];
        t_retry += (float)retry[count];
        t_fail += (float)fail[count];

        printf("%i\t%4.2f      \t%4.2f      \t%i\t\t%i\n",count+1, diff[count],
               igRetry[count], retry[count], fail[count]);
    }


    // set averages

    a_diff = t_diff / count;
    a_igRetry = t_igRetry / count;
    a_retry = t_retry / count;
    a_fail = t_fail / count;
```

```
    // set average response time for one request

    one_diff = a_diff / RESPONSES;
    one_igRetry = a_igRetry / RESPONSES;

    puts("-------------------------------------------------------------------");
    printf("TOTALS\t%4.2f      \t%4.2f      \t%4.0f\t\t%4.0f\n", t_diff, t_igRetry,
           t_retry, t_fail);
    puts("-------------------------------------------------------------------");
    printf("AVERAGE\t%4.2f      \t%4.2f      \t%4.1f      \t%4.1f\n", a_diff,
            a_igRetry, a_retry, a_fail);
    puts("-------------------------------------------------------------------");
    printf("AVERAGE\nFOR ONE\nRESP.\t%f      \t%f\n", one_diff, one_igRetry);


    // write to file

    fprintf(fp,"Number of responses = %i\n", RESPONSES);
    fprintf(fp,"Time-out value = %4.2f seconds\n\n", TIME_OUT);

    fprintf(fp,"COUNT\tTIME [s]  \tT-Retry [s]  \tRETRIES      \tFAILS\n");
    fputs("-------------------------------------------------------------------\n",fp);

    for (count=0; count < COUNT; count++)
        fprintf(fp,"%i\t%4.2f      \t%4.2f      \t%i\t\t%i\n",count+1, diff[count],
                igRetry[count], retry[count], fail[count]);

    fputs("-------------------------------------------------------------------\n",fp);
    fprintf(fp,"TOTALS\t%4.2f      \t%4.2f      \t%4.2f\t\t%4.2f\n", t_diff,
            t_igRetry, t_retry, t_fail);
    fputs("-------------------------------------------------------------------\n",fp);
    fprintf(fp,"AVERAGE\t%4.2f      \t%4.2f      \t%4.2f      \t%4.2f\n", a_diff,
            a_igRetry, a_retry, a_fail);
    fputs("-------------------------------------------------------------------\n",fp);
    fprintf(fp,"AVERAGE\nFOR ONE\nRESP.\t%f      \t%f\n", one_diff, one_igRetry);
    fclose (fp);


    // sound 'THE END'

    sound (1000);
    delay (2000);
    nosound();

}



//_____________________________________________________________________________
//                    Initialise

void Initialise (int *dbTablePos)
{
  int ccode;


    // initialize IPX

    ccode = IPXInitialize();
    if (ccode != 0){
        puts ("IPX is not installed !!");
        exit (1);
    }


    // initialise communication with the database server

    ccode = DBinit (&g_dbSocket,
                    &g_dbTable,
                    &g_numberDBservers,
                    &g_sendDb);
    if (ccode == 1){
        puts ("No Database servers could be located");
        exit (1);
    }

    *dbTablePos = 0;
}
```

```
//_____________________________________________________________________________
//                      GetIDRec

BYTE GetIDRec (WORD ID, BYTE request)
{
  BYTE c;
  int flag=0;

    memcpy(g_sendDb.data, &ID, sizeof(WORD));
    g_sendDb.requestType = request;
    do {

        c = DBRequest (&g_receiveDb,
                       &g_sendDb,
                       &g_return_from_db_flag,
                       &g_dbTable,
                       g_dbSocket,
                       g_dbTablePosition,
                       Esr_ReceiveDb,
                       &g_ipxheaderSendDb,
                       &g_ipxheaderReceiveDb,
                       &g_ecbSendDb,
                       &g_ecbReceiveDb);

        if ((c==0xC1) || (c==0xCC)){


            // find another database server

            g_dbTablePosition++;

            if (g_numberDBservers <= g_dbTablePosition){


                // locate database servers one more time

                flag++;
                if (flag > 1) break;
                g_numberDBservers = LocateDBServer (&g_dbTable);
                if (g_numberDBservers == 0) {
                    puts ("No DB servers!");
                    exit (1);
                }
                g_dbTablePosition = 0;
            }

        }
        else break;

    } while (flag < 2);


    return c;
}



//_____________________________________________________________________________
//                      GetMenu

BYTE GetMenu (WORD ID, int *retry)
{
  BYTE ccode;
    ccode = GetIDRec (ID, MENU_RECORD);
    *retry += g_sendDb.retry;
    return ccode;
}


//_____________________________________________________________________________
//                      Esr_ReceiveDb

void far Esr_ReceiveDb (void)
{
    _AX = _ES;
    _DS = _AX;
    if (g_sendDb.requestNumber == g_receiveDb.requestNumber)
        g_return_from_db_flag = 1;
    else IPXListenForPacket( &g_ecbReceiveDb );
}
```

APPENDIX B1

VM DATABASE ADMINISTRATOR'S MANUAL

Action Summary

As the VM database administrator you will perform the following actions:

1. Keep all the files and programs you will use on a file server (in a directory called VM_DB at the root level of the SYS volume). Login name =VM_DATABASE.
2. Set-up the VM database
3. Set-up the VM database servers
4. Maintain the VM database
5. Receive information from and co-ordinate all the file server administrators.

Please refer to the documentation entitled "Information on the Virtual Machine".

B1.1 Set-up the VM Database

The VM database files are to be located in a directory on any file server called SYS:VM_DB. This must be the home directory for a user called VM_DATABASE who will be the VM database administrator. The programs CDB.EXE, DB.EXE, LOADDB.EXE and BATCH.EXE should be copied to this directory.

Before setting up the database the following will have to be done:

- Obtain the names of the batch files that will start applications and the file servers that they reside on from the file server administrators.
- Design the structure of the menu.
- Determine the VM group access rights (in collaboration with the file server administrators) that should be assigned to each application.

The database is set-up using the program called CDB.EXE (CDB is derived from the words CREATE DATABASE). It is best to use a PC with a colour screen and a mouse when using the CDB program. This program is described below.

The program is started by typing CDB at the DOS command line (It does not take any parameters). On start-up, the 'FILES MENU' with the following five choices will be seen:

1. Ap_acces.dbf
2. Ap_batch.dbf
3. User.dbf
4. Menu.dbf
5. Password.dbf

You can exit the program when you see the FILES MENU by hitting the ESC key.

Item three is not used at present in the VM and so should not be selected. The highlight can be moved using either the cursor keys or by typing the number of the choice or by moving the mouse pointer over one of the choices. Selection is made by either hitting the ENTER key or by clicking either the left or the right mouse buttons.

B1.1.1 Ap_acces.dbf

This is the file that contains information about application names and access rights to that application by the VM groups.

If this is selected, a data entry screen with five columns will be seen. Pressing the F1 key will provide help information. You can move around the fields using either the mouse or the keyboard (refer to the information on the help screen if you want to use the keyboard). Below is a description of these fields:

1. The first column is the application ID (or appID) field. The appID field is automatically filled for you in hexadecimal notation corresponding to the row number (starting from one). Each row is a record in the AP_ACCES.DBF file. You will not be able to type anything into this field.

2. The second column is the Application Name field and should contain a descriptive name of the application. The name can be up to 20 characters long.

3. The third column is the 'Access Rights' field and should contain the access rights for the VM groups. This field is one byte wide and the bits represent the VM groups that can start the application. Each bit number (from zero up to seven) represents a VM group number. If all the bits are zero (i.e. a byte value of zero) then only a person who has been defined as belonging to the unlimited access VM group can start that application. Such persons can start the application no matter what the byte value of this field is.

   Suppose the following three groups have been defined:

   - STUDENT (VM group = 0)
   - LECTURER (VM group = 1)
   - MANAGER (VM group = 3)

   If a value of 3 in hexadecimal (00000011 in binary) is placed into the Access Rights field, then users in both the STUDENT and LECTURER VM groups will be able to start the application because bit zero and bit one both have the value one.

If a value of 1 in hexadecimal (00000001 in binary) is placed into the Access Rights field, then only users in the STUDENT VM group will be able to start the application because only bit zero has the value one.

If a value of 2 in hexadecimal (00000010 in binary) is placed into the Access Rights field, then only users in the LECTURER VM group will be able to start the application because only bit one has the value one.

4. The fourth column is the Operating System field and is one byte long. This field is not used at present.

5. The fifth column is the Hardware field and is two bytes long. This field is not used at present.

To return to the FILES MENU, hit the ESC key. A pop-up menu with the following two options will appear:

Save then EXIT

EXIT

As in the FILES MENU the highlight can be moved by either hitting the first letter, or the cursor keys or by the mouse. The first option will save the changes made in to the AP_ACCES.DBF file. The other option will return you to the FILES MENU without saving.

****

IMPORTANT! *Do not insert or delete any row in the screen unless you want to (manually) modify the MENU.DBF file as well to reflect all the changes you are making. If you want to remove an application, use the reset command on that row (by pressing F5).*

****

B1.1.2 Ap_batch.dbf

This is the file that contains information about where the applications are located and the names of the batch files that will start them.

If this is selected, a data entry screen with four columns and a menu containing a list of applications will be seen. The list of applications have been provided to minimise the possibility of mistake when filling the field in the data entry screen. An application from the list can only be selected with a mouse. Pressing the F1 key will provide some useful keyboard help information. You can move around the fields in the data entry screen using either the mouse or the keyboard (refer to the information on the help screen if you want to use the keyboard). Whenever you are in field that you can edit, the name of the field will appear at the lower left corner of the data entry screen. The order in which the rows are filled does not matter. Below is a description of the fields in the data entry screen:

1. The first column displays the record number (starting from zero) corresponding to the displayed row. This field is automatically filled for you in decimal notation. Each row is a record in the AP_ACCES.DBF file. You will not be able to type anything into this field.
2. The second column is the 'Resident Server Name' field and should contain a two-letter name of the file server where the application batch file is located.
3. The third column is the 'Application ID' field and should contain the appID for the application. This field is two bytes wide. This field can be filled by:
   - Selecting the required application from the application list menu by moving the mouse to the application name there.
   - Click the right mouse button on the desired application to fill an internal buffer with the application ID (you will not see this buffer).

- Move the mouse pointer back to the data entry screen and click the right mouse button over the row containing the application ID field that has to be filled.

4. The fourth column is the 'Batch File Name' field. The name of the batch file (including extension) in the 'SYS:PUBLIC/BIN' directory of the server specified in the second column of that row should be typed into this field.

To return to the FILES MENU, hit the ESC key. A pop-up menu with the following two options will appear:

Save then EXIT

EXIT

As in the FILES MENU the highlight can be moved by either hitting the first letter, or the cursor keys or by the mouse. The first option will save the changes made in to the AP_BATCH.DBF file. The other option will return you to the FILES MENU without saving.

****

IMPORTANT! *Remember to run the program called BATCH.EXE if the AP_BATCH.DBF file is modified in any way. BATCH.EXE sorts the AP_BATCH.DBF file and sets up an index file called AP_BATCH.NDX that speeds up the accessing of information from AP_BATCH.DBF.*

****

B1.1.3 User.dbf

This option has no use at present. If this option is selected, you can return to the FILES MENU by hitting the ESC key. A pop-up menu with the following two options will appear:

Save then EXIT

EXIT

As in the FILES MENU the highlight can be moved by either hitting the first letter, or the cursor keys or by the mouse. The first option will save the changes made in to the USER.DBF file. The other option will return you to the FILES MENU without saving.

B1.1.4 Menu.dbf

This is the file that contains all the menus displayed to the user.

If this is selected, a data entry screen called the Menu Editor, a menu called Applications List and a menu called Menu list is shown. All the windows shown can be resized using the mouse. To resize a window, move the mouse pointer to a corner of the window and click and drag the pointer using the left mouse button. To move a window, position the mouse pointer on any border of that window except the corner and click and drag using the left mouse button.

The two menus have been provided to minimise the possibility of mistake when filling in the fields in the data entry screen. There is no help information for this screen. Some preparation is required before creating the menu. It is best to understand the contents of the menus and the relationship between the menus before trying to create the menus with CDB.

The Menu Editor really consists of two parts; the data entry part on the left and the command buttons on the right. Where ever the highlight is in the Menu Editor screen, the name of the field (or an instruction if the highlight is on a button) will appear at the lower left corner of the Menu Editor screen. Everything on the Menu Editor is selected by moving the mouse pointer over the required field or button and clicking the left mouse button (the highlight does not follow). Anything from the two menu screens is selected by moving the mouse pointer over the desired menu or application name and clicking the right mouse button (the highlight will follow). Whatever has been selected from the two menu screens can be inserted in the name field in the Menu Editor by clicking the right mouse button when the mouse pointer is in the desired position. If

the name of an application has been placed in the name field, the letter 'A' will appear to the left of the application name and the application ID will appear to its right. If the name of a menu has been placed in the name field, the letter 'M' will appear to the left of the application name and the menu ID will appear to its right.

A description of the buttons now follow:

- NEXT: If this command is selected, the menu with a menuID that is one greater than the menuID of the current menu will be shown. If the current menu is the last menu, then no change will be observed.
- PREVIOUS: If this command is selected, the menu with a menuID that is one less than the menuID of the current menu will be shown. If the current menu is the first menu, then no change will be observed.
- APP LIST: This allows the highlight to move into the Applications List menu from the keyboard.
- MENU LIST: This allows the highlight to move into the Menu List menu from the keyboard.
- GOTO: ##: You can move to the menu with the menuID ## by supplying a hexadecimal value for ##. If the value you supply is greater than the last menuID, then you will be taken to the last menu. If the value you supply is less than the first menuID, then you will be taken to the first menu.
- APPEND: Allows a new record to be appended and moves you into that menu.
- RESET MENU: Will delete all the contents of the current menu and remove all references to this menu from all the other menus.
- RESET ITEM: If this is selected, then an entry in the name field can be reset to NULL values.
- RESET appID: ##: Will reset all references to the supplied appID from all the menus.

- DELETE: Will delete the current menu. USE WITH CARE!
- INSERT: Will insert a new menu into the current position. USE WITH CARE!

Following are some useful keys that will achieve similar results to some of the command buttons:

- ENTER: To make a selection
- UP and DOWN cursor keys: To move among the fields and the command buttons.
- F2: Pastes whatever has been selected from the Applications List or Menu List menus (or NULL values if the RESET button was selected) into the name field.
- PAGE UP: To go back a menu (similar to PREVIOUS)
- PAGE DOWN: To go to the next menu (similar to NEXT)
- ESC: To return to the Menu Editor screen if in one of the List menus or to go back to the FILES MENU if already in the Menu Editor.

B1.1.5 Password.dbf

This file contains all the information about the VM groups, the VM group login name for file servers and their base-password. The base-password is expanded to 125 characters by the VM menu program for the user.

If this option is selected a data entry screen called the Password Screen will be shown. Every record in the file is displayed on the screen in two rows separated by a horizontal line. There is no help information for this screen. The displayed fields for each record are:

- ACCESS: Indicates which record number this is (starting from zero). This is also sets the VM group number to be used in conjunction with the Access Rights field in the AP_ACCES.DBF file. The last record in the PASSWORD.DBF file will always have the ACCESS field set to FF in hexadecimal. This is the unlimited access group.

- GROUP: The name of the VM group.
- USER: The name by which the members of the VM group will be logged into other file servers.
- The last field in the record is for the base-password. This can be up to 128 characters long.

To return to the FILES MENU, hit the ESC key. A pop-up menu with the following two options will appear:

Save then EXIT

EXIT

As in the FILES MENU the highlight can be moved by either hitting the first letter, or the cursor keys or by the mouse. The first option will save the changes made in to the PASSWORD.DBF file. The other option will return you to the FILES MENU without saving.

B1.2 Set-up the VM Database Servers

The computers to act as database servers are selected. Good choices are IBM PCs or compatibles with a lot of RAM (around 4 Mbytes). The program called DB.EXE to start the VM database server is then run on each computer which is to be a server. The DB program can be obtained from the network by logging in as VM_DATABASE to the file server where these files are located.

Syntax: DB [<file server name>]

If the file server name is not specified, the program will look for the files in the directory from which DB was run. If the file server name is specified, then the DB program will log into the file server specified. You will be asked for user VM_DATABASE's password for that file server. You will also be asked to type in a password that should be supplied from the keyboard whenever you wish to terminate

the VM database server program. After loading the database files into memory, DB will log out of the specified file server.

B1.3 Maintain the VM Database

From time to time, the database will have to be modified. This can be achieved using the program CDB.EXE that has already been explained above. After the necessary changes have been incorporated, the new database has to be loaded into the database computers. There is a program called LOADDB.EXE that can do this. This program will bring down each database server then load the new database and restart the database server. It will do this for every database server on the network.

Syntax: LOADDB <file server name>

The file server name has to be supplied and this specifies where the VM database files are located. An error report will be displayed on the screen after completion for each VM database server that has been updated.

B1.4 Receive Information From and Co-ordinate All the File Server Administrators.

The file server administrators provide information on the names of the batch files to start the applications on the file servers they are in charge of. You will provide information as to what applications are already available in the VM so that there will not be any unnecessary duplication. Collectively, the access rights, the names of the groups, etc., will have to be determined.

APPENDIX B2

FILE SERVER ADMINISTRATOR'S MANUAL

Action Summary

The file server administrator will have to set up the file server according to Novell's recommendations and in addition will have to include the following steps.

1. give the file server a unique two letter name
2. put the map root H:= <home directory> command in the system login script
3. copy VM programs to relevant directories
4. create special groups that will allow virtual machine access
5. place every user into one (and only one) of the special groups
6. create application batch files and place it in the 'sys:public/bin' directory.
7. inform the database administrator about the batch files
8. Rename Novell's login and logout programs to IN180393.EXE and OT180393.EXE respectively.

Please refer to the documentation entitled "Information on the Virtual Machine".

B2.1 File Server Name

In the VM, having long descriptive file server names has no value because users do not have to know these names. The file servers on the network are given unique two letter names. This is the shortest name length permitted by NetWare. The main reason for this is to reduce the number of characters that a user will have to type to login to a file server. In a single file server setting and in the original NetWare environment, the user would type:

```
LOGIN USERNAME
```

Now the user has to type:

```
LG SN/USERNAME
```

(where SN is the two letter server name and LG is the login command in the VM).

The total number of characters that the user has to type remains the same in both environments. Please consult the VM database administrator for a two letter file server name that will be unique in the VM.

B2.2 Set the H: drive

Place the following command into the system login script:

```
MAP ROOT H: = <path to home directory>
```

(where path to home directory is replaced with the real path to the home directory. If the first available drive is F:, and this has been set to the user's home directory, then the command would be `MAP ROOT H: = F:`)

The purpose of this instruction is to enable the VM programs to locate the user's home directory at the root level of drive H:.

B2.3 Copy All the VM Programs to the File Server

There is a batch file provided to copy files to relevant directories for the operation of the VM. It is called VM_COPY.BAT. You have to be defined as a SUPERVISOR on the server to be able to run this batch file. Before running this batch file, create a

directory on your file server called SYS:PUBLIC/BIN. Most of the copied files will be located there

Syntax: VM_COPY <full path to location of files> <server name>.

The contents of VM_COPY.BAT are listed below so that you will know what new programs have been loaded into the file server's hard disk.

```
@ECHO OFF
if ?==%1 goto message
if .==.%1 goto message
goto transfer

:message
echo usage: vm_copy (path to files) (server name)
goto end

:transfer
ncopy %1\vm_menu.exe %2\sys:public\bin
ncopy %1\lg.exe %2\sys:public\bin
ncopy %1\lgo.exe %2\sys:public\bin
ncopy %1\mp.exe %2\sys:public\bin
ncopy %1\regvm.exe %2\sys:public\bin
ncopy %1\getparam.exe %2\sys:public\bin
ncopy %1\vm_creat.exe %2\sys:public\bin
ncopy %1\vm.bat %2\sys:public\bin
ncopy %1\lg.exe %2\sys:login
ncopy %1\lgo.exe %2\sys:login

:end
@ECHO ON
```

An example of its usage (if all the listed files are present in a directory called VM on the floppy disk in drive A:) is:

```
VM_COPY A:\VM EG
```

B2.4 Special VM Groups

These groups can be created using the VM_CREAT.EXE program. You have to have logged in as the supervisor to use this program. VM_CREAT.EXE is in the SYS:PUBLIC/BIN directory.

Syntax: VM_CREAT [<password file>]

If the password file name is not supplied, VM_CREAT will request the relevant information from a VM database server in which case there must be at least one VM database server operational in the network. It is best to get the information from the VM database server. If the operation has been successful, you will not receive any error information at all. Some new groups and some new users have been created on your file server. The groups are VM groups and the details can be obtained from the VM database administrator.

B2.4.1 Users

Each user is created on only one file server in the network. The user's login name should be unique on that file server. When the user is informed about his login name however, he is told a small lie. The login name given to the user will be of the format: <server name>/<user name>, where he only has to supply <user name> if his PC is already attached to his home server. If the user is moved to another file server, his given login name will have to change. All users that have been created on your file server are placed in one of the VM groups. Each user should only belong to one. The group they belong to determines their access category. Then restrictions are imposed on the special users that were created with the VM_CREAT program. Restrictions are set as with any other user or group.

The special users have the following account restrictions:

- unlimited account
- unlimited number of simultaneous connections
- unlimited time restriction
- special users are not allowed to change their password

B2.5 Application Batch Files

All programs are started from a batch file that prepares the MS-DOS environment and creates search mappings that will permit the application to locate files it needs for its operation. In this way the actual access is made easy for the user. When an application is added or deleted from the file server the database administrator has to be informed to make the necessary changes to the VM database. Batch files should be placed in the SYS:PUBLIC/BIN directory.

B2.5.1 Structure of Application Batch Files

```
@ECHO OFF
regvm i <name of server>
mp i <full path specification to set as a search drive>
.
.
.
mp d <full path specification to set as a search drive>
regvm d <name of server>
ECHO ON
```

Example: Loadf.bat is the batch file to start the application Loadflow.

```
@ECHO OFF
regvm i EG
mp i EG/SYS:AP/LOADFLOW
getparam
```

```
LOADFLOW %VMPARAM%
mp d EG/SYS:AP/LOADFLOW
regvm d EG
ECHO ON
```

The batch file should never include drive letters in relative path specifications. The only exception to this is the 'H:' drive. This is because it is difficult to determine at this time which directory all other network drives are mapped to. All path specifications should also include the server name if possible.

A description of the operation of the batch file now follows:

REGVM is a program that changes a server's attachment count in the Server Attachment Count (SAC) file. The second line of the 'loadf.bat' file would increment the count for the server named 'EG'. (The SAC file keeps relevant information about a user in a location specified by the TEMP environmental variable. Each session will have new information. The SAC file is created by the LOGIN program (called LG) at the beginning of a session and is deleted by the LOGOUT program (called LGO) at the end of a session.)

MP is a program that is essentially a replacement for Novell's MAP utility when working with search drives. The third line in the batch file increments a drive attachment count to the directory 'EG/SYS:AP/LOADFLOW'. If a search drive mapping to this directory does not exist, it creates one at this stage. MP also sets the MS-DOS path variable in all the environments up to the original parent.

The fourth line sets the environmental variable VMPARAM to any command line parameters that should be supplied to the program 'Loadflow'. The variable VMPARAM is set in all the environments up to the original parent. 'Getparam' enables programs that could not otherwise be started from a menu to be started from there. 'Getparam' should be used with care. If users do not usually supply command line

parameters to the program the batch file is trying to start, they might find it irritating to be queried for them.

The fifth line starts a program called LOADFLOW and supplies to it the command line parameters found in the environmental variable VMPARAM. If 'Getparam' had not been used in the preceding line, the fifth line would change to simply LOADFLOW.

When this program terminates, the sixth line will decrement the drive attachment count in the SAC file to the indicated directory and delete the search mapping if the drive attachment count is zero. The MS-DOS path variable is also edited to remove reference to the specified directory if the drive attachment count is zero.

The seventh line decrements the server attachment count. This enables a PC to determine when to disconnect from a file server.

B2.5.2 Batch File Names

As far as possible, batch files to start applications should be given unique names. All the file servers that have a particular application can have a batch file with the same name to start up that particular application, but different applications should have different batch file names to start them up. That way, if a user is attached to more than one server, he will not inadvertently start up the wrong application with that batch file name. Since the user starts up the application of choice through a menu, the name of the batch file is unimportant to the user. Even if the application changes location, the user is shielded from that change as long as he selects that application from the menu.

B2.6 Relationship With Other Administrators

As a file server administrator, you will have to sit on a committee that supervises the operation of the VM. This committee will consist of all the file server administrators and VM database administrator. Together, you will have to define the VM groups and the access capabilities of the groups toward the applications in the VM.

B2.7 Rename Novell's Login and Logout Programs

Novell's login program LOGIN.EXE should be renamed to IN180393.EXE and the logout program LOGOUT.EXE should be renamed to OT180393.EXE. This operation is performed so that the ordinary user will no longer have access to the original utilities because these programs are are not VM friendly. The VM login and logout programs actually use Novell's programs and search for the renamed programs. These programs must be located in the LOGIN directory and the PUBLIC directory of the file server.

APPENDIX B3

USER'S MANUAL

The Virtual Machine (VM) allows users to locate and start applications that they are allowed to use through a menu system. This user's manual gives pertinent instructions on the use of the system. If you experience any difficulty or problems in the use of the VM, please report them to the network administrator who supplied you with your username.

The following three important steps summarise the use of the VM:

1. Login - to enter the system
2. Run VM - to get the menu of applications
3. Logout after using the system

B3.1 Login — to Enter the System

Each session is begun by logging in the network system. To do this, type:

```
LG USERNAME <ENTER>
```

where LG is the command to login and USERNAME is the name by which you are known to the system. The USERNAME is given to you by a network administrator. After this you will be requested to supply your password. The password you type in will not be displayed on the screen. This is a security feature. If your login attempt is

successful, you will be able to proceed to step two; if not, step one will have to be repeated. Example: `LG EG/BANDA`

B3.2 Run VM — to Get the Menu of Applications

A menu of the applications that are available can be obtained by typing:

VM <ENTER>

Your choice on the menu can be selected using either the mouse or the cursor keys. If you use the mouse, then move the mouse pointer over your choice, and when your choice is highlighted click the left mouse button. If you use the cursor keys, then move the highlight over the appropriate choice and then press the ENTER key. Instead of using the cursor keys, you can also select your choice by pressing the key that has the number of choice on the menu and when that choice is highlighted, pressing the ENTER key.

In this manner, you can step through the menus until a menu containing the desired application is selected. At this stage, selecting the desired application will result in that application being started if you are allowed to start that application. If the application is not started, it means that you have not been given permission to use that application by the network administrators.

If you have not started an application and you want to go back to a previous menu, you can step backwards through the menus by hitting the ESC key. If you hit the ESC key at the main menu, you will exit the menu program.

Typing VM again at the command line will restart the menu program.

B3.3 Logout After Using the System

Each session is ended by logging out of network system. To do this, type:

LGO <ENTER>

It is important that you remember to logout of the system when leaving. This will prevent unauthorised access of data in your directories on the VM.

APPENDIX B4

VM APPLICATION DEVELOPER'S MANUAL

The Virtual Machine (VM) is a computer network system built around Novell NetWare that enables users to transparently access applications located on different file servers and allows an easy and scalable management of the system. The technique used is:

- Every user in the network is divided into one of several VM groups according to access rights to applications.
- Each VM group has a VM group login name to other file servers.
- PC software running on the users PC will present a menu of applications to the user and depending on his choice will use the VM group login name (of the VM group that he belongs to) to login to the file server where the chosen application is located.
- The PC software receives all required information from a VM database server.

As an VM applications developer, you will be creating programs that will run on a user's PC and will communicate with the VM database server. Knowledge of the C programming language and the NetWare C Interface for DOS is required. If your application requires data not presently found in the VM database, you will have to modify the VM database server program. Instructions on how to do this is found in section B4.1.

The library of routines that are provided with this manual makes the communication with the VM database server easier. Each function in the library is listed alphabetically in section B4.2 with explanatory notes on its purpose and usage.

B4.1 Modifying the VM Database Server

Please refer to the following documentation:

- Information on the VM
- The VM Database Server Program — DB.C

If the application you are developing requires data not found in the database, you will have to write an application that will create the database. This will need an interface for the VM database administrator to manage the data. All the information in the new database file will have to be in a format that can be represented by an array of C structures. If you are using a commercial Database Management System (DBMS) then you will need to write a program that will convert the data into a format representable by a C structure.

If this file is to be loaded into the memory of the VM database server PC, do the following:

1. Insert a line in the UVM.H file that will define the service. Examples of current services are:

```
#define MENU_RECORD             2       /* Menu service */
#define APPLICATION_RECORD      3       /* Application service */
#define PASSWORD_RECORD         4       /* Password service */
#define LOAD_DATABASE           5       /* Load new copy of the
                                           database files service */
```

   The new service could be defined as (for example):

```
#define NEW_DB_RECORD           7       /* New service */
```

2. Declare a far pointer (as a global variable) in DB.C to the structure defining one record of the new database file. An example of a current declaration is:

```
struct apBatch
    {
    char server[SERVER_NAME_LENGTH];
    WORD appID;
    char batchFile[13];
    }far *g_apBatch;      /* A far pointer to one record of the
                             file Ap_batch.dbf stored in RAM    */
```

The new structure pointer could be declared as (for example):

```
struct newDBFile
    {
    char field_1;
    char field_2[24];
    BYTE field_3;
    int  field_4;
    }far *g_newFile;      /* A far pointer to one record of the
                             file New_file.dbf stored in RAM    */
```

3. Define a routine that will service requests from PCs for information from this new database file and append this routine to the end of the code in the DB.C file. An example of a currently defined routine is:

```
/* The menu service routine */

void GetMenu (void)
{
  WORD menuID;

    /* Pick up the request from the receiveWs queue */

    menuID = *(WORD *) &g_receiveWs[ g_serviceQueuePtr ].data[0];

    /* Process it and place the response in the sendWs buffer */

    if (menuID <= g_maxMenuID){
        memcpy(g_sendWs.data, &g_menu[menuID],
               sizeof (struct menu));
        g_sendWs.responseType = 0;

    }
    else g_sendWs.responseType = 1;

    /* Notice that the response type has also been set above */
}
```