REMOTEBOOT OF WINDOWS 95/98 FROM OS/2 WARP SERVER

MICHO ĐURĐEVICH

Abstract. We explain how to set up diskless Windows 95/98 workstations that remote boot from OS/2 Warp Server, using the 802.2 RIPL Service. We present several different configurations of Windows RIPL clients, and discuss a couple of interesting problems appearing in the game. The basics of the OS/2 Warp Server RIPL Service is presented, too.

1. Introduction

This article is devoted to studying a very interesting way of integrating Windows with OS/2 Warp Server working environment. We shall explain in detail how to set up a remote boot diskless workstation to load Windows 95/98 (together with the appropriate applications) directly from an OS/2 Warp Server. This opens an exciting new possibility to extend IBM's and Serenity Systems' OS/2-based managed client philosophy to Win32 platforms.

In other words, we are going to set up RIPL (Remote Initialization and Program Load) for Win95/98 clients. The main strategy used here is relatively straightforward. The first real-mode phase of Win95/98 boot process is essentially a plain DOS. The transition to protected-mode regime occurs during the second boot phase, when the graphical part is loaded. Accordingly, our Win95/98 client will be understood, from the OS/2 Warp Server viewpoint, as a DOS client. This client will be configured appropriately, so it will be able to RIPL a full-blown version of Windows.

It should be noted however, that this is not a documented feature of Windows nor OS/2. And it is not free of problems (but all the problems I know about are really not critical). Windows 95 Resource Kit contains detailed instructions how to enable remote boot of Win95 clients, but for specific non-OS/2 servers (as Novell Netware or NT). Some of these ideas are used here in a modified form.

Our solution differs from IBM's WorkSpace On-Demand Feature for Windows, which also extensively uses the Warp Server RIPL subsystem. The main difference is that in our case the whole Windows operating system *resides on the server* and so the client workstations can be *diskless*.

In contrast, the WorkSpace On-Demand solution requires that the client Windows operating system (Win95, Win98 or WinNT) be installed on the client workstation (which obviously must have a hard disk for this purpose!). The mentioned installation process (which is server-driven and unattended) is relatively lengthy, and requires several reboots of the client. If the client fails, it is necessary to repeat the whole lengthy procedure again ... In this environment the RIPL is used [4]:

- During the client installation process, to load the operating system install image and other basic software from the server, and locally initiate the Windows installation process.

Supported by the United States-Mexico Foundation for Science.

- Subsequently (during normal operations) only to load from the server a small 'hybrid boot image' program which will read the first sector of the local hard disk and enable local booting of the corresponding Windows operating system.

Of course, our solution is not comparable in versatility, stability and WPS integration with WSOD-managed Windows workstations. However I hope this will change in time ... The article is organized in the following way:

We shall start by presenting the basics of the OS/2 Warp Server RIPL Service architecture. We shall consider both DOS and OS/2 RIPL clients, and analyze the basic server and client configuration/initialization files.

As already mentioned, our Windows RIPL clients will be interpreted, from the OS/2 Warp Server viewpoint, simply as DOS clients. However a number of additional steps should be made before DOS clients can properly boot Windows 95/98.

In Section 3 we shall discuss the main procedure for setting up a single Win95 client. The procedure will be presented in two 'basic flavors': The simplest case when all Windows files are on the server, and a slightly more complicated case when the client has a RAM-drive [C:] (imitating in a certain sense the hard-disk and introducing more flexibility in the game).

In Section 4 we shall briefly consider the problematics of a single Win98 RIPL client. Interestingly, it seems that the best Windows RIPL client is given by a *Lite version* of Win98, in which the operating system is cleaned-up from Internet Explorer components.

In Section 5 we shall talk about some interesting problems appearing with our Win95/98 RIPL installations, and it will be explained how to solve (some of) them.

Section 6 briefly discusses the 'clientization' problematics. It will be sketched how to make multiple clients to share a single Win95/98 tree on the server, and in particular how to set up machine directories on the server.

Finally, in Section 7 some concluding remarks and observations are made. The paper ends with two appendices. The first appendix discusses a 'purely Microsoft' networking solution, where the Microsoft redirection software is used during the DOS boot phase. A non-trivial step here is to create a mini-registry file, containing the appropriate information used by the Microsoft networking. In the second appendix we provide a 'blueprint' mini-registry file, suitable for editing and compiling.

In this paper we shall not consider diskless NT-clients. These RIPL configurations are more complex, and are built along the lines of the OS/2 RIPL clients. We shall discuss RIPLing WinNT in the extended version of the paper. The latest informations about RIPLing Windows from OS/2 can be found at the author's website.

Acknowledgments: *I am very grateful to Kim Cheung and Bob St. John for various technical and conceptual explanations regarding OS/2 RIPL, managed client solutions and related business models. Muchas gracias to Carlos de Luna for getting me interested in the managed client/RIPL aspects of Warp Server. I would also like to thank Bob Greenwald for interesting remarks concerning the use of Microsoft redirection software, and related problem of long filenames.*

2. OS/2 Warp Server 802.2 RIPL Service

2.1. **Installation**

The installation of OS/2 Warp Server RIPL Service is done in three steps. First, during the server installation, we should select the RIPL Support for OS/2 and DOS as a service to add. At this point we should fix the drive that will hold the

main RIPL-tree. The second step consists in invoking `RIPLINST` utility, that will create a copy of the appropriate version of OS/2 on the server, that will be shared among RIPL-clients. Finally, we should apply the Post-installation Utility Program `GETRPL.EXE`. Basically, this program adjusts access control profiles, creates/updates client initialization files, and ensures that necessary files are in the appropriate directories.

We refer to the OS/2 Warp Server documentation, for more informations about installing the RIPL Service.

2.2. **Boot Block Files**

In a very first phase of the remote-boot process, before the client's operating system starts, its network adapter communicates with the server and downloads (in the client memory) the minimal system of files necessary for a proper operating system initialization in the next phase (for this to work, the network adapter should be capable to perform a network boot using 802.2 RIPL). These files form a so-called *boot block*. The information about a boot block is stored in a file of the form `<name>.CNF` where `<name>` is the identifier of the requester class. During the boot block processing, the client runs in a kind of a 'pra-DOS' mode. The `CNF`-files are located in `\IBMLAN\RPL` directory of the drive where the RIPL service is installed.

Here is a sample `CNF`-file suitable for 3Com 905X PCI-adapters and DOS requesters:

```
BASE 7C0H
RPL DOS\RPLBOOT.SYS
LDR DOS\RPLLOADR.COM
DAT DOS\3C90XPCI\PROTOCOL.INI
DAT E:\IBMLAN\DOSLAN\LSP\DXM.MSG
EXE E:\IBMLAN\DOSLAN\LSP\NETBIND.COM ~ ~ ~
DRV E:\IBMLAN\DOSLAN\LSP\DXMT0MOD.SYS PBA=0~S=12~ST=12~C=14~O=N ~ ~
DRV E:\IBMLAN\DOSLAN\LSP\DXME0MOD.SYS ~ 10 ~
DRV E:\IBMLAN\DOSLAN\LSP\DXMA0MOD.SYS 001 ~ ~
DRV E:\IBMLAN\DOSLAN\LSP\DOS\EL90X.DOS ~ 3 ~
DRV E:\IBMLAN\DOSLAN\LSP\PROTMAN.DOS /I: ~ ~
```

A similar `CNF`-file for OS/2 requesters:

```
RPL DOS\RPLBOOT.SYS
DAT DOS\MFSD20.SYS
ORG 1000H
LDR OS2.40\OS2LDR ~ OS2LDR UFSD.SYS MFSD20.SYS
DAT DOS\UFSD.SYS
DAT DOS\3C90XPCI\OS2\PROTOCOL.INI
DAT E:\IBMLAN\DOSLAN\LSP\DXM.MSG
EXE E:\IBMLAN\DOSLAN\LSP\NETBIND.COM ~ ~ ~
DRV E:\IBMLAN\DOSLAN\LSP\DXMJ0MOD.SYS ~ 17 ~
DRV E:\IBMLAN\DOSLAN\LSP\DXMA0MOD.SYS 001 ~ ~
DRV E:\IBMLAN\DOSLAN\LSP\DOS\EL90X.DOS ~ 3 ~
DRV E:\IBMLAN\DOSLAN\LSP\PROTMAN.DOS /I: ~ ~
```

The paths that are not fully qualified are relative to the `IBMLAN\RPL` directory of the main RIPL drive. The instructions figuring in `CNF`-files have the following meaning:

`RPL` Only one `RPL`-line should be present. It specifies the device driver that controls the boot block execution and passes the control to the operating system loader. Normally, it is `RPLBOOT.SYS`. It is not surprising that this file is loaded first.

`LDR` Only one `LDR`-line should be present. It specifies the operating system loader. For OS/2 its `OS2LDR` and for DOS it is `RPLLOADR.COM` which reads the corresponding boot disk image.

`DAT` Special 'data files' containing the information used by other files in the boot block.

`DRV` Specifies device drivers loaded during the boot block processing phase. These lines are read in the reverse order, from the bottom to the top. In a certain sense they are analogous to `DEVICE` statements in the `CONFIG.SYS` of DOS.

`EXE` Specifies programs that should be executed. The lines are executed in the reverse order, and they correspond to the `AUTOEXEC.BAT` of DOS.

`BASE` This is for DOS requesters only, it specifies the hexadecimal segment number which is the base address for the boot block. Only one entry is allowed.

`ORG` For OS/2 requesters only, tells us the hexadecimal segment number of a contiguous memory address where the operating system should be loaded. Only one `ORG` entry should be present.

As we see, some of the `CNF`-lines end with *parameters* (associated with the corresponding boot block files) and tilde symbols. The tilde symbols are used to separate fields in these parameter lists.

2.3. **The Structure of RPL.MAP Control File**

This is the main configuration file for the OS/2 Warp Server RIPL Service. It is located in `\IBMLAN\RPL` directory of the drive where the RIPL service is installed. The file `RPL.MAP` has two parts: The *server records section* and the *clients records section*.

The first part consists of lines having the following format:

Φ_1 `<filename>.cnf` Φ_3 Φ_4 Φ_5 Φ_6 `<Descriptive~Comment>` `~ ~ ,,,` Φ_B Φ_C `~ ~`

where

Φ_1 A word consisting of twelve 'y' or twelve 'x' characters.

Φ_3 Number of retries before default boot takes place—this makes sense only if several RIPL servers are present.

Φ_4 Allowed time interval for client retries.

Φ_5 Acknowledge $\in \{A, N\}$.

Φ_6 For DOS it is the share name of the folder where `DCDB\IMAGES` is located. Normally, the value is `IBMLAN$`. For OS/2 the value is a single tilde character.

Φ_B The value of the LASTDRIVE variable for DOS (normally Z), and a single tilde for OS/2.

Φ_C Workstation type identifier.

The second part consists of lines describing single RIPL-requesters. Each line is of the form:

Ψ_1 `<WorkstationName>` `~` Ψ_4 `<RIPLServerName>` Ψ_6 Ψ_7 Ψ_8 Ψ_9 Ψ_A Ψ_B Ψ_C `~`

where

Ψ_1 The network adapter ID. This is a twelve-digit hexadecimal number.
Ψ_4 The boot image filename for DOS (without any path, that is specified in the server records section). The `FIT`-filename for OS/2 (with the paths relative to `IBMLAN\RPL`).
Ψ_6 Boot drive letter for OS/2, domain name for DOS.
Ψ_x Here $x \in \{7, 8, 9\}$. This is the parameter that should be passed to device drivers 1/2/3 of the LAN Support Program. These files are part of the boot block, and have the names of the form `DXM?0MOD.SYS`.
Ψ_A Additional memory for these device drivers, in kilobytes (the default value is '`,,,`').
Ψ_B A single tilde character for OS/2 and Z for DOS.
Ψ_C Workstation type identifier.

2.4. **OS/2-Requesters and FIT-Files**

The method that OS/2 Warp Server uses to manage the file/directory structure of OS/2 RIPL clients is very powerful and flexible. It is based on so-called File Index Tables. These `.FIT` files specify:

(*i*) Which files on the server are visible from the RIPL workstation;

(*ii*) How these files and directories will be represented on the client workstation. This is achieved by declaring (on a single/group basis) how local workstation file/directory names should translate into the 'real' server file/directory names.

The `FIT`-files are stored in `IBMLAN\RPL\FITS`. A `FIT`-file starts with a heading specifying a default network share. The rest of the file are the lines of the form:

```
PrototypeField      ServerTranslation
```

Here `PrototypeField` is a workstation file name (with a full path) or a directory (also, fully qualified). It is possible to use wildcard characters (?,*) in this field. The `ServerTranslation` field is the actual filename translation. It can be a full netname, or a location relative to the netshare specified in the heading of the file. The second field can not contain any wildcard characters. The wildcard characters in the first field can be used only for *files*, coupled with a *directory* location in the second field. It is not possible to use both * and ? within the same field.

Here is an example of a toy `FIT`-file:

```
\\AURORA\RPLFILES


Z:\CONFIG.SYS MACHINES\XOCHITL\CONFIG.SYS
Z:\OS2KRNL MACHINES\XOCHITL\OS2KRNL
Z:\OS2 OS2.45\OS2
Z:\OS2\SYSTEM\SWAPPER.DAT \\AURORA\WKRFILES\XOCHITL-SWAP\SWAPPER.DAT
Z:\ \\AURORA\WRKFILES\XOCHITL
Z:\PAPERS\*.TEX \\AURORA\WRKFILES\XOCHITL\MICHO
```

The directory translations imply all corresponding subdirectory translations, together with all files. However, this can be *overridden* by specifying separate subdirectory/file translations, as in the above example. Precisely, if we have several mutually matching prototypes compatible with a given workstation file system object, then the *longest* matching prototype takes the precedence.

2.5. **DOS-Requesters and Boot Disk Images**

In contrast to OS/2 requesters, DOS requesters do not use FIT-tables and initialize from a virtual floppy disk image. These image files have the extension `IMG` and are located in `DCDB\IMAGES` subdirectory of the main `IBMLAN` folder. The program used to make a disk image is `MAKEIMG.EXE` and it is located in the `NETPROG` directory of `IBMLAN`. It is possible to make a disk image in two ways: From a given bootable floppy, or using special boot disk description files together with a copy of DOS (that should be stored in the `DOSLAN\DOS` subfolder of `IBMLAN`).

There is a special 'filtering' mechanism, invoked by OS/2 Warp Server, that allows us to share one disk image among many clients of the same type. The mechanism translates certain very special variable strings figuring in critical files of a floppy disk image, into client-specific strings using the information stored in `RPL.MAP`.

The variable strings are of the form `~~~~~`m where the hex digit m refers to the workstation record in the `RPL.MAP` file. In addition, if the replacement string has more than 6 characters, we should add the appropriate amount of tilde characters in the variable (but the minimum number of tilde characters is 5, in order to invoke the filtering mechanism).

For example, a line of the following form is standardly present in `AUTOEXEC.BAT` of all DOS requesters:

```
CONNECT ~~~~~B ~~~~~5 ~~~~~2 ~~~~~6
```

If the server name is `AURORA`, the domain name `IMATH` and the workstation name `DOS98`, then the above line would appear to the client as

```
CONNECT Z AURORA DOS98 IMATH
```

which would basically establish `Z=\\AURORA\IBMLAN$` and `Y=\\AURORA\WRKFILES`. The program `CONNECT.EXE` allows a basic 16-bit redirection to work, without the full set of files for the 16-bit networking (which would not fit on a single floppy). The necessary additional networking files are

```
DLSHELP.SYS  NETWORK.INI  NET.EXE  NET.MSG  NETH.MSG
```

2.6. **A Summary of the Remote Boot Process**

Step 1: The network adapter of the client establishes the communication with the server. The server identifies the NIC twelve-digit hexadecimal ID, and looks in the workstations section of `RPL.MAP` to find the workstation type (and in particular to figure out if it is DOS or OS/2 requester, and which `.IMG` or `.FIT` file should be used).

Step 2: The server then looks at the server records section of `RPL.MAP` to find the matching line for the given workstation type. This fixes the boot block structure (a `CNF`-file).

Step 3: The boot block is sent to the client, and `RPLBOOT.SYS` takes over the control of the client booting.

Step 4: At the end of the boot block processing, `RPLBOOT.SYS` passes the control to the Operating System Loader (`RPLLOADR.COM` or `OS2LDR`) that fires up the corresponding operating system.

3. A Single Win95 RIPL Client

3.1. **The Simplest Configuration**

We are now going to present a simple procedure to set up Win95 diskless RIPL clients. This procedure consists of 8 important steps.

Step 1: Set up an OS/2 Warp Server machine, and enable RIPL support (for DOS and OS/2). This is a relatively complex procedure but it is a standard OS/2 Warp Server stuff. Some of the highlights are given in the previous section.

Step 2: Physically build a client workstation. As already mentioned, we need a network adapter that supports RIPL (I used 3Com EtherLink 905C-TX-M) and the system BIOS should support booting from networks. For the moment, install a hard disk on the client!

Step 3: Install Windows 95 on the client. Here we should do a normal install, without networking. For example, it is possible to use the very first upgrade version, 3.5 inch floppy disks. I have also played with Win95B, with very similar results ...

Step 4: On the server side, set up a DOS remote boot requester, which will boot from a special floppy disk image. The boot disk image should be created from a bootable Windows 95 floppy. Make sure the disk contains the following Win95 files

```
IO.SYS MSDOS.SYS COMMAND.COM IFSHLP.SYS HIMEM.SYS
```

as well as the following Aurora files:

```
DLSHELP.SYS NETWORK.INI NET.EXE NET.MSG
NETH.MSG CONNECT.EXE CONFIG.SYS AUTOEXEC.BAT
```

Add the line `DEVICE=IFSHLP.SYS` in the `CONFIG.SYS` file. Make sure that the system smoothly remote-boots to DOS. After some initial RIPL processing messages you should see the 'Starting Windows 95' message, following by processing of `CONFIG.SYS` and `AUTOEXEC.BAT` files. You should end up with a DOS prompt (pointing to the disk image [A:]) and the system should make the following network identifications

```
Z=\\SERVERNAME\IBMLAN$
```

and

```
Y=\\SERVERNAME\WRKFILES
```

Step 5: Choose a directory on the server (on a JFS, HPFS or HPFS386 partition) that will serve as the Windows 95 remote [C:] drive. Copy the entire client's Windows 95 tree from the client to this directory. Do the same with the client's `Program Files` directory. For example, you can use WinZip on the client and Info-ZIP unzip on the server. Make sure you copied all files, including read-only, system and hidden. On the server side, define and enable a sharing alias for the above mentioned directory (for example `\\SERVERNAME\W95`).

Step 6: Find and copy the program `SETMDIR.EXE`, from Windows 95 installation cabs to the remote-[C:] directory. This program allows us to manually set up Registry path for Windows 95, before GUI loads.

Step 7: Remote boot Windows 95 :) When the client boots to DOS95, go to `Z:\DOSLAN\NET` and execute

```
NET USE C: \\SERVERNAME\W95
```

This will disconnect the physical drive [C:] and reinterpret [C:] as the desired network drive. Now, from the new [C:] execute

```
SETMDIR /R:C:\WINDOWS
```

If everything is fine, we should get a confirmation that the Registry pointer has been set up to `C:\WINDOWS\SYSTEM.DAT`. However, there is a small but non-trivial probability that `SETMDIR` fails at this point, irreversibly hanging the workstation. In this case, we should work without `SETMDIR` using the method described in the next subsection.

Keep your fingers crossed. Repeat 6 times something appropriate that OS/2 will like for sure (remember, there exist exactly 6 Platonic polihedras in the 4-dimensional Euclidean space, in all higher dimensions there are only 3!). Then execute `WIN`.

Step 8: Check how your system works and do some manual adjustments. At this point (during the first boot) you could get an error message saying that the master boot record has been modified (check also in *Control Panel–System–Performance*) and that the drives use MS-DOS compatibility mode (after all, this is expected!). Your new system will not recognize/see VFAT long filenames, so you will need to change (from the server side) the long filenames to something standard (for example, change the files/directories including `Program Files` and screen saver filenames). Also, it will be necessary to edit Registry settings, to reflect the new file/directory names (this can be also done before copying the windows/program file trees). In such a way, you should get more or less the same Win95 system as initially, with a difference that some icons may not display properly, being converted to generic Windows icons. More about solving this interesting problem in Section 6.

Finally, you can physically disconnect the hard disk from the client workstation. It should boot a little faster in this case. Interestingly, it takes a very long time to shutdown or restart (the 'Please Wait While Your Computer Shuts Down' screen remains for about 6 minutes on my IBM PC350 machine).

Of course, the whole Win95 RIPL could be automated, putting the appropriate instructions into `AUTOEXEC.BAT`.

3.2. **Configurations with a RAM Drive [C:]**

Some of the benefits of introducing a RAM drive on a client workstation are:

- This drive can hold important frequently accessed files, including the swap file, temporary files etc, minimalizing the network load. Of course, putting the swap file on the RAM-drive implies that effectively there will be no virtual memory!
- We can put the system Registry on the RAM-drive (being copied from the server during the workstation startup). This means that all the changes on the Registry will not be saved, until we manually copy `SYSTEM.DAT` and `USER.DAT` back to their fixed locations on the server. By doing this, we increase the system integrity in several ways.
- It helps to solve some network disk-access problems for certain irregularly behaving files. It also plays an important role in setting up networking on RIPL clients.

It is necessary to perform a number of additional steps before a machine with a RAM-drive [C:] can be successfully RIPLed. To enable a RAM-drive, we have to copy `RAMDRIVE.SYS` to the (virtual) boot disk [A:] and load it during the boot process. For example, a `CONFIG.SYS` statement

```
DEVICEHIGH=A:\RAMDRIVE.SYS 16384 512 128 /E
```

would create a RAM-disk [C:] of 16MB, with 512 bytes sector size and 128 root directory entries (it is assumed that there is not any local hard disk on the workstation!).

Registry Recompile

The main problem here is how to change the host Windows drive letter from 'C' (as it was in the previous section) to another letter, say 'F'. The following simple procedure solves this:

(*i*) Export the whole Registry of the RIPL workstation to a text file, say `X.REG`.

(*ii*) Using a text editor, replace all the references to drive 'C' in `X.REG`, with the references to drive 'F'. Later on, we can change some references back to 'C' as a small portion of Windows will stay on the RAM-drive.

(*iii*) Re-build, using the DOS-mode regedit, the `SYSTEM.DAT` and `USER.DAT` files from this new `X.REG` file.

Interestingly, as a bonus, the recompiled Registry will be (probably considerably) smaller! Furthermore, if necessary change 'C' to 'F' in `SYSTEM.INI` and `WIN.INI`.

Setting Up MSDOS.SYS

We can replace the almost empty `MSDOS.SYS` on the boot disk with a 'real' Win95 `MSDOS.SYS`. Make sure that it contains the following line

```
WinBootDir=C:\
```

within the `[Paths]` section. Also, make sure that `LoadGUI=0` and `Logo=0` in `[Options]`.

Setting Up The Initialization Image of [C:]

Create a directory in `\\SERVERNAME\W95` to hold the initial files that will be copied to the workstation. Let it be say `C_INIT`.

Put the new Registry files `SYSTEM.DAT` and `USER.DAT` in this directory, together with `COMMAND.COM` and files you like to go to [C:] before Win95 loads. Ensure all hidden, system and read-only attributes are removed from `SYSTEM.DAT` and `USER.DAT` in `C_INIT`.

Preparing to Load Win95

Before loading the graphical interface (and after establishing a network identification `F=\\SERVERNAME\W95`) we should execute the following commands (putting them in the appropriate batch file/`AUTOEXEC`):

```
set path=C:\;F:\WINDOWS;F:\WINDOWS\COMMAND
copy F:\C\_INIT\*.* C:\
attrib +R +H +S C:\SYSTEM.DAT
attrib +R +H +S C:\USER.DAT
set comspec=C:\COMMAND.COM
```

Furthermore, re-allocate the swap file, by including the line

```
pagingfile=C:\WIN386.SWP
```

in the `[386Enh]` section of `SYSTEM.INI`.

Load Win95 GUI

Execute `WIN`. After the GUI comes up, you will probably have a couple of broken shortcut links. Manually change the program references from 'C' (left from the previous RIPL setup) to 'F'.

Note that we have not used at all the program `SETMDIR` to define the Registry location, in contrast to the solution described in the previous subsection. Instead, the Registry location has been defined in the `MSDOS.SYS` file.

4. Win98 RIPL Clients

Interestingly Win98 (without IE shell integration) RIPLes better than Win95 (after all, Win98 comes with many subtle improvements, updated drivers, and so on). In spite of this fact, Microsoft does not support remote-boot configurations of Win98 from any server.

Here the principal ideas are the same as for Win95 RIPL (that is, install first on a local hard drive, then copy to the server and perform the appropriate manual adjustments) however, there are some subtle differences.

First of all, it seems that `SETMDIR` utility (available with Win95 only!) does not work correctly at all with Win98. Hence, it is necessary to apply the above mentioned alternative method, by defining `WinBootDir` in the `[Paths]` section of `MSDOS.SYS`.

Secondly, Win95 disconnects automatically from its virtual boot floppy. This does not happen with Win98. Therefore, it is necessary to execute the `RPLTERM` utility (to disconnect from the virtual boot floppy disk, found in `Z:\DOSLAN\NET`) prior to loading the graphical part of Windows.

The standard configuration of Win98 (with IE integrated) generates much more network traffic during the boot process and various operations (as expected, because of all this IE extra load). Hence it is very important to re-configure IE stuff properly, replacing the long filenames Registry references (like 'Temporary Internet Files') with short counterparts.

In order to avoid a bunch of IE-generated troubles and speed up the client workstation, it is highly recommendable to remove the IE shell integration from the OS. This procedure will be discussed in more detail, from the RIPL viewpoint, in an extended version of this article.

We refer to `98LITE` website `http://www.98lite.net` for more informations on IE removal, various 'lite' configurations of Win98 and accompanying installation software.

Basically, the construction consists in replacing `EXPLORER.EXE` (in `WINDOWS` directory) as well as `SHELL32.DLL` and `COMDLG32.DLL` (in `WINDOWS\SYSTEM`) with their Win95 counterparts. In such a way we end up with a considerably faster, cleaner and smoother system, generating much less network load. Furthermore, it would be necessary to replace the NotePad and WordPad, with Win95 versions. In order to save the server disk space, we can completely remove the IE files and clean up the Registry, by removing unnecessary entries and recompiling (remember, workstations starting from their RAM drive [C:] transfer the Registry from the server).

The problem with a delayed shutdown and reboot *completely disappeared* when RIPLing Win98. My diskless workstations smoothly shut down and reboot, in all possible configurations (with/out IE integration, and with/out the RAM drive [C:]).

5. Problems and Solutions

5.1. **Problems With Icons**

It seems that Windows has problems to cache/access some icons properly, when RIPLed for the first time. The grade of this problem varies from case to case.

To solve the icons problem: At first, define the maximal icon cache by introducing a new string value *"Max Cached Icons"* (including the quotations!) and setting it to 2048. The value should be within

`HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Explorer`

Next remove (from DOS) the hidden `SHELLICONCACHE` file from the `WINDOWS` folder (in our case it will be truncated to `SHELLICO`).

If the above two steps do not cure the problem, manually edit the Registry and find icon references. They are mostly related to `SHELL32.DLL` file and icons are referenced, for example, as

$$\texttt{C:\WINDOWS\SYSTEM\SHELL32.DLL,}n$$

where $n \in \mathbb{N} \cup \{0\}$ is the icon number within the `SHELL32` library. In many cases, it is the *DefaultIcon* key. Make sure the icons are NOT referenced as

$$\texttt{C:\WINDOWS\SYSTEM\SHELL32.DLL,-}m$$

If yes, change the referencing model to the first format. Also, if you have a problem with the icons stored in `EXPLORER.EXE` change the Registry references to the similar icons in `SHELL32.DLL`. For example, *My Computer* icon originally comes from `EXPLORER.EXE`. This particular icon is referenced within

`HKEY_CLASSES_ROOT\CLSID\{20D04FE0-3AEA-1069-A2D8-08002B30309D}`

and you can change it to any other icon.

After all these adjustments, delete again (from DOS) the `SHELLICO` file. If the icons are still not showing up properly, force the system to re-read the icons info by changing the icon size by few points and then going back (*Desktop–Properties–Appearance–Item=Icon*).

It is interesting to mention that the RAM-drive setup allows us to avoid changing the file reference (from `EXPLORER.EXE` to `SHELL32.DLL`) for *My Computer* icon (necessarily changing the standard icon). However, in order to avoid *My Computer* icon being replaced with a generic Windows icon, it is necessary to include `EXPLORER.EXE` in `C\_INIT` in order to *copy* it to C before GUI loads, and change the icon path in Registry. We can also re-define the shell pointer in `SYSTEM.INI` accordingly.

5.2. **32-bit Redirection and Long Filenames**

As we have already observed, Windows RIPL workstations normally would not recognize long filenames. The explanation of this phenomenon is simple. We have used a real-mode DOS-compatible software to map the system drives, and DOS does not know anything about long filenames. Therefore, the filesystem is DOS-like, from the very beginning. The connections stay active all the time, including after the full GUI comes up. So Windows will see the same 8.3-picture.

It is worth noticing that some of the programs 'demanding' long file names can be made very friendly by simple binary edit of the executable/dlls and/or renaming the offending files.

In order to get long filenames, we should use a redirection software supporting long filenames, working within the 32-bit Windows shell. There exist two natural ways to make this work:

(*i*) Dynamically switch from DOS to Windows (16-bit to 32-bit) mode with the help of the appropriate `VXD`-driver.

(*ii*) Install the Client for Microsoft Networks.

The first method requires the use of Microsoft networking in the DOS boot phase. The second approach is much simpler. The basic idea is to keep the initial DOS-mode connection for the minimal Windows with networking, and create new network drives using the 32-bit redirection (`NET.EXE` from a DOS window). Of course, in order to do this we should install the Client for Microsoft Networks (this should be done from the RIPL workstation, once it is up and running).

It is important to mention that this method works for RIPL workstations *using both* Microsoft and IBM (coming with Aurora) 16-bit redirection software.

5.3. **Performance/Stability Issues**

The Performance Tab of System Properties shows (besides already mentioned boot sector change and real-mode disk access notifications) that the virtual memory is using the DOS-compatibility mode. This is actually not a problem, it appears to be normal for RIPL workstations (the same happens with workstations RIPLed from NT-server [5]).

Note that the system initially boots to its 'virtual disk' [A:] represented by the above mentioned floppy disk image. However, after the GUI loads, Windows will forget about it, and point to its standard floppy drive (assuming the workstation has a floppy drive!). On the other hand, this does not happen when Windows is RIPLed in a safe mode. In other words, we have assumed here that the standard `RPLTERM.EXE` utility has not been used at all! It does not hurt to execute `RPLTERM` before `WIN`, and in this way Windows will boot properly in the safe mode too.

Sometimes the client freezes when an application is launched from a DOS window. It seems that this does not happen when the same programs are launched from the *Run* window or Windows Explorer.

6. Working With Many Clients

Once we have a single-client solution, it is easy to set up multiple clients. We can take advantage of the OS/2 Warp Server filtering mechanism we mentioned in Section 2.

For each client, we should define its *machine directory* where all client-specific files should be stored (such as the system Registry, the appropriate `.INI` files, working files, caches, and preferences of various programs).

For example, we can simply use a working area `MACHINENAME` created by OS/2 Warp Server for every RIPL requester. This is a subdirectory of

```
\\SERVERNAME\WRKFILES=IBMLAN\RPLUSER
```

For Win95/98 clients possessing a RAM-drive configuration, it would be necessary to copy the appropriate Windows files from the machine directory to the RAM-drive, during the DOS boot phase. For example, the appropriate `AUTOEXEC.BAT` instructions could look like

```
copy Y:\~~~~~~~2\WIN\*.* C:\
attrib +h +r +s C:\USER.DAT
attrib +h +r +s C:\SYSTEM.DAT
```

where it is assumed that Registry files are among the ones that are copied to [C:]. The client would see the first line as

```
copy Y:\MACHINENAME\WIN\*.* C:\
```

For Win95 clients without a RAM-drive, we can set the registry pointer by `SETMDIR`. Alternatively, we can define `WinBootDir` in `MSDOS.SYS`:

```
WinBootDir=Y:\~~~~~~~2\WIN
```

This works for both Win95 and Win98, however it is assumed that the 16-bit redirections are made using the IBM redirection software (Microsoft's `NET.EXE` would not work because it *requires* the Registry from `WinBootDir`, as discussed in Appendix A).

For simplicity, we have assumed here that the client has the full read/write access to all files from its machine directory.

7. Concluding Observations and Remarks

Sometimes the Windows fails to load, after installing the networking. I have found that loading `EMM386.EXE` from `CONFIG.SYS` would always solve the problem. There are many possible parameters, but the following basic variation must be present:

```
DEVICE=A:\EMM386.EXE /Y=C:\WINDOWS\EMM386.EXE
```

We have assumed here that our workstation starts from a RAM-drive [C:]. The switch `/Y` tells Windows where to find the copy of `EMM386.EXE` to be used during GUI initialization (the copy from [A:] would have obviously not worked, if we have executed `RPLTERM` prior to `WIN`).

From the practical viewpoint, the most interesting topic is to see which Windows applications can be installed within the RIPL environment. In principle, every normal application should be RIPL-deployable (perhaps after some non-essential modifications).

As an illustration, let us discuss what should be done to make Mathematica V3/4 RIPL-friendly. Mathematica is a highly professional software package designed for almost all kinds of symbolic calculations in modern mathematics.

The first problem is that Mathematica makes an extensive use of long filenames. Thus we should access Mathematica using the 32-bit networking. To install the software on the OS/2 Warp Server, it is sufficient to copy Mathematica tree from an appropriate Windows machine (with a HD).

The second problem is that Mathematica will immediately try, upon execution, to write something in `Progam Files` directory, requiring a lot of long subdirectories/filenames. Normally, this will fail because the principal Windows folders are under the 8.3-filesystem. A simple solution here is to introduce a special undocumented environment variable `MATHEMATICA_PREFERENCES` and assign to it the appropriate network drive/directory name. For example, we can put

```
SET MATHEMATICA_PREFERENCES=X:\~~~~~~~2
```

in the client `AUTOEXEC.BAT` file. Here [X:] is a network drive realized using the 32-bit redirection and `RIPLCLIENT` is the ID of our RIPL workstation (we could point to the standard *working area* `WRKFILES`).

Finally, we should change Mathematica kernel parameters (to reflect the new location). In such a way we can share one copy of Mathematica among many users.

Interestingly, the 'confusing Windows' effect discussed in Appendix A (not to recognize if its redirection software has been loaded) can be achieved by executing `SNAPSHOT`, or even `SNAPSHOT /M:0` just before `WIN` (and *after* making network connections). Of course, this is not at all compatible with configurations loading dynamical transition drivers (such as `SNAPSHOT.VXD`).

Appendix A. A Microsoft-only Networking Solution

In this Appendix we shall explain how to uniformize the client architecture, using only Microsoft redirection software (both 16-bit and 32-bit). In particular, it will be explained how to create a mini-registry file, suitable for DOS-mode networking.

At first, we shall consider workstations starting from a local floppy disk, and explain how to set up the DOS-mode Microsoft networking. The most subtle part of this procedure consists in creating a mini-registry file which contains the appropriate network information. As we shall see, there is a funny way to trick the Setup program, to do the main job for us.

Next, we consider 'true RIPL' workstations, booting from a virtual floppy disk image on the OS/2 Warp server. This requires a small modification of the 'real boot floppy' configuration.

All considerations are applicable to both Win95 and Win98 RIPL configurations, except the method of tricking the Setup, which works only with Win95. The installation Setup of Win98 is designed from the ground up NOT to recognize remote-boot installations, as the whole Win98 is officially non-RIPLable at all!?!

A.1. **Workstations Starting From a Local Floppy**

Our main objective here is to make a bootable floppy disk, suitable for RIPL, containing the purely Microsoft redirection software. In other words, this software should be used to connect to the server and map network drives during the DOS boot phase.

The Microsoft version of `NET.EXE` *requires a registry file* to read the corresponding network information. Consequently, it is necessary to include a Registry file on the boot floppy. Having in mind that `NET.EXE` actually needs only a *very small* part of the real Registry, we may create a special *mini-registry* file, just for this purpose. Of course, we could try to put the full system Registry on the boot floppy, but it is often too big to fit on a single floppy disk (together with other critical files). In order to create the registry file, we can use the Win95 installation Setup.

(*i*) Copy the files (from Win95 CD for example)

```
DOSSETUP.BIN    SETUP.EXE    EXTRACT.EXE
WINSETUP.BIN  PRECOPY1.CAB   DELTEMP.COM
SUHELPER.BIN  PRECOPY2.CAB  SMARTDRV.EXE
OEMSETUP.BIN    MINI.CAB    SCANDISK.EXE
```

into an appropriate directory on the server. These files are necessary for the server-based Setup of Win95 to perform the client-specific operations.

(*ii*) Set up a diskless RIPL DOS-requester (using IBM redirection software provided with Aurora) possessing a real floppy drive. More precisely, the machine should run Win95 in DOS mode. The client should also have a RAM-drive [C:] of say at least 8MB.

(*iii*) Enable the client access to the above mentioned Setup directory.

(*iv*) From the client side, connect to the OS/2 Warp server, make the appropriate network drive identifications, then `RPLTERM` (to enable the real floppy drive) and execute

```
SETUP /T:TempFilesPath
```

This will initiate the Windows installation Setup.

(v) Go through the 'Installation Process' and choose minimal components without any special hardware selection (this is just to make the Setup happy, otherwise it could easily crash on any nonstandard point, as discussed in [2]).

If everything goes fine, at the end of the procedure you will be asked to insert a blank floppy in the client floppy disk drive. At this point, if you decide to proceed, you should make sure that a full Win95 installation media is available to the client. The setup will create a bootable floppy, containing Microsoft network software, other critical files, and–most important for us–a raw mini-registry `SYSTEM.DAT`!!!

Better yet, we can exit the Setup just before the step of creating the boot floppy (at the 'Start Copying Files' screen). The mini-registry files will be in `C:\WINDOWS` directory (under names `SYSTEM.NEW` and `USER.NEW`, note that these files will marked as system, hidden and read-only). Using this, we can easily manually make the boot floppy. Before exiting the Setup, it is not a bad idea to make a backup copy of the *temporary files* folder. It contains various valuable files that could be useful in other experiments ...

An interesting variation of this theme is to use workstations *with a hard disk* which will hold all client-related files (later on, we can easily move these files to a client location on the server). Client files will go to the `C:\WINDOWS` directory, and a copy of the boot floppy files would be saved in the `SUBOOT` subdirectory.

For this to work correctly, we should compose a `MSBATCH.INF` file containing instructions `HDBoot=0`, `RPLSetup=0`, `WorkstationSetup=1` and `SaveSuBOOT=1` in the `[Network]` section. The last instruction ensures that the boot floppy files are saved in the `SUBOOT` subdirectory of the main machine directory (`C:\WINDOWS` in our case).

In principle, we could have defined `InstallDir` (this `[Setup]` variable points to the machine-specific directory) as a network directory, however such configurations have problems with OS/2 Warp Server: in all my experiments the setup was crashing! A sample `MSBATCH.INF` is given in [2] and a detailed documentation on its structure can be found in [1]-Appendix D. The correct setup call remains the same

```
SETUP /T:TempFilesPath
```

We should make several subtle adjustments to the boot floppy. At first, we should include the appropriate NIC-driver. Secondly, we should manually edit the `PROTOCOL.INI` file and introduce the appropriate lines in the files `CONFIG.SYS`, `AUTOEXEC.BAT` and `MSDOS.SYS`.

The raw mini-registry should be further edited (in order to be usable with your NIC-driver and `NET.EXE`). In order to do this, export it to a text file,

```
REGEDIT /L:system.dat-location /R:user.dat-location /E minireg.reg
```

then edit the corresponding network sections, and then compile to create the new mini-registry:

```
REGEDIT /L:new-system.dat /R:new-user.dat /C minireg.reg
```

Alternatively, especially if you prefer simplicity over complexity, you can use a *blueprint mini-registry* file listed in Appendix B (are you sure you *really* do not want to play with the Windows installation Setup?).

After creating the mini-registry, you should carefully adjust the boot floppy. Make sure it contains the following files:

SYSTEM.DAT	PROTMAN.EXE	NDISHLP.SYS	SMARTDRV.EXE
USER.DAT	PROTMAN.DOS	IFSHLP.SYS	RAMDRIVE.SYS
CONFIG.SYS	NET.EXE	EMM386.EXE	HIMEM.SYS
PROTOCOL.INI	NET.MSG	MSDOS.SYS	IO.SYS
AUTOEXEC.BAT	NETH.MSG	SNAPSHOT.EXE	COMMAND.COM

Furthermore, make sure that `CONFIG.SYS` contains the line

```
DEVICE=PROTMAN.DOS /I:A:\
```

where the parameter `/I:` points to the location of `PROTOCOL.INI`. It is crucial that `PROTOCOL.INI` contains the correct information.

The program `SMARTDRV.EXE` greatly reduces the floppy disk activity, and the time necessary to perform necessary DOS-mode tasks.

An interesting technical problem here is that Windows figures out if its own redirection software is loaded prior to the graphical interface (`NET.EXE` works 16-bit in DOS, and 32-bit in Windows) and then *refuses to load* :(This is where the program `SNAPSHOT.EXE` enters the game. It tricks Windows *not to recognize* when its 16-bit networking is running. Normally, this should be executed before using any redirections:

```
SNAPSHOT /M:XYZ
```

where `XYZ` is the amount of memory in kilobytes that should be reserved for the real mode (=DOS) networking.

The second role of `SNAPSHOT.EXE` is to prepare the system to a *dynamical transition* from 16-bit to 32-bit networking, that occurs during the GUI initialization. This transition is achieved by loading a `VXD`-version of `SNAPSHOT` from the main Registry (see [1]-Chapter 4). Both `SNAPSHOT.EXE` and `SNAPSHOT.VXD` can be found in Windows installation cabs. In the present article we shall not consider configurations involving `SNAPSHOT.VXD`.

A.2. **Workstations Starting from a Virtual Disk Image**

These RIPL configurations are obtained by a simple modification of the previous setup. There is one subtle point here: a possible conflict between the network software on the (virtual) floppy and the software loaded during the boot-block execution phase (loaded from the corresponding `CNF`-file). Perhaps the simplest solution here is to:

(*i*) Create a RAM-drive [C:]. Copy `NET.EXE`, `NETH.MSG`, `NET.MSG`, `SYSTEM.DAT`, `PROTMAN.EXE`, `RPLTERM.EXE`, `COMMAND.COM`, `SNAPSHOT.EXE` and `SMARTDRV.EXE` to it (from `AUTOEXEC.BAT` for example). Re-set the variable `COMSPEC` to `C:\COMMAND.COM`.

(*ii*) Execute `RPLTERM` from [C:] and then connect to the server using `NET.EXE`.

Appendix B. A Mini-Registry Sample

In this Appendix we provide a 'blueprint' mini-registry file, suitable for editing and compiling.

Before compiling, the text of the form '`AURORA:{X}`' should be edited to reflect your workstation configuration. More precisely, such expressions should be *replaced* by the appropriate values of `X`. This registry file is intended for workstations having a RAM-drive [C:] with the Windows boot directory `C:\WINDOWS`. Some long lines are split in this listing.

```
REGEDIT4

[HKEY_LOCAL_MACHINE]

[HKEY_LOCAL_MACHINE\Software]

[HKEY_LOCAL_MACHINE\Software\Microsoft]

[HKEY_LOCAL_MACHINE\Software\Microsoft\Windows]

[HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion]
"SystemRoot"="AURORA:{WinDirPath}"

[HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Setup]
"detectedlastdrive"="Z"
"BootDir"="C:\\WINDOWS"
"BootHost"="C:\\WINDOWS"
"OldWinBootDir"="C:\\WINDOWS\\WINBOOT"
"WinDir"="C:\\WINDOWS"
"AppsDir"="AURORA:{\\\\shared-win-UNCpath\\}"
"WinbootDir"="C:\\WINDOWS"
"HostWinBootDir"="C:\\WINDOWS"
"MachineDir"="C:\\WINDOWS"
"WinAdminDir"="C:\\WINDOWS"
"SharedDir"="AURORA:{\\\\shared-win-UNCpath\\}"
"SysDir"="AURORA:{\\\\shared-win-UNCpath\\}"
"SourcePath"="AURORA:{\\\\shared-win-UNCpath\\}"
"SourcePathType"=hex:05,00,00,00
"lastdrive"="32"

[HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion
                                        \Setup\WinbootDir]
"devdir"="C:\\WINDOWS"

[HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion
                                        \Network]

[HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion
                                        \Network\Real Mode Net]
"transport"="*netbeui,*nwlink,ndishlp.sys"
"netcard"="AURORA:{NetCard-Driver}"
"LoadRMDrivers"=hex:01,00,00,00
"preferredredir"="VREDIR"
"Transition"=hex:01
"SetupN"=hex:01
"SetupNPath"="AURORA:{\\\\shared-win-UNCpath\\}"
"StaticDrive"="AURORA:{WinDrive},C"

[HKEY_LOCAL_MACHINE\System]
```

```
[HKEY_LOCAL_MACHINE\System\CurrentControlSet]

[HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services]

[HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\VxD]

[HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\VxD\VNETSUP]
"ComputerName"="AURORA:{ComputerName}"
"Workgroup"="AURORA:{WorkGroup}"
"Comment"="AURORA:{A Nice Comment}"
"MaintainServerList"="2"
"LMAnnounce"="0"
"StaticVxD"="vnetsup.vxd"
"Start"=hex:00
"NetClean"=hex:01

[HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control]

[HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\ComputerName]

[HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\ComputerName
                                                    \ComputerName]
"ComputerName"="AURORA:{ComputerName}"

[HKEY_LOCAL_MACHINE\Enum]

[HKEY_USERS]

[HKEY_USERS\.Default]
```

References

[1] *Microsoft Windows 95 Resource Kit*, Microsoft Professional Editions (1995)

[2] Joe R Doupnik: *Installing Win95 on Netware Servers (read-only) for Diskless Clients*, Utah State University, June (1996).

[3] *WorkSpace On-Demand Administrator Guide*, IBM Corporation (1998)

[4] Khoa Huynh, Boyd Jasperson, Rich Wahl: *WorkSpace On-Demand 2.0 Feature for Windows Clients Performance White Paper*, IBM Corporation (1998)

[5] *Windows NT Server Documentation*, Microsoft Corporation (1996)

Instituto de Matematicas, UNAM, Area de la Investigacion Cientifica, Circuito Exterior, Ciudad Universitaria, México DF, cp 04510, MEXICO

E-mail address: micho@matem.unam.mx

http://www.matem.unam.mx/~micho