BY GUY C. YOST

Streamlining Management of WinFrame Servers: Part I Exploring the Techniques

This article, the first in a three-part series, explores various techniques to streamline management and operation of large-scale WinFrame implementations, some of which can be applied to NT environments in general.

AS the popularity of WinFrame increases and the cost of server hardware decreases, many organizations are taking on large-scale deployment of WinFrame Server farms that host hundreds of user connections across multiple servers.

As with other network environments, there are ways to design a system to favor either management efforts or performance — the classic trade-off that IT administrators have faced for more than 15 years. For example, before the days of switched networks and inexpensive 100MBps Ethernet, it was not uncommon to install client applications to the workstation's local drive rather than using a centralized copy on a file server. Although this practice improved application performance and alleviated traffic on over-congested network segments, the laborious solution also increased the cost of ownership, as administrators would continually hop from desktop to desktop to keep the "network" running.

Fortunately, the days of feeling sorry for a network cable because lots of 1s and 0s are running over it are gone, and administrators are getting back to designs that leverage the foremost purpose of a network: to share resources. However, the reason I mention this particular example is because I've seen a lot of WinFrame implementations that follow a "cookie-cutter," out-of-the-book design that doesn't favor the administrator or management efforts.

If you know what you're doing, you don't have to follow stock rules to the letter. By bending and tweaking the technology a bit, you can design low maintenance systems that will save everyone money as well as their sanity. However, always be sure that you clearly document your innovations.

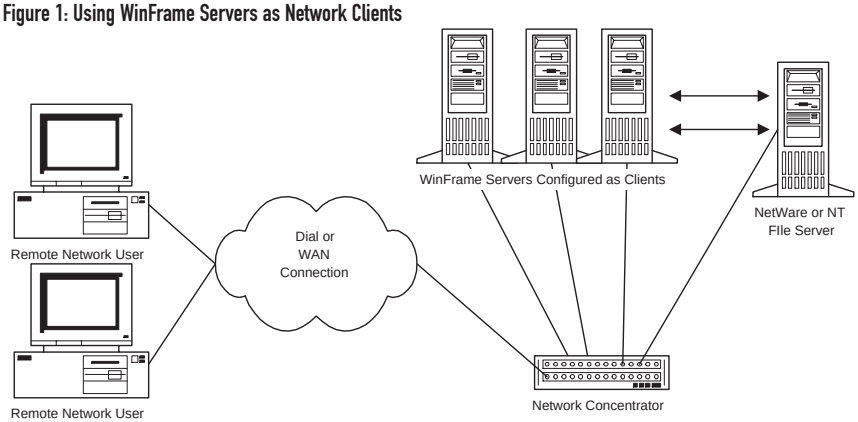
This article, the first in a three-part series, explores various techniques I've used to streamline management and operation of large-scale WinFrame implementations, some of which can be applied to NT environments in general. Parts II and III will examine additional techniques used to automate routine processes and WinFrame server creation/disaster recovery.

WINFRAME: CLIENT OR SERVER?

Remember that a WinFrame "server" is actually acting as a multi-user network client rather than a server, even though the industry categorizes WinFrame technology as a "server." That is, the network client is the remote ICA client along with its corresponding WF server session. It is important to understand this distinction when designing your WinFrame network because design strategies for servers and clients are very different (i.e., administrators try to centralize resources on servers while making distributed client configurations as generic and easily supportable as possible).

Given the client nature of WinFrame servers, you want to configure them as you would other clients. For example, if remote WinFrame users need to access a central database and you have 10 WinFrame servers that support 300 users, would it be better to replicate the database across every WinFrame server or have the user sessions attach to a centralized shared copy of the database? The answer may seem obvious; however, when dealing with applications, stock WinFrame designs lose site of this rule. For the same reason you wouldn't replicate the binaries for your networked applications across every desktop, you wouldn't want to support multiple installations of user

Figure 1: Using WinFrame Servers as Network Clients



applications on every WinFrame server. The exception may be small installations where one or two servers are in production and there is no anticipation of expansion.

Working with environments that host large numbers of users and consequently large numbers of WinFrame servers, I've found that centralizing the user application on a file server greatly simplifies application update and upgrade efforts (which seem to be ongoing). When a new DLL or EXE file is needed to patch or upgrade an application, you simply replace the single centralized instance and all users benefit. Only the "network client" portion of the application needs to be installed on the WinFrame server in order to get the proper registry entries to support 32-bit applications, and that exercise occurs much less frequently than updating specific DLL or EXE files.

So, as we progress through new environments, we find reasons why old rules prevail. Figure 1 shows the conceptual layout of using a central file server in a WinFrame environment as you would in a standard client/server network.

As noted, the central file server can be NetWare, NT, or other. In the next section, the options for connecting the NT-centric WinFrame server farm to a NetWare (or other non-NT based NOS) are explored.

GATEWAY WHICH WAY?

The idea of a gateway has evolved somewhat over the years. Traditionally, it referred to a device that simply allowed communication between two distinct network types, whether the difference be in the network protocol, NOS or OS, or transport architecture.

Today, that meaning hasn't been lost but there is an implied "one-to-many" functionality. For example, popular IPX-to-IP gateways provide a single IP address interface for many IPX clients. Internal IP-to-external IP proxies and Network Address Translation devices serve a similar purpose. Even the NT-to-NetWare gateway uses a single NetWare user ID to allow several NT users access to NetWare resources.

If you know what you're doing, you don't have to follow stock rules to the letter. By bending and tweaking the technology a bit, you can design low maintenance systems that will save everyone money as well as their sanity.

Being faced with the task of connecting a 2,000-user WinFrame environment to a NetWare NDS environment, I closely examined my gateway options. There were two choices: Either "map" one NetWare user to 2,000 NT user IDs or map one NT ID to 2,000 NetWare IDs. Given the trials of NT domains with large networks and the fact that the organization was already versed in NDS management, my direction became clear; however, there were obstacles. The only gateway product supported by Citrix adhered to the former option: One NetWare user account is used to allow multiple NT IDs into the network. This is accomplished using the NWGS service that comes with NT/WinFrame. Wanting to go the opposite direction, using the IntranetWare Client for NT on the WinFrame servers was ideal. That way, remote users would be presented with an NDS login, and user accounts would not need to be duplicated or translated in the NT domain. A remote user would still need to authenticate to the WinFrame/NT domain; however, as either anonymous or an explicit (generic) user. We configured Win-Frame's auto-login feature to transparently force every connection made to login to the NT domain as a predefined (single) user. This allowed the first authentication challenge seen by the users to be an NDS login.

After testing and researching that approach, we found that we could get the majority of the desired functionality. However, some design shortcomings were discovered on the part of the IntranetWare Client — mainly stemming from the fact that the client was designed for single-user access and not for numerous users connecting to the same machine simultaneously. We didn't accept defeat and pursued ways to overcome the shortcomings. We were successful by incorporating the following "tweaks":

- Because the IntranetWare 4.11a client is designed for single use, it uses WinFrame's MSGINA.DLL (which presents the initial login control and interface) instead of Novell's NWGINA.DLL. This forced us to use profiles as defined by the NT domain rather than the NDS user. Because we were using only one generic NT user account (via the WinFrame auto-login feature), this limited us to having one profile for multiple users. As special needs for special users became inevitable, we quickly integrated Novell's Application Launcher (NAL) into the system. After the NDS user name and password is accepted, normal login script logic and application processing resumes. NAL uses the NDS ID and associated groups/OUs to determine what additional applications users have access to (in addition to the stock profile for the NT auto-login user). NAL is launched in the login script only for members of the NAL group and stock users are simply presented with the generic desktop as defined by the profile for the auto-login user.

- The 4.11a client records the last NDS login context in the HKEY_LOCAL_MACHINE registry that is global to all users that access the WinFrame server. This resulted in the default current login context being set by the subsequent

Figure 2: Locking the NDS Context in the NT Registry

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\NetWareWorkstation\
Parameters\Trees] = "OUR_ENTERPRISE_NETWORK"="CONTEXT.ORG"
```

Figure 3: Distributing Files Across Servers Using Batch Files

```
DIST D \WINFRAME\SYSTEM32\MUMBLE.DLL

DIST.BAT

for %%C in (01 02 03 04 05 06 07 08 09 10 11 12) do call WFCPY.bat %%C %1 %2

WFCOPY.BAT

net use k: \\wfserver%1\%2$
copy %2:%3 k:%3
net use k: /delete
```

login. Because this system had 2,000 users who belonged to more than 300 NDS OUs, we wanted to ensure that the same (root) context was consistently displayed for all users so they wouldn't have to fully qualify their lengthy NDS login name. To "freeze" the default context simply use the REGEDT32 utility to assign read-only permissions for all users (including SYSTEM, EVERYONE and ADMIN) to the key shown in Figure 2.

In this example, OUR_ENTERPRISE_NETWORK is the NDS tree name and CONTEXT.ORG corresponds to the default OU.O context you want to enforce for all logins.

Addressing these two known issues allowed us to maintain the original "reverse" gateway concept even though neither Novell nor Citrix officially supported it.

TEMP DIRECTORY MANAGEMENT

WinFrame hosts 32-bit Windows applications, which along with the NT OS, will generate temporary files, just as on local NT workstations. Effective throughput of an ICA session relies on minimizing unnecessary data transfer over the connection. If the TEMP directory for a session is defined on the client local drive, then the entire contents of the TEMP directory will have to traverse the network, which directly impacts mouse speed and keyboard responsiveness.

Because an ICA session resides on the WF server, it makes sense to have the TEMP files generated by a session reside on the hosting server rather than traverse the network. WinFrame is designed to support this and will automatically clean up TEMP directories created on the server after a session is terminated.

By default, users connecting to a WinFrame server are automatically assigned a unique temporary directory that is created from the system's TEMP directory. For example, if the system's TEMP variable is set to TEMP=E:\TEMP, then the first user on the system may receive E:\TEMP\1B as that session's TEMP directory while the second user gets E:\TEMP\2C as their TEMP directory. To ensure this functionality, as administrator, issue the "FLATTEMP /DISABLE" command at the WinFrame DOS prompt. Because the ultimate TEMP directory name assigned is random and generated at the time of login, it is difficult for administrators to predict exactly where application-specific temp files will be stored. This presented a problem configuring several applications that required a hard-coded path setting for the TEMP directory. To overcome this obstacle, we ran a batch file upon exit of the login script that used the SUBST command to assign a virtual T: drive to the currently defined %TEMP% variable and simply used a generic "T:\" as the TEMP parameter in application configurations. The simple command was:

```
SUBST T: %TEMP%
```

Because the command runs after the login script and before the desktop is presented, all applications with T:\ specified as their temporary directory reliably use whatever TEMP (E:\TEMP\XX) directory that was assigned to the session.

UNAVOIDABLE FILE DISTRIBUTION

There's no doubt that centralized applications and data (as referred to in Figure 1) will greatly reduce the amount of effort required to maintain large WinFrame installations. However, there will always be an ongoing need to distribute files across WinFrame servers, such as OS-specific DLLs, executables, and IP HOST tables. We researched and tested several third-party packages whose sole function is to "synchronize" the contents of one "source" server to other "target" servers. Even though some of the packages we tested performed as advertised, we opted not to incorporate them due to our "generic" configuration of the WinFrame servers and the cost, configuration, administration, and indiscriminant network overhead associated with the software. In a different setting, where applications and data were not centralized, stronger consideration for using a synchronization product would have been warranted. Instead, I wrote a quick little batch process that allows us to replicate select files across servers when needed.

The name of the main batch file that calls batch file WFCOPY.BAT is DIST.BAT. The reason for the two batch file sequence is to simulate a typical programming language FOR loop with variables. For example, the command shown in Figure 2 will distribute the file "MUMBLE.DLL" on the current server into every other server's D:\WINFRAME\SYSTEM32 directory. The DIST.BAT file takes two parameters: the first being the drive letter on both the source and destination where the file is located, and the second being the full path and file name of the file to be distributed.

A key to the routine presented in Figure 3 is that the WinFrame servers are named sequentially as WFSERVER01, WFSERVER02, WFSERVER03, etc. Running this as administrator allows connecting to the administrative share that is inherent to all servers (\\WFSERVER01\D$) using a temporary K: drive and then copying the exact path from the source server to the destination servers.

SUMMARY

Recent projects have proven to my clients and me that a high level of stability and a surprisingly low level of maintenance can be achieved using strategies presented in this series. I'm not tooting my horn, but rather I want you to have confidence in your instincts when I tell you that the principles introduced have been proven on the largest Citrix and PeopleSoft installation in the world. I won't tell you it was easy on

my nerves to insist stock design rules be ignored and implement the "unsupported" path, but I was very fortunate to have the faith and support of a fabulous management team behind me. The result is a system that almost takes care of itself as a result of using available technology in a way that everyone told us could not be done.

Part II will explore various routine server-task-automation techniques and rapid WinFrame server creation for deployment, maintenance, and disaster recovery purposes. ts

NaSPA member Guy C. Yost is the owner of Redstone Consulting, an IT management consulting firm in New York. He has authored several books on networking for Que Publishing, including Learning NetWare 4.1, *and* NetWare 4.1 SmartStart, *and contributes to* Technical Support *magazine as an author, columnist and technical editor. Guy also develops and conducts seminars on networking with Windows NT, UNIX, NetWare and Internet/intranet technologies across the United States and Canada. He can be reached at (518) 674-5606 or gyost@logical.net.*