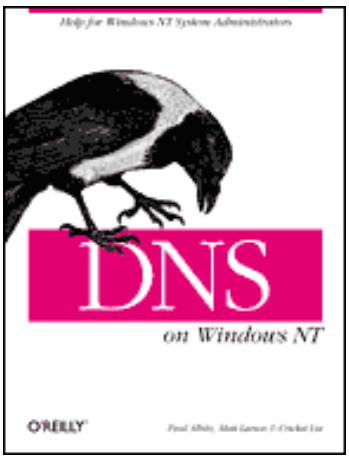
DNS on Windows NT

Sample Chapters

Sample Chapters 4 & 10 of *DNS on Windows NT*:



DNS on Windows NT

By Paul Albitz, Matt Larson & Cricket Liu
1st Edition October 1998
1-56592-511-4, Order Number: 5114
348 pages, $34.95

Sample Chapter 4: Setting Up the Microsoft DNS Server

In this chapter:

"It seems very pretty," she said when she had finished it, "but it's rather hard to understand!" (You see she didn't like to confess, even to herself, that she couldn't make it out at all.) "Somehow it seems to fill my head with ideas --only I don't exactly know what they are!"

If you have been diligently reading each chapter of this book, you're probably anxious to get a name server running. This chapter is for you. Let's set up a couple of name servers. Some of you may have read the table of contents and skipped directly to this chapter. (Shame on you!) If you are one of those people who cuts corners, be aware that we may use concepts from earlier chapters and expect you to understand them.

Several factors influence how you should set up your name servers. The biggest factor is what sort of access you have to the Internet: complete access (that is, you can *ftp* to *ftp.uu.net*), limited access (limited by a security firewall), or no access at all. This chapter assumes you have complete access. We'll discuss the other cases in Chapter 13, *Miscellaneous*.

In this chapter, we'll set up two name servers for a fictitious domain, as an example for you to follow in setting up your own domain. We'll cover the topics in this chapter in enough detail to get your first two name servers running. Subsequent chapters will fill in the holes and go into greater depth. If you already have your name servers running, skim through this chapter to familiarize yourself with the terms we use or just to verify that you didn't miss something when you set up your servers.

Our Domain

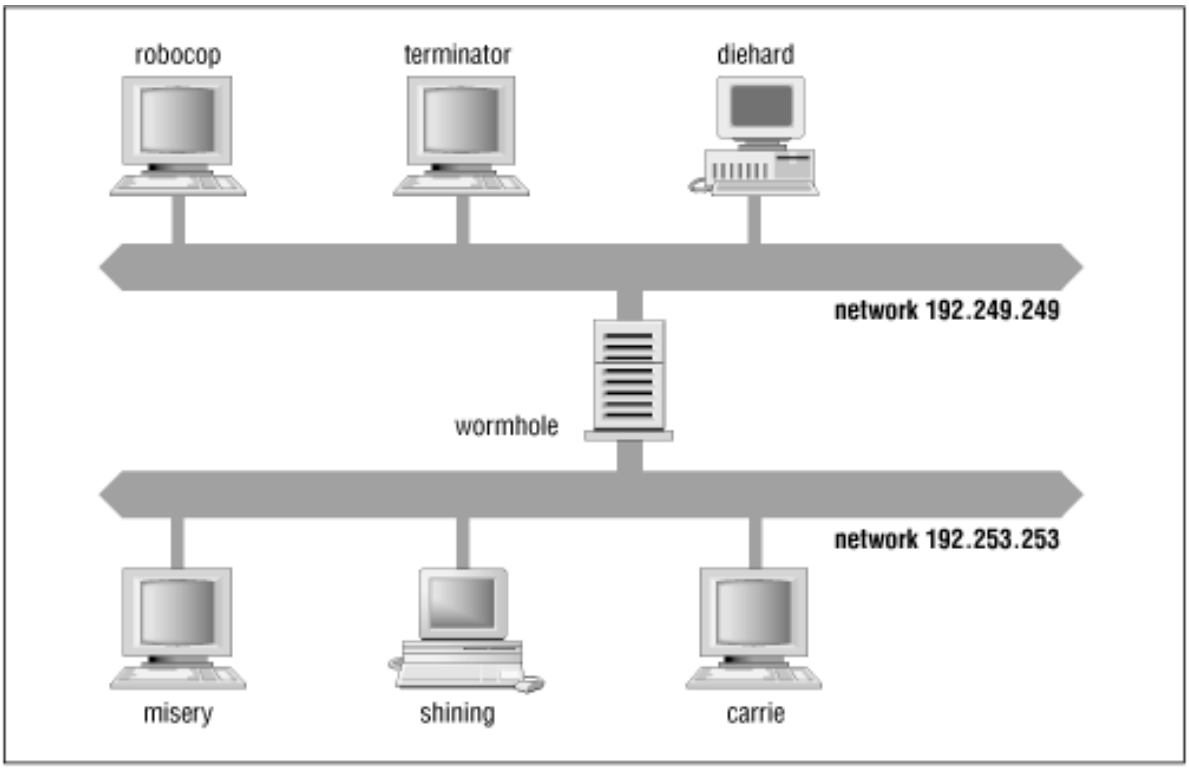
Our fictitious domain is for a college. Movie University studies all aspects of the film industry and researches novel ways to distribute films. One of the most promising projects is research into using Ethernet as the distribution medium. After talking with the folks at the InterNIC, they have decided on the domain name *movie.edu*. A recent grant has enabled them to connect to the Internet.

Movie U. currently has two Ethernets, and they have plans for another network or two. The Ethernets have network numbers 192.249.249.0 and 192.253.253.0. A portion of Movie U.'s host table shows the following entries:

```
127.0.0.1       localhost

# These are our killer machines

192.249.249.2  robocop.movie.edu robocop
192.249.249.3  terminator.movie.edu terminator bigt
192.249.249.4  diehard.movie.edu diehard dh

# These machines are in horror(ible) shape and will be replaced
# soon.

192.253.253.2  misery.movie.edu misery
192.253.253.3  shining.movie.edu shining
192.253.253.4  carrie.movie.edu carrie

# A wormhole is a fictitious phenomenon that instantly transports
# space travelers over long distances and is not known to be
# stable.  The only difference between wormholes and routers is
# that routers don't transport packets as instantly--especially
# ours.

192.249.249.1  wormhole.movie.edu wormhole wh wh249
192.253.253.1  wormhole.movie.edu wormhole wh wh253
```

And the network is pictured in Figure 4-1.

Figure 4-1. The Movie University network



DNS Manager

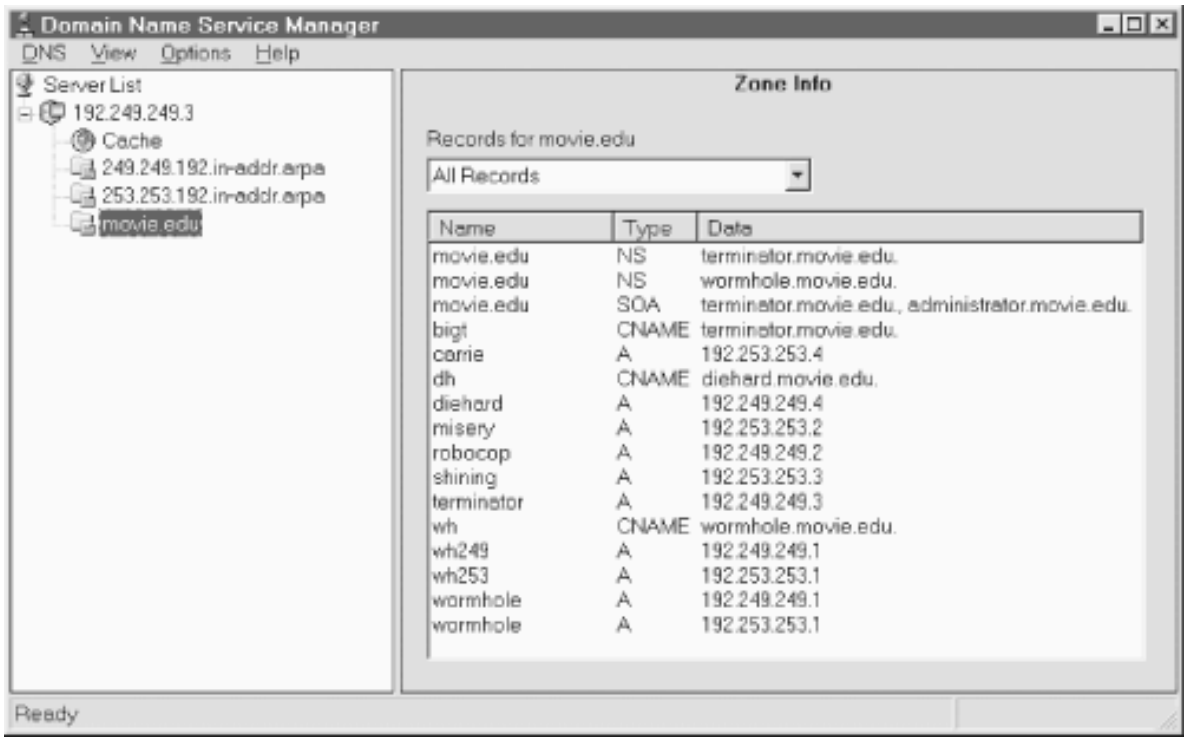
To manage a Microsoft DNS Server and maintain your DNS data, you'll use a tool called *DNS Manager*. It has a graphical user interface (surprise) and is capable of managing multiple name servers. DNS Manager is similar in design and operation to WINS Manager and DHCP Manager, if you're already familiar with those tools.

DNS Manager is located on the **Administrative Tools (Common)** menu, provided you've already installed the DNS service. If you don't see DNS Manager on that menu, see Appendix B, *Installing the DNS Server from CD-ROM*, for instructions on installing the DNS service. You can also run DNS Manager on Windows NT Workstation by copying the executable, *%SystemRoot%\system32\dnsadmin.exe*, from a Windows NT Server installation.

DNS Manager communicates with the Microsoft DNS Server using a proprietary management protocol built on Microsoft's RPC (remote procedure call). That means DNS Manager only manages Microsoft's DNS Server and not other name servers, like BIND.

The main DNS Manager window looks like Figure 4-2 (or will look like it, after we've set everything up in the course of this chapter):

Figure 4-2. DNS Manager main window



The left pane shows name servers, zones, and domains, while the right pane shows either name server statistics or resource records.

This particular DNS Manager knows about only one name server, with IP address 192.249.249.3. That name server is authoritative for three zones: *movie.edu*, *249.249.192.in-addr.arpa*, and *253.253.192.in-addr.arpa*. If any of these zones had subdomains, they would show up as subfolders under the appropriate zone. For example, *comedies.movie.edu* would be represented as a folder called *comedies* under *movie.edu*. Also notice the Cache icon. As you might expect, selecting this icon displays the contents of the name server's cache of resource records from previous queries. If you're familiar with the BIND name server, you probably know that the only way to examine that name server's cache is by dumping it to a file.

If a name server is selected in the left pane (instead of a zone, as in Figure 4-2), then the right panel displays usage statistics for that name server. You can see the total number of queries processed by the server and whether the transport was UDP or TCP, the total number of recursive queries, and the total number of lookups to a WINS server--more on that in Chapter 7, *Maintaining the Microsoft DNS Server*.

Of course, there are pull-down menus:

DNS

The really important commands are here: adding new name servers, creating zones and domains, and creating resource records. You can also delete objects and view objects' properties. We'll explain the various commands throughout this chapter.

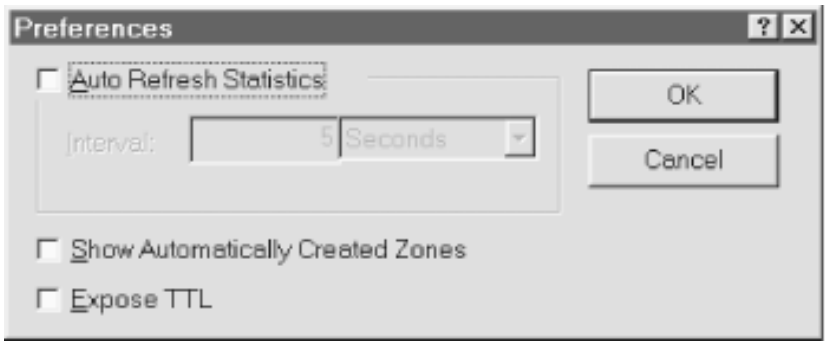
View

Only two commands here: **Split** lets you move the split bar between the two panes (something you can do by just selecting and dragging the bar). **Refresh** causes DNS Manager to query the name server for what's currently selected in the left pane and update its display in the right pane. In other words, DNS Manager's display doesn't track the contents of the name server in real time and is not updated automatically. *We can't overemphasize the importance of the Refresh command:* you should use it all the time to make sure that what you see on the screen is the same information the name server has in memory. Fortunately, the function key F5 is a quick shortcut for **Refresh**.

Options

This menu has only **Preferences**, which opens a window with DNS Manager settings as in Figure 4-3.

Figure 4-3. DNS Manager preferences



Auto Refresh Statistics causes DNS Manager to automatically update the previously mentioned name server statistics at the interval you select. Note that this automatic update only refreshes name server statistics, which you see only when you have a server • selected in the left pane. You'll usually have a zone selected on the left and view the zone's contents on the right. That information isn't updated automatically no matter what this setting--you'll still need to select **View • Refresh** (or hit F5) to see changes.

Show Automatically Created Zones displays three zones that every Microsoft name server is authoritative for: *0.in-addr.arpa*, *127.in-addr.arpa*, and *255.in-addr.arpa*. We'll explain what these zones are for later on in this chapter.

Expose TTL causes DNS to show the time to live (TTL) on every resource record it displays. Turning off this option doesn't affect the TTL of any records, just whether or not DNS Manager shows you the TTL.

The default for all three options is off.

Setting Up DNS Data

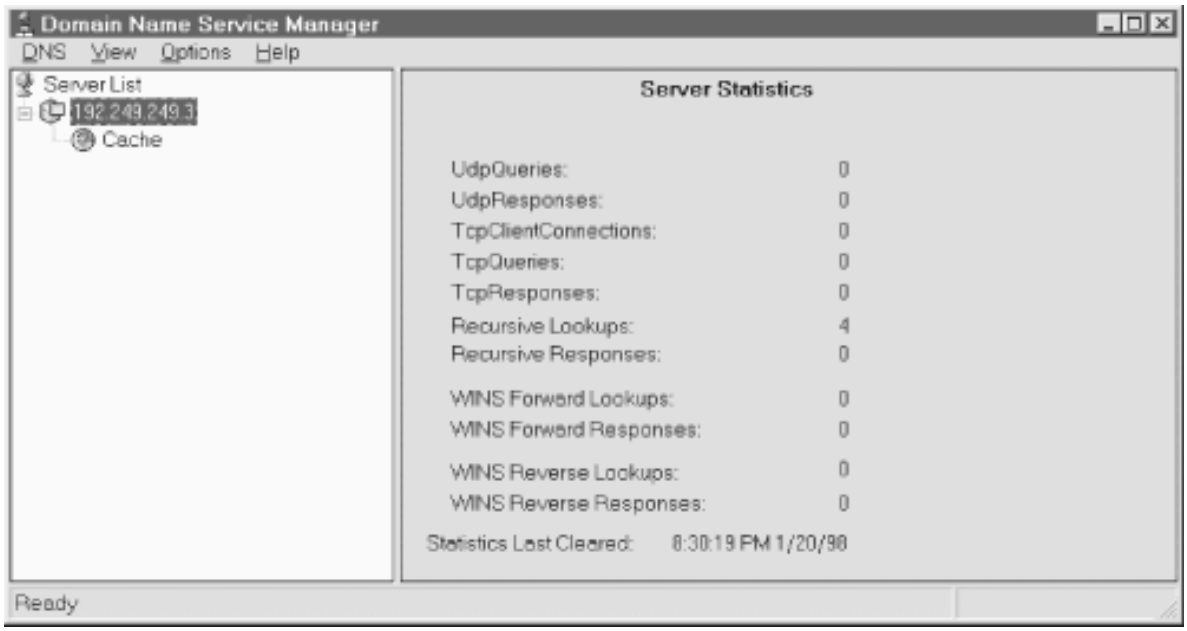
Let's configure the first of Movie U.'s name servers. We'll use DNS Manager for most of this process, so start it up if you haven't already done so. You don't have to run DNS Manager on the machine running the name server, but for now it's easier if you do. You'll also need to have Administrator privileges to use DNS Manager; otherwise, you'll only be able to start the application, not manage any name servers with it.

Adding a New Server to DNS Manager

The first step is configuring DNS Manager to manage the *primary master name server* for your zone. The primary master for a zone -- also called just the *primary* --stores information about the zone on its disk. You make all changes to your zone on the primary master.

Select **DNS • New Server**, and then enter the IP address of your primary master. DNS Manager adds an icon in the left pane for that name server as in Figure 4-4.

Figure 4-4. DNS Manager with a new server



192.249.249.3 is *terminator's* IP addresses in the Movie U. network. This name server is not (yet) authoritative for any zones, so only the Cache icon appears under the IP address. Since the name server itself is selected on the left, its usage statistics appear on the right. This name server hasn't been very busy.

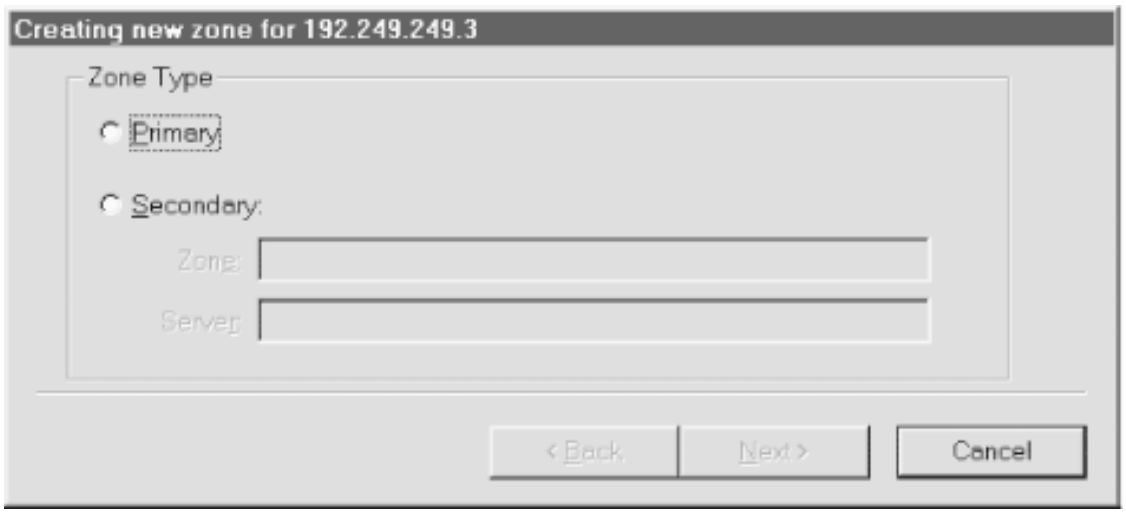
It's important to understand what we just did here. We told DNS Manager about a name server for it to manage. DNS Manager added that name server to its configuration, queried it for a list of zones and statistics, and displayed them. DNS Manager did *not* start the name server on the target machine. If the name server isn't already installed and running, DNS Manager can't manage it and will complain with the rather cryptic error, "There are no more endpoints available from the endpoint mapper." That error, which appears at the bottom of the right pane, means that DNS Manager couldn't contact the name server on the target machine.

Selecting **New Server** just adds that name server to DNS Manager's list of servers it knows about. As you might expect, selecting the server and choosing **DNS • Delete** (or just pressing the Delete key) removes the server from DNS Manager's configuration but doesn't change anything on the name server itself. The server will still be running--you can use **New Server** to add it, and you'll be right back where you started.

Creating a New Zone

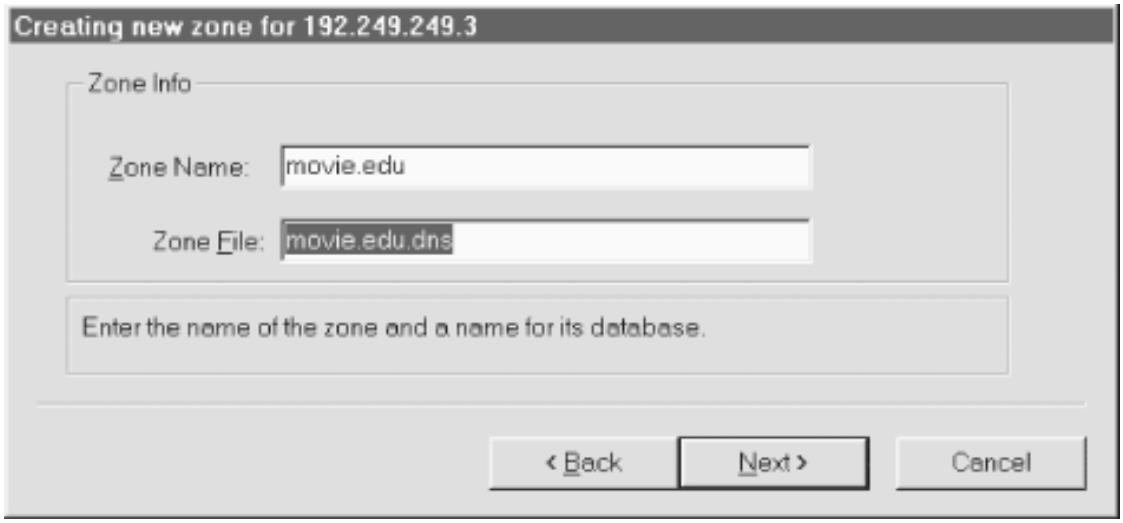
Now it's time to create the *movie.edu* zone. Select the name server on the left where you want to create the zone. (There's only one server now, *terminator*, but DNS Manager could know about multiple servers.) Choose **DNS • New Zone**. You'll see a window as in Figure 4-5.

Figure 4-5. Creating a new zone, first window



Choose **Primary**, and then click **Next**. In the next window, type the domain name of the zone in the **Zone Name** field. In our case, it's *movie.edu*. Notice that after you press Tab, the **Zone File** field is automatically filled in (see Figure 4-6).

Figure 4-6. Creating a new zone, second window



The zone file, also called a *zone database file*, is the zone's permanent storage location. It's the file on the name server's disk where all the information about the zone is stored: it contains all the zone's resource records. Other name servers require you to edit the zone database file to make changes to the zone, but DNS Manager allows you to avoid any hand editing. As a result, you probably won't see the zone database files very much. We'll talk about their format later in this chapter.

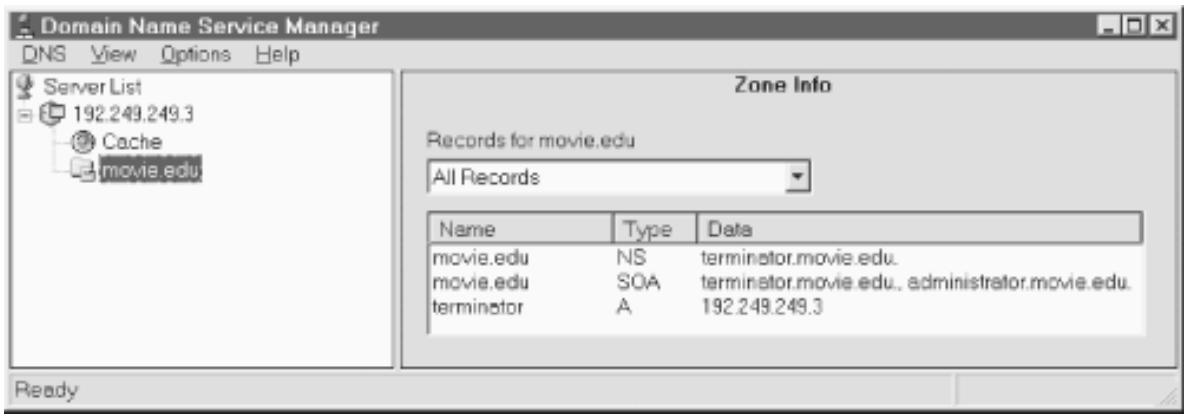
Even if you won't be looking at it often, you need to specify a zone database filename when you create a zone. The server expects these files to be in *%SystemRoot%\System32\DNS*. Microsoft's suggested naming convention uses the domain name of the zone followed by the *.dns* extension. You can name the zone file whatever you want, but as long as DNS Manager fills in the field for you, we recommend sticking with its suggestion. You may be familiar with other naming conventions, such as *db.* followed by a partial domain name, like *db.movie*. In fact, that's the recommendation in our sister book, *DNS and BIND*.

When you've entered a zone name and zone filename, click **Next**, and you'll see a window as in Figure 4-7.

Figure 4-7. Creating a new zone, third window

We're not quite sure why this window is necessary, but there it is. Click **Finish** to create the zone. You'll see a window like the one pictured in Figure 4-8.

Figure 4-8. DNS Manager with a new zone



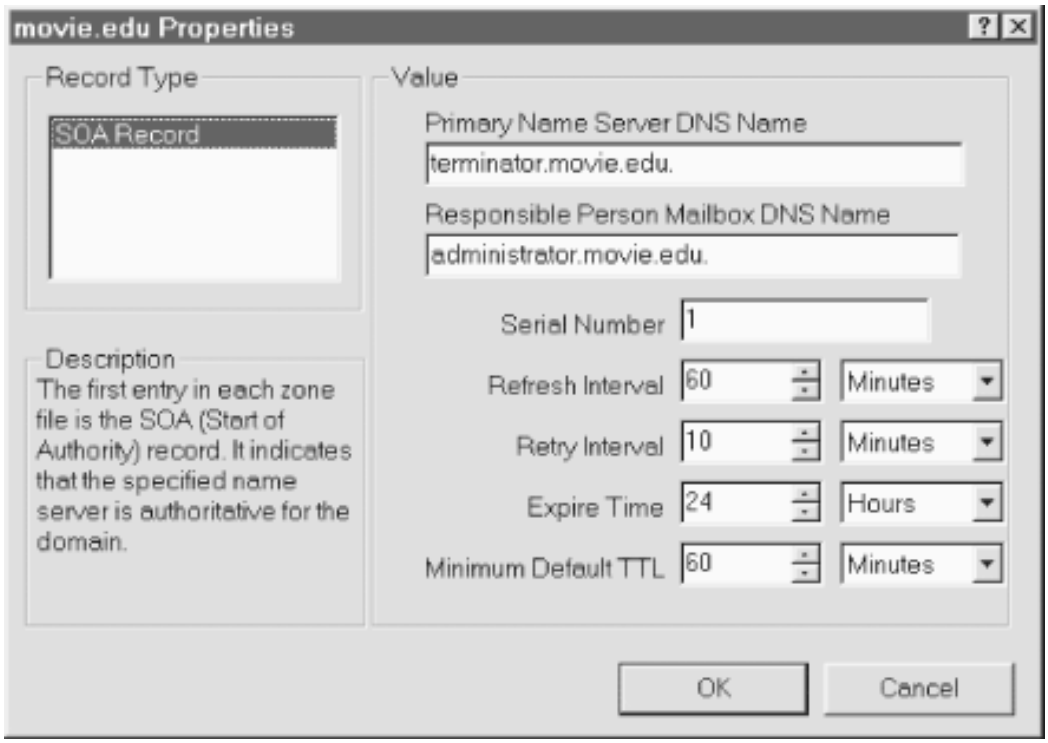
DNS Manager has created the zone and a few resource records. Let's talk about them one by one.

The SOA record

We'll start with the second record displayed. It's the SOA (start of authority) resource record. The SOA record indicates that this name server is the best source of information for the data within this zone. Our name server is *authoritative* for the zone *movie.edu* because of the SOA record. (Remember, this SOA record is attached to the domain name of the zone, *movie.edu*.) An SOA record is required in each zone. There can be one, and only one, SOA record in a zone.

Double-click the SOA record to view its details. You'll see a window like the one in Figure 4-9.

Figure 4-9. *The* movie.edu *SOA record*



The first field is the name of the primary master name server for this zone. (You may hear it called the MNAME field, which is its official name.) The second name (*administrator.movie.edu.*) is the email address of the person in charge of the data (if you replace the first dot with an at sign). DNS Manager defaults to *administrator*, but in other zones you'll often see *root*, *postmaster*, or *hostmaster* as the email address. Name servers won't use these names--they are meant for human consumption. If you notice a problem in someone's zone, you can send an email message to the listed email address.

Most of the remaining fields are for use by slave name servers and are discussed when we introduce slave name servers, later in this chapter. For now, assume these are reasonable values.

The NS record

The first record is an NS (name server) resource record. There should be one NS record for each name server for the zone. Like the SOA record, the NS records are attached to the zone's domain name. In our example, the NS records are attached to *movie.edu*. Right now there's only one name server (the primary master), but as we configure slave name servers, we'll add NS records. DNS Manager created an NS record for *terminator* because it's a name server for *movie.edu*.

The A record

The final automatically created record is an address record or A record. This record type fulfills the main purpose of DNS: it provides a name-to-address mapping. Each A record maps a domain name, like *terminator.movie.edu*, to an IP address, like 192.249.249.3.

When you create a new zone, DNS Manager creates an address record for the primary name server. It uses the host name configured in the primary master's DNS configuration.

Note that some abbreviating is going on in DNS Manager's display. For the SOA and NS records, the fully qualified domain name is shown, *movie.edu*. But for the A record, DNS Manager only displays *terminator* instead of *terminator.movie.edu*. DNS Manager normally displays a relative (that is, abbreviated) domain name on the right, so you have to look at what zone or domain is selected on the left to construct the fully qualified domain name. Only when records are attached to the name of a zone does it display a fully qualified name, as is the case with the *movie.edu* SOA and NS records.

You're probably anxious to add resource records for the rest of your zone, but it's best to create the reverse mapping (*in-addr.arpa*) zones first.

Creating a New Reverse Mapping Zone

Zones like *movie.edu* handle the mapping of names to addresses using A records. But mapping addresses back to names--reverse mapping--is just as important. As you recall from Chapter 2, *How Does DNS Work?*, a special portion of the name space, the *in-addr.arpa* domain, is designated for reverse mapping. Each domain name in *in-addr.arpa* corresponds to every possible IP address, and PTR records attached to a domain name provide the actual reverse mapping. Just think of a PTR record as the opposite of an A record.

So after we create *movie.edu*, we're not done. Movie U. has two class C networks, 192.249.249.0 and 192.253.253.0. We need to create the corresponding *in-addr.arpa* zones for reverse mapping with DNS Manager: *249.249.192.in-addr.arpa* and *253.253.192.in-addr.arpa*.

Creating an *in-addr.arpa* zone is the same as creating any other zone:

1. Select **DNS • New Zone**.
2. Choose **Primary**, and click **Next**.
3. In the next window, enter the name of the zone (249.249.192.in-addr.arpa, in this case), and press **Tab.**
4. We recommend accepting the automatically generated zone filename, but you can change it at this point if you'd like.
5. Click **Next**.
6. In the final window, click **Finish**.

Note that, just as it did with the *movie.edu* zone, DNS Manager automatically creates the SOA record and an NS record.

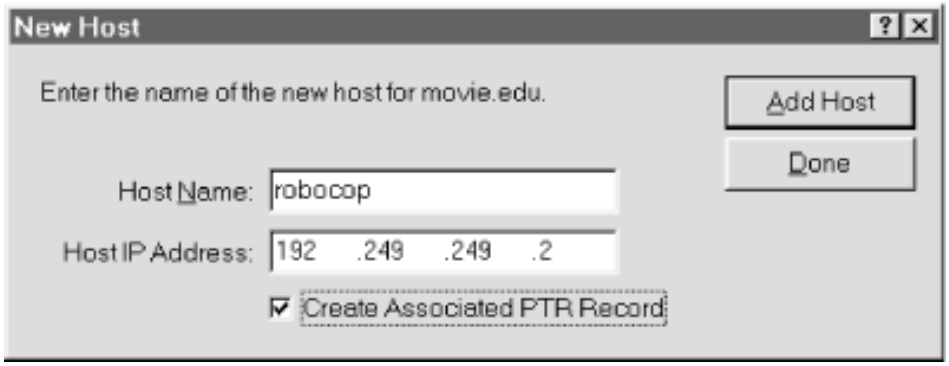
For Movie U., we'll repeat this process to create the *253.253.192.in-addr.arpa* zone. You would create *in-addr.arpa* zones according to the networks you have. Usually there's one *in-addr.arpa* zone per Class C (or sub-Class C) network. Larger networks, like Class A or Class B, are usually broken into several *in-addr.arpa* zones to make management easier. The zones usually correspond to subnets. This topic is covered in more detail in Chapter 9, *Parenting* .

Adding Resource Records

Now that we've created Movie U.'s zones, we can add information about all its machines. Each machine requires two resource records: an A record in the *movie.edu* zone to provide name-to-address mapping and a PTR record in the appropriate *in-addr.arpa* zone to provide address-to-name mapping. Adding the A record is intuitive, but it's easy to forget about the PTR record. DNS Manager makes the job easier with the **New Host** command, which creates an A record and a PTR record in one pass.

Select a forward mapping zone (like *movie.edu*), and choose **DNS • New Host.** Enter the name of the host and its IP address. To create the PTR record as well, you also need to check the **Create Associated PTR Record** box. The window looks like the one in Figure 4-10.

Figure 4-10. The new host window



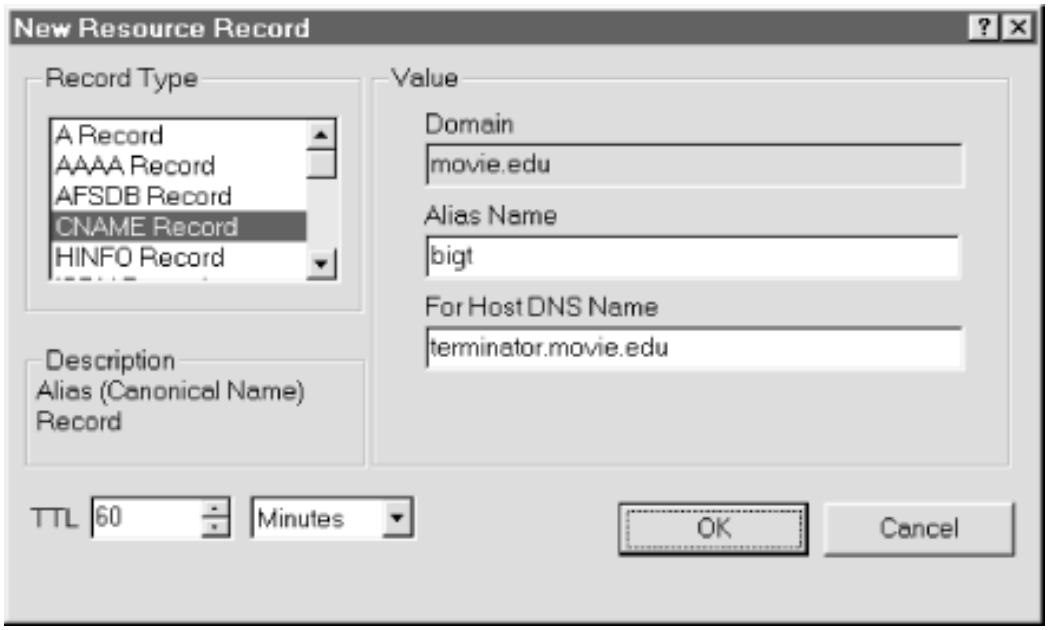
You'll notice that we typed a relative domain name (*robocop*) and not a fully qualified domain name (*robocop.movie.edu.)* DNS Manager requires a relative domain name in this field. It appends the domain name of the zone selected in DNS Manager's left pane to create a fully qualified domain name. Don't worry--if you try to enter a fully qualified domain name, DNS Manager will give you an error message.

Aliases

Looking back at Movie U.'s host table in the beginning of the chapter, you'll see that some hosts have aliases. (The aliases are any additional names after the first one listed.) For example, *terminator* is also known as *bigt*. There's a special resource record called the CNAME record that's used to make an alias. The name of the record is confusing, because CNAME is short for canonical name, which means the "real" name of the host. But a CNAME record doesn't make a canonical name; it makes an alias. All other types of records make a canonical name. We recommend thinking of it this way: CNAME records *point* to canonical names, while other record types *make* canonical names.

To create an alias, use the **New Record** command. Select the zone you want to add the record to on the left, and choose **DNS • New Record**. Unlike with **New Host**, you can create a record in any zone. The window looks like the one in Figure 4-11.

Figure 4-11. Creating a CNAME record



This window lets you add one of 17 different types of resource records. When you select the record type in the upper-left field, a brief description of the record type appears in the lower left, and the fields on the right change to accommodate the proper kind of data for the record type. Note that you can add A records and PTR records with this window, too.

We've selected CNAME, so fields for the alias name and canonical name (labeled as **For DNS Host Name**) appear on the right. The input in Figure 4-11 will generate an alias from *bigt.movie.edu* to *terminator.movie.edu*. The **Domain** field is just a reminder of the current domain. As was the case with the **New Host** command, you must enter a single-label (that is, no periods) name in the **Alias Name** field: the **Alias Name** field is always relative to the current domain. But there is no such restriction for the canonical name field. You can point an alias anywhere. We could alias *bigt.movie.edu* to *www.whitehouse.gov* if we wanted to. If you leave off the domain in the canonical name field, the zone's domain name will be appended automatically.

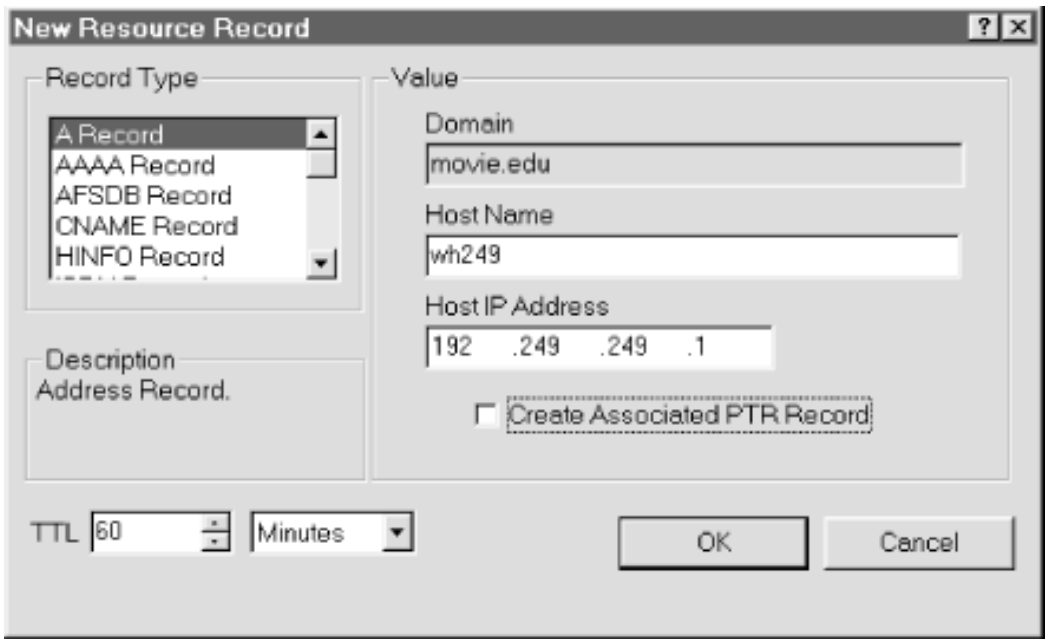
It's important to know that the name server handles CNAME records in a different manner than aliases are handled in the host table. When a name server looks up a name and finds a CNAME record, it replaces the alias name with the canonical name and looks up that new name. For example, when the name server looks up *bigt.movie.edu*, it finds a CNAME record pointing to *terminator.movie.edu*. Then it looks up *terminator.movie.edu*, and its address is returned.

One thing you must remember about aliases like *bigt* --they should never appear in the data portion (that is, on the right side) of a resource record. Stated differently: always use the canonical name (*terminator*) in the data portion of the resource record. Notice that the NS records use the canonical name.

Sometimes you can use an A record to get the effect of an alias. Suppose you have a router, like *wormhole*, and you want to check one of the interfaces. One common troubleshooting technique is to *ping* the interface to verify that it is responding. If you *ping* the name *wormhole*, the name server returns the addresses of both interfaces when the name is looked up. *ping* uses the first address in the list. But which address is first?

The solution is to create two A records for *wormhole* with **New Record**. (The first of the two records is shown in Figure 4-12.)

Figure 4-12. *Creating the first of two A records for* wormhole



With the host table, we chose the address we wanted by using either *wh249* or *wh253* -- each name referred to *one* of the host's addresses. To provide equivalent capability with DNS, we didn't make *wh249* and *wh253* into aliases (CNAME records). That would result in both addresses for *wormhole* being returned when the alias was looked up. Instead, we used address records. Now, to check the operation of the 192.253.253.1 interface on *wormhole*, we *ping* *wh253* since it refers to only one address. The same applies to *wh249*.

To state this as a general rule: if a host is multihomed (has more than one network interface), create an address (A) record for each alias unique to one address. Create a CNAME record for each alias common to all the addresses.

One more note about PTR records

We now have two A records, *wormhole.movie.edu* and *wh249.movie.edu*, pointing to the same address, 192.249.249.1. We also have a PTR record pointing from *1.249.249.192.in-addr.arpa* to *wormhole.movie.edu*. (This record was added automatically to the *249.249.192.in-addr.arpa* zone by the **New Host** command. Remember that addresses are looked up as names: the IP address is reversed, and *in-addr.arpa* is appended.) Thus, 192.249.249.1 maps to *wormhole.movie.edu* and not to *wh249.movie.edu*. Should you create another PTR record that maps 192.249.249.1 to *wh249.movie.edu* ? You *can* create two PTR records--it's perfectly legal--but most systems are not prepared to see more than one name for an address. We recommend that you don't bother with multiple PTR records since so few systems can use them.

Where Is All This Information Stored?

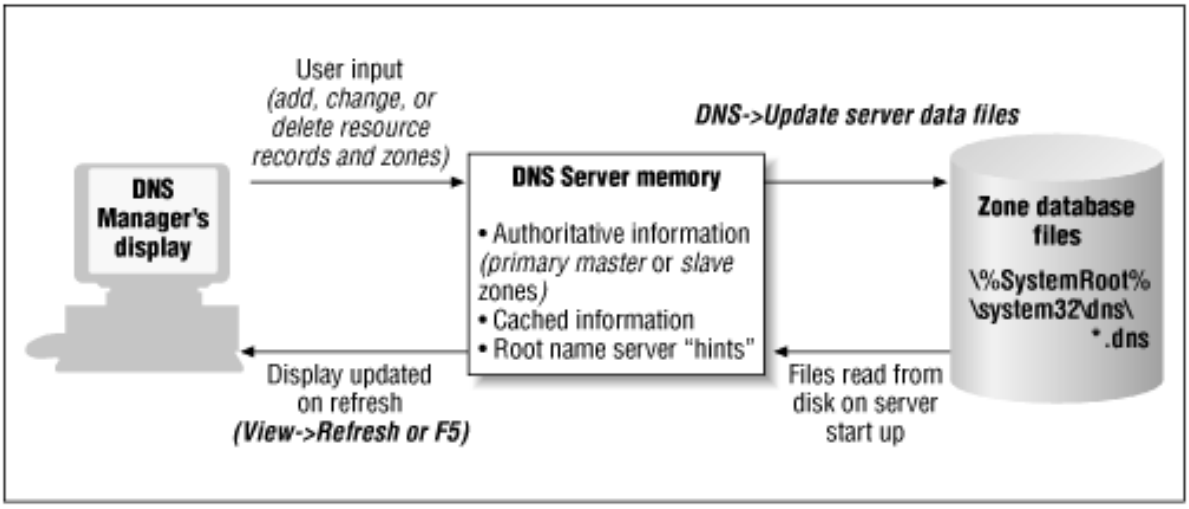
You may be wondering what's been happening to all the resource records we've been entering. Where are they being stored? The answer is: in the memory of the DNS Server process. We mentioned earlier that DNS Manager communicates with the DNS Server using an RPC mechanism. As you add records to a zone with DNS Manager, they are added "on the fly" to the name server's memory. Of course, the name server's memory is transient--when the name server process stops, its memory is lost. Obviously a permanent storage location is needed.

This is where the zone database files we specified when we created the zones come in. The zone database files are the zones' permanent storage location, holding all the zones' resource records. If you use DNS Manager to make a change to a zone, the copy of the zone in the name server's memory is changed, and a flag is set to update that zone's database file. The name server updates the zone database file when it exits, unless you tell it to update it sooner. The command **DNS • Update Server Data Files** causes the name server to update the zone database files of all zones it's a primary for, regardless of whether or not they've been modified. To avoid losing data, we recommend using **DNS • Update Server Data Files** after any changes--use it like you use the **Save** command in other applications. Of course, the difference here is that the server will save your data if it exits gracefully. You don't have to use **DNS • Update Server Data Files** after a batch of changes, but it doesn't hurt anything and you can sleep better.

As you've probably guessed, when the name server starts up, it reads the zone database files into memory. When you select **View • Refresh** or press F5, DNS Manager queries the name server and updates its display.

If you've been keeping track, you realize that DNS information exists in three places: zone database files, the name server's memory, and DNS Manager's window. The diagram in Figure 4-13 helps explain how the information flows.

Figure 4-13. *Where everything is stored*



The Zone Database Files

Let's take a look at the zone database files for Movie U. After inputting the remaining host table entries, we end up with the display shown previously in Figure 4-2. (Of course, this view shows only the contents of *movie.edu*. The *249.249.192.in-addr.arpa* and *253.253.192.in-addr.arpa* zones are populated with PTR records.)

Next we select **DNS • Update Server Data Files**, and the server generates three files in *%SystemRoot%\System32\DNS* : *movie.edu.dns*, *249.249.192.in-addr.arpa.dns*, and *253.253.192.in-addr.arpa.dns*. They look like the following:

Contents of *movie.edu.dns*:

```
;
;  Database file movie.edu.dns for movie.edu zone.
;      Zone version:  4l
;

@             IN       SOA terminator.movie.edu.administrator.movie.edu.(
                       1           ; serial number
                       3600        ; refresh
                       600         ; retry
                       86400       ; expire
                       3600      ) ; minimum TTL

;
;  Zone NS records
;

@              IN     NS      terminator

;
;  Zone records
;

bigt                    IN             CNAME    terminator
carrie                  IN             A       192.253.253.4
dh                      IN             CNAME   diehard
diehard                 IN             A       192.249.249.4
misery                  IN             A       192.253.253.2
robocop                 IN             A       192.249.249.2
shining                 IN             A       192.253.253.3
terminator              IN             A       192.249.249.3
wh                      IN             CNAME    wormhole
wh249                   IN             A       192.249.249.1
wh253                   IN             A       192.253.253.1
wormhole                IN             A       192.249.249.1
                        IN             A       192.253.253.1
```

Contents of *249.249.192.in-addr.arpa.dns*:

```
;
;  Database file 249.249.192.in-addr.arpa.dns for 249.249.192.in-addr.arpa zone.
;      Zone version:  5l
```

```
;

@               IN      SOA     terminator.movie.edu.administrator.movie.edu.(
                        5           ; serial number
                        3600        ; refresh
                        600         ; retry
                        86400       ; expire
                        3600      ) ; minimum TTL

;
;  Zone NS records
;

@               IN    NS      terminator.movie.edu.

;
;  Zone records
;

1                       IN      PTR     wormhole.movie.edu.
2                       IN      PTR     robocop.movie.edu.
3                       IN      PTR     terminator.movie.edu.
4                       IN      PTR     diehard.movie.edu.
```

Contents of *253.253.192.in-addr.arpa.dns*:

```
;
;  Database file 253.253.192.in-addr.arpa.dns for 253.253.192.in-addr.arpa zone.
;      Zone version:  4l
;

@               IN      SOA terminator.movie.edu.administrator.movie.edu.(
                        4           ; serial number
                        3600        ; refresh
                        600         ; retry
                        86400       ; expire
                        3600      ) ; minimum TTL

;
;  Zone NS records
;

@                IN   NS      terminator.movie.edu.

;
;  Zone records
;

1                       IN      PTR     wormhole.movie.edu.
2                       IN      PTR     misery.movie.edu.
3                       IN      PTR     shining.movie.edu.
4                       IN      PTR     carrie.movie.edu.
```

Zone Database File Format

The format of zone database files is specified in the DNS standards. That means all name servers, whether Microsoft's DNS Server or the BIND name server, can read each other's zone database files.

You've probably already guessed that the semicolon is the comment character. It can appear anywhere on a line, and anything to the right is a comment and is ignored by the name server. Blank lines are okay, too.

Each resource record must start in the first column of the file--no preceding whitespace. (Don't be confused by the examples in this book, which are indented because of the way the book is formatted.) Resource records are case insensitive--you can use uppercase or lowercase. The server doesn't preserve the case, though. It matches the case of the reply to the case of the query. For example, if a record is written as *terminator* in the zone database file but you query for *Terminator*, the server responds with *Terminator*.

Resource records are broken up into fields, with any amount of whitespace (tabs or spaces) separating the fields.

The first field, called the owner, is the domain name of the record. Put another way, it's the node in the name space that the resource record is attached to. You've seen the domain name on the *left* side of the *right* pane of DNS Manager. (Got that?)

The next field in our examples is the class, IN, which stands for Internet. There are other classes, but none of them are currently in widespread use. Our examples use only the IN class.

The field after that is the record type. We've already discussed record types SOA, NS, A, PTR, and CNAME, and you've probably browsed through the list of other record types in DNS Manager's **New Record** window. The type simply specifies what type of data is associated with the domain name on the right: A means IP address, NS means the name of an authoritative name server, and so on.

And that's a good lead in to the final field, the RDATA or resource record data. This field holds the kind of data specified by the record type. This field can be divided into multiple subfields depending on the type. For example, A records may specify only one parameter: an IP address. But the SOA record specifies seven parameters: remember all those fields in Figure 4-9?

Speaking of the SOA record, you'll notice in the examples that it's the only record spanning multiple lines. You can use parentheses to allow a resource record to span multiple lines. This trick works for all record types, not just SOA.

Domain names appear a lot in resource records. The left side of every resource record is a domain name, and the right side (RDATA field) of many record types also contains domain names (for example, NS and SOA records). Using a fully qualified domain name in each case is perfectly legal, but it would be a lot of work: imagine having to type *movie.edu* at the end of every host name if you were entering these files by hand. Fortunately, abbreviations are allowed. You need to understand them to decipher the zone database files in this chapter, because the records generated by the Microsoft DNS Server use these abbreviations.

Appending domains

Every zone has a domain name: it's just the name of the zone. (This probably strikes you as pretty obvious.) This domain name is the key to the most useful shortcut. This domain name is the *origin* of all the data in the database file. The origin is appended to all names in the file not ending in a dot. The origin is different for each file, because each file is associated with a different zone, each of which has a different name.

Since the origin is appended to names, instead of entering *robocop*'s address in *movie.edu.dns* as this:

```
robocop.movie.edu.    IN A     192.249.249.2
```

the server generated it like this:

```
robocop    IN A     192.249.249.2
```

In *192.249.249.in-addr.arpa.dns,* this is the long way to write this record:

```
2.249.249.192.in-addr.arpa.  IN PTR robocop.movie.edu.
```

But since *249.249.192.in-addr.arpa* is the origin, the server generated:

```
2  IN PTR robocop.movie.edu.
```

Notice that all the fully qualified domain names in the file end in a dot. That tells the server that this domain name is complete and should be left alone. Suppose you forgot the trailing dot. An entry like this:

```
robocop.movie.edu    IN A     192.249.249.2
```

turns into an entry for *robocop.movie.edu.movie.edu,* and you didn't intend that at all.

@ notation

If the domain name is the *same* as the origin, the name can be specified with an at sign (@). This is most often seen in the SOA record in database files generated by hand, but the Microsoft DNS server also uses the @ notation in the NS records. In the *movie.edu.dns* file in the previous example, the @ stands for *movie.edu*. Of course, in the *249.249.192.in-addr.arpa.dns* file, the @ stands for *249.249.192.in-addr.arpa,* and in the *253.253.192.in-addr.arpa.dns* file . . . well, you get the idea.

Repeat last name

If a resource record name (that starts in column one) is a space or tab, then the name from the last resource record is used. This shortcut gets used when there are multiple resource records for a name. Here is an example where there are two address records for one name:

```
wormhole   IN A     192.249.249.1
           IN A     192.253.253.1
```

In the second address record, the name *wormhole* is implied. You can use this shortcut even if the resource records are of different types--for example, if *wormhole* also had a TXT (arbitrary text) record.

The Loopback Address

Those of you familiar with the BIND name server may be wondering if we forgot about the loopback address. If we were setting up a BIND name server, it would need one additional zone database file to cover the *loopback* network: the special address that hosts use to direct traffic to themselves. This network is (almost) always 127.0.0.0, and the host number is (almost) always 127.0.0.1. Therefore, the name of this file would be *0.0.127.in-addr.arpa.dns,* and it would look like the other *in-addr.dns* files.

The following would be the contents of file *0.0.127.in-addr.arpa.dns*:

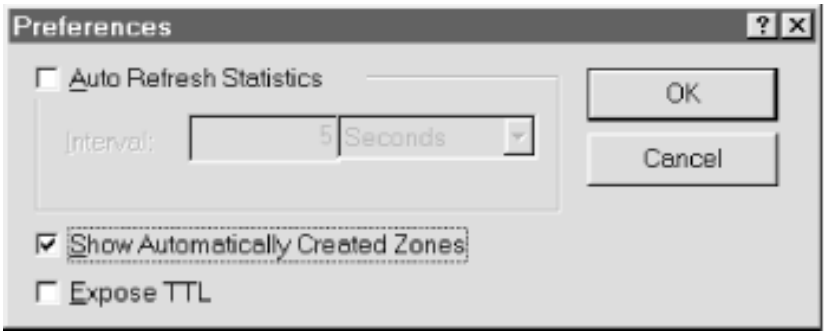
```
@                IN     SOA    terminator.movie.edu.administrator.movie.edu.(
                        1           ; serial number
                        3600        ; refresh
                        600         ; retry
                        86400       ; expire
```

```
                        3600      ) ; minimum TTL

;
;  Zone NS records
;

@                IN   NS      terminator.movie.edu.

;
;  Zone records
;

1                   IN       PTR  localhost.
```

Why do name servers need this file? Think about it for a second. No one was given responsibility for network 127.0.0.0, yet systems use it for a loopback address. Since no one has direct responsibility, everyone who uses it is responsible for it individually. If you omit this file on a BIND name server, it will still operate. However, a lookup of 127.0.0.1 might fail: the name server will send the query to a root name server that might not be configured to map 127.0.0.1 to a name.

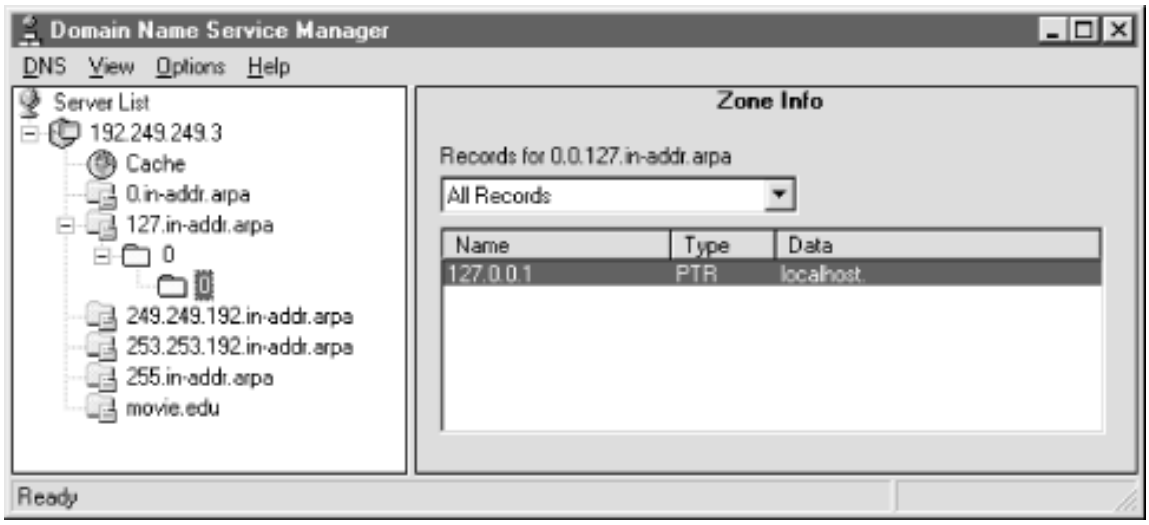
With the Microsoft DNS Server, you don't have to worry about creating this file and making your name server authoritative for the *in-addr.arpa* zone corresponding to network 127.0.0.0. The server is authoritative for this zone by default. It's called an *automatically created zone* and is visible in DNS Manager only if an option is set. Select **Options • Preferences**, and you'll see the window shown in Figure 4-14.

Figure 4-14. ***DNS Manager preferences set to show automatically created zones***



Check the **Show Automatically Created Zones** box, and click **OK**. Then select the name server in the left pane, and refresh (**View • Refresh** or F5). You'll see three more zones in your display as shown in Figure 4-15.

Figure 4-15. ***DNS Manager showing automatically created zones***



We've drilled down into the *127.in-addr.arpa* zone to show that there's a PTR record for *1.0.0.127.in-addr.arpa* pointing to the domain name *localhost*. In other words, a Microsoft DNS Server will reverse-map the IP address 127.0.0.1 to the domain name *localhost* "out of the box" without any work on your part.

The *0.in-addr.arpa* and *255.in-addr.arpa* zones are empty, save for NS and A records. Some hosts attempt to reverse-map the IP addresses 0.0.0.0 and 255.255.255.255, and these zones cause the local server to return an immediate NXDOMAIN (name not found) error for those queries, rather then asking a root name server.

The Root Cache Data

Besides needing to know your local information, the name server also needs to know where the name servers for the root zone are. (Remember the resolution process starts at the root zone, so knowing which name servers are authoritative for the root zone is critical.) This information must be retrieved from the Internet host *ftp.rs.internic.net* (198.41.0.7). Use anonymous *ftp* to retrieve the file *named.root* from the *domain* subdirectory. The file, called the *root name server cache file*, should be named *%SystemRoot%\System32\DNS\ cache.dns* on your name server. Here's the version of the file that was current when this book was published:

```
;       This file holds the information on root name servers needed to
;       initialize cache of Internet domain name servers
;       (e.g. reference this file in the "cache  .  <file>"
;       configuration file of BIND domain name servers).
;
;       This file is made available by InterNIC registration services
;       under anonymous FTP as
;           file                /domain/named.root
;           on server           FTP.RS.INTERNIC.NET
;       -OR- under Gopher at    RS.INTERNIC.NET
;           under menu          InterNIC Registration Services (NSI)
;              submenu          InterNIC Registration Archives
;           file                named.root
;
;       last update:    Aug 22, 1997
;       related version of root zone:   1997082200
;
;
; formerly NS.INTERNIC.NET
;
```

```
.                       3600000  IN  NS    A.ROOT-SERVERS.NET.
A.ROOT-SERVERS.NET.     3600000      A     198.41.0.4
;
; formerly NS1.ISI.EDU
;
.                       3600000      NS    B.ROOT-SERVERS.NET.
B.ROOT-SERVERS.NET.     3600000      A     128.9.0.107
;
; formerly C.PSI.NET
;
.                       3600000      NS    C.ROOT-SERVERS.NET.
C.ROOT-SERVERS.NET.     3600000      A     192.33.4.12
;
; formerly TERP.UMD.EDU
;
.                       3600000      NS    D.ROOT-SERVERS.NET.
D.ROOT-SERVERS.NET.     3600000      A     128.8.10.90
;
; formerly NS.NASA.GOV
;
.                       3600000      NS    E.ROOT-SERVERS.NET.
E.ROOT-SERVERS.NET.     3600000      A     192.203.230.10
;
; formerly NS.ISC.ORG
;
.                       3600000      NS    F.ROOT-SERVERS.NET.
F.ROOT-SERVERS.NET.     3600000      A     192.5.5.241
;
; formerly NS.NIC.DDN.MIL
;
.                       3600000      NS    G.ROOT-SERVERS.NET.
G.ROOT-SERVERS.NET.     3600000      A     192.112.36.4
;
; formerly AOS.ARL.ARMY.MIL
;
.                       3600000      NS    H.ROOT-SERVERS.NET.
H.ROOT-SERVERS.NET.     3600000      A     128.63.2.53
;
; formerly NIC.NORDU.NET
;
.                       3600000      NS    I.ROOT-SERVERS.NET.
I.ROOT-SERVERS.NET.     3600000      A     192.36.148.17
;
; temporarily housed at NSI (InterNIC)
;
.                       3600000      NS    J.ROOT-SERVERS.NET.
J.ROOT-SERVERS.NET.     3600000      A     198.41.0.10
;
; housed in LINX, operated by RIPE NCC
;
.                       3600000      NS    K.ROOT-SERVERS.NET.
K.ROOT-SERVERS.NET.     3600000      A     193.0.14.129
```

```
;
; temporarily housed at ISI (IANA)
;
.                        3600000      NS    L.ROOT-SERVERS.NET.
L.ROOT-SERVERS.NET.      3600000      A     198.32.64.12
;
; housed in Japan, operated by WIDE
;
.                        3600000      NS    M.ROOT-SERVERS.NET.
M.ROOT-SERVERS.NET.      3600000      A     202.12.27.33
; End of File
```

The domain name "." refers to the root domain. Since the root domain's name servers change over time, don't assume *this* list is current. Pull a new version of *named.root*.

It's worth noting that the root NS records are not put into the cache and used directly. Rather, upon startup the server queries one of the root servers in the cache file for the list of root servers. The list returned is the one used by the name server to start the resolution process and is the list you see when you double-click the Cache icon. When the name server exits, its list of root name servers is written to the cache file.

The nice thing about this behavior is that if you use an older cache file, as long as at least one of the name servers specified has the correct list of root name servers, your name server will discover the up-to-date list. And because the name server updates *cache.dns* on exit, it'll have the right list to begin with the next time it starts. The whole purpose of this logic is to ensure the name server has the current list of root name servers, which is very important for the resolution process to work properly.

You might be wondering if you can put information other than root name server NS records and their corresponding A records in the cache file. You can, but we don't recommend it. For one thing, the name server only rewrites root NS records (and their A records) to the cache file, so any changes you make will be overwritten eventually. In addition, BIND name servers don't have this feature. If your DNS architecture depends on any preloaded information in the cache, you're locked into using only Microsoft DNS Server as a name server platform. Yet another reason we think this is a bad idea is that you have to remember the information is there and maintain it by hand when it changes. We suggest you let the name server populate its own cache and don't interfere.

You may be wondering what the *3600000*s are for. In older versions of this file, this number used to be *99999999*. It dates back to the behavior of early versions of BIND, the reference implementation of the name server. The BIND name server used to put the contents of the cache file directly into its cache, and it had to know how long to keep these records active. The *99999999*s meant a *very long time*. The root name server data was to be kept active for as long as the server ran. Since both BIND and the Microsoft DNS Server now store the cache file data in a special place and don't discard it if it times out, the TTL is unnecessary. But it's not harmful to have the *3600000*s, and it makes for interesting DNS folklore when you pass responsibility to the next name server administrator.

One final note about the *cache.dns* file. *To install a new cache.dns file, you must first stop the server*. Remember, the server will overwrite the *cache.dns* file when it exits. Let's say you install a new file and then stop and start the server. As soon as the server stopped, your new file was overwritten with the server's list of root name servers. It then restarts, reads that list back in, and queries one of those name servers for the list of root name servers (which will almost certainly be the same list). The solution is to stop the server, replace the *cache.dns* file, and restart the server.

Running a Primary Master Name Server

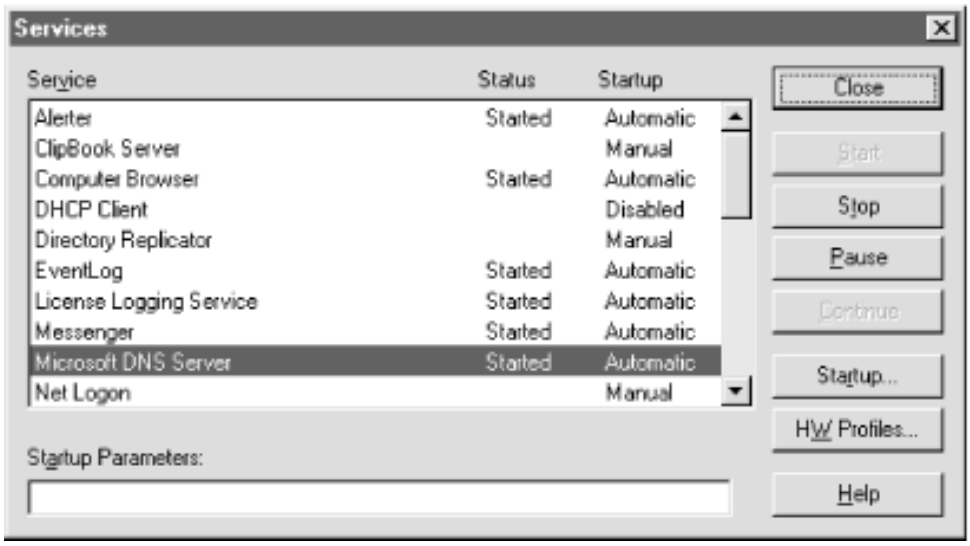
Your primary name server is already up and running; you've been talking to it via DNS Manager. You've created a zone and populated it with information. Then you directed the server to write out zone database files with the **DNS • Update Server Data Files** command. To be sure that everything is okay, you should stop and restart the server and then check

the Event Log for any messages or errors.

Starting and Stopping the DNS Server

You start and stop the DNS server just like any other NT service: with the **Services** Control Panel document. Open the Control Panel with **Start • Settings • Control Panel**. Double-click the **Services** icon, and you'll see the **Services** window as shown in Figure 4-16.

Figure 4-16. ***Windows NT services control window***



Your system should look like this: the server should be running (that is, it should be started). Select the server as we've done by clicking anywhere on the Microsoft DNS Server line. Click **Stop**. You'll be prompted for confirmation; click **Yes**. After the server stops, click **Start**. In a few seconds, the server should be running again.

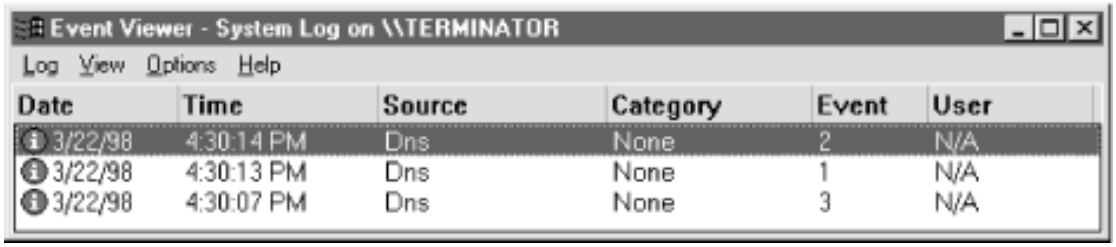
While you've got this window open, check to make sure that the DNS Server is being started automatically on bootup. You want to see **Automatic** in the **Startup** column (and not **Manual** or **Disabled**). To change the startup behavior, just click the **Startup** button.

You can also start and stop the DNS Server from the DOS command line: net start dns will start the server, and net stop dns stops it.

Check the Event Log for Messages and Errors

Now you need to check the Event Log. Start the Event Viewer by selecting **Start • Programs • Administrative Tools (Common) • Event Viewer**. Be sure you're looking at system events: choose **Log • System**. You should see a window like the one in Figure 4-17.

Figure 4-17. ***Event Viewer***



DNS Server Event ID 3 is "The DNS Server has shutdown." Event ID 1 is "Starting Microsoft DNS Server (v4.0 ServicePack3)," and Event ID 2 is "The DNS Server has started." (More events are listed in Chapter 7.) These three events are just what you want to see: a normal server shutdown and startup. We're reading from bottom to top since

Event Viewer's default view is newest events first. We also cleared the Event Log before we stopped and started the server--that's why only DNS events are showing.

If there were any other messages or errors, we'd take steps to correct them now. To be honest, we didn't expect any problems because we entered all the data via DNS Manager. Since it performs some syntax and sanity checking, it's hard to enter bad data to make the name server upset enough to complain in the Event Log. Still, it doesn't hurt to check. If you ever start editing zone database files by hand, you definitely need to check the Event Log.

Testing Your Setup with nslookup

If you have correctly set up your local domain and your connection to the Internet is up, you should be able to look up a local and remote name. We'll step you through the lookups with *nslookup*. This book contains an entire chapter on this topic (Chapter 11, *nslookup*), but we will cover *nslookup* in enough detail here to do basic name server testing.

Look up a local name

nslookup can be used to look up any type of resource record, and it can be directed to query any name server. By default, it looks up A (address) records using the name server on the local system. To look up a host's address with *nslookup*, run *nslookup* with the host's name as the only argument. A lookup of a local name should return almost instantly.

We ran *nslookup* to look up *carrie* :

```
C:\> nslookup carrie
Server:  terminator.movie.edu
Address:  192.249.249.3

Name:    carrie.movie.edu
Address:  192.253.253.4
```

If looking up a local name works, your local name server has been configured properly for your domain. If the lookup fails, you'll see something like this:

```
*** terminator.movie.edu can't find carrie: Non-existent domain
```

This means that either *carrie* is not in your data--check DNS Manager or the zone database file--or some name server error occurred (but you should have caught the error when you checked the Event Log).

Look up a local address

When *nslookup* is given an address to look up, it knows to make a PTR query instead of an address query. We ran *nslookup* to look up *carrie*'s address:

```
C:\> nslookup 192.253.253.4
Server:  terminator.movie.edu
Address:  192.249.249.3

Name:    carrie.movie.edu
Address:  192.253.253.4
```

If looking up an address works, your local name server has been configured properly for your *in-addr.arpa* domain. If the lookup fails, you'll see the same error messages as when you looked up a name.

Look up a remote name

The next step is to use the local name server to look up a remote name, like *ftp.uu.net*, or another system you know on the Internet. This command may not return as quickly as the last one. If *nslookup* fails to get a response from your name server, it will wait a little longer than a minute before giving up:

```
C:\> nslookup ftp.uu.net.
Server:  terminator.movie.edu
Address:  192.249.249.3

Name:    ftp.uu.net
Address:  192.48.96.9
```

If this works, your name server knows where the root name servers are and how to contact them to find information about domains other than your own. If it fails, either you forgot to initialize the cache file (and a message in the Event Log will show up) or the network is broken somewhere and you can't reach the name servers for the remote domain. Try a different remote domain name.

If these first three lookups succeeded, congratulations! You have a primary master name server up and running. At this point, you are ready to start configuring your slave name server.

One more test

While you are testing, though, run one more test. Try having a remote name server look up a name in your zone. This is going to work only if your parent name servers have already delegated your zone to the name server you just set up. If your parent required you to have your two name servers running before delegating your zone, skip ahead to the section "Running a Slave Name Server ."

To make *nslookup* use a remote name server to look up a local name, give the local host's name as the first argument, and the remote server's name as the second argument. Again, if this doesn't work, it may take a little longer than a minute before *nslookup* gives you an error message. For instance, to have *gatekeeper.dec.com* look up *carrie* :

```
C:\> nslookup carrie gatekeeper.dec.com.
Server:  gatekeeper.dec.com.
Address:  204.123.2.2

Name:    carrie.movie.edu
Address:  192.253.253.4
```

If the first two lookups worked, but using a remote name server to look up a local name failed, you may not be registered with your parent name server. That is not a problem at first, because systems within your zone can look up the names of other systems within and outside your zone. You'll be able to send email and *ftp* to local and remote systems. Some systems won't allow FTP connections if they can't map your address back to a name. But not being registered will shortly become a problem. Hosts outside of your zone cannot look up names within your zone. You will be able to send email to friends in remote domains, but you won't get their responses. To fix this problem, contact someone responsible for your parent zone and have them check the delegation of your zone.

Running a Slave Name Server

You need to set up another name server for robustness. You can (and probably will) set up more than two name servers. Two servers are the minimum. If you have only one name server and it goes down, no one can look up names in your zone. A second name server splits the load with the first server or handles the whole load if the first server is down. You *could* set up another primary master name server, but we don't recommend it. Set up a slave name server.

How does a server know if it is a primary master or a slave for a zone? The DNS Server configuration information in the Registry tells the server it is a primary master or a slave on a per zone basis. The NS records don't tell us which servers are primary master for a zone and which servers are slave for a zone--they only say who the servers are. (Globally, DNS doesn't care; as far as the actual name resolution goes, slave servers are as good as primary master servers.)

What is different between a primary master name server and a slave name server? The crucial difference is where the server gets its data. A primary master name server reads its data from files. A slave name server loads its data over the network from another name server. This process is called a *zone transfer*.

A slave name server is not limited to loading zones from a primary master name server; a slave server can load from another slave server.

The big advantage of slave name servers is that you only maintain one set of the DNS database files, the ones on the primary master name server. You don't have to worry about synchronizing the files among name servers; the slaves do that for you.

A slave name server doesn't need to retrieve *all* of its db files over the network; the *cache.dns* is the same as on a primary master, so keep a local copy on the slave.

> One point about slaves may become confusing: slaves used to be called *secondary master* name servers. The terminology was changed since DNS Manager came out, and now everyone "in the know" uses the term *slave*. We'll use the term *slave* in this book, but you'll see that DNS Manager still uses the term *secondary*. As we said, the two are synonymous.

Add a New Server to DNS Manager

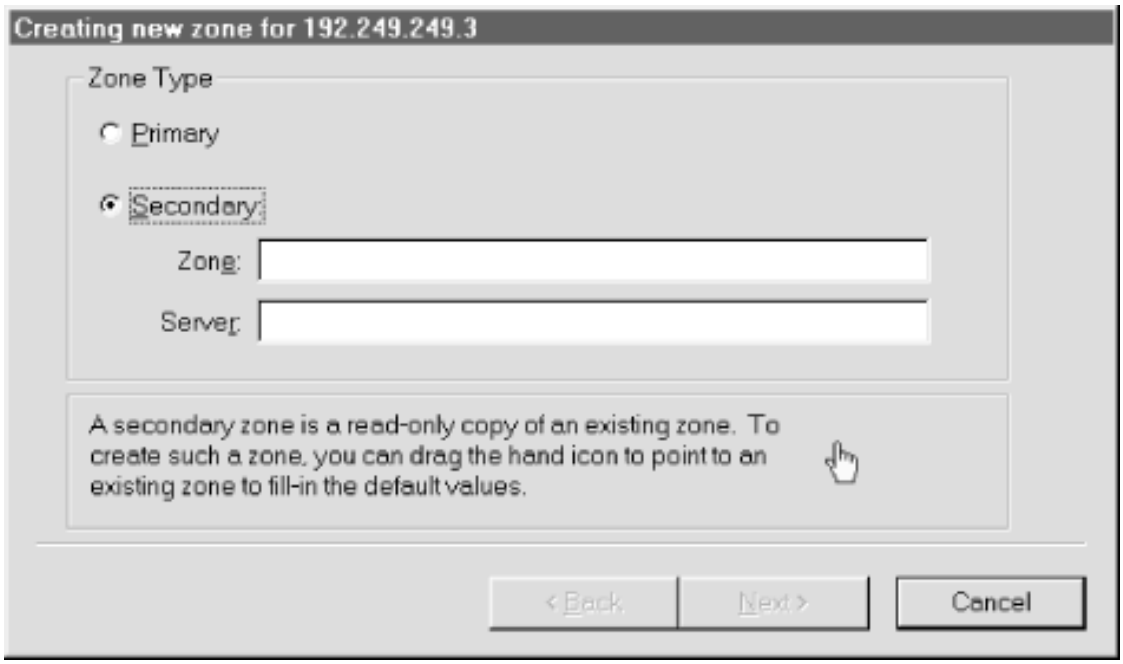
The first step in configuring a slave server is to add the server to DNS Manager's world view. Just as we did when configuring the primary master, select **DNS • New Server**, and then enter the IP address of the slave. In this case our slave will be *wormhole* with IP address 192.249.249.1. Of course, the DNS Server has to be installed and running on the slave-to-be for DNS Manager to be able to manage it.

After you've added the slave to DNS Manager's configuration, double-click the IP address of the primary master name server to list the zones it's authoritative for. Having this list of zones visible makes things easier in the next step.

Create a New Zone

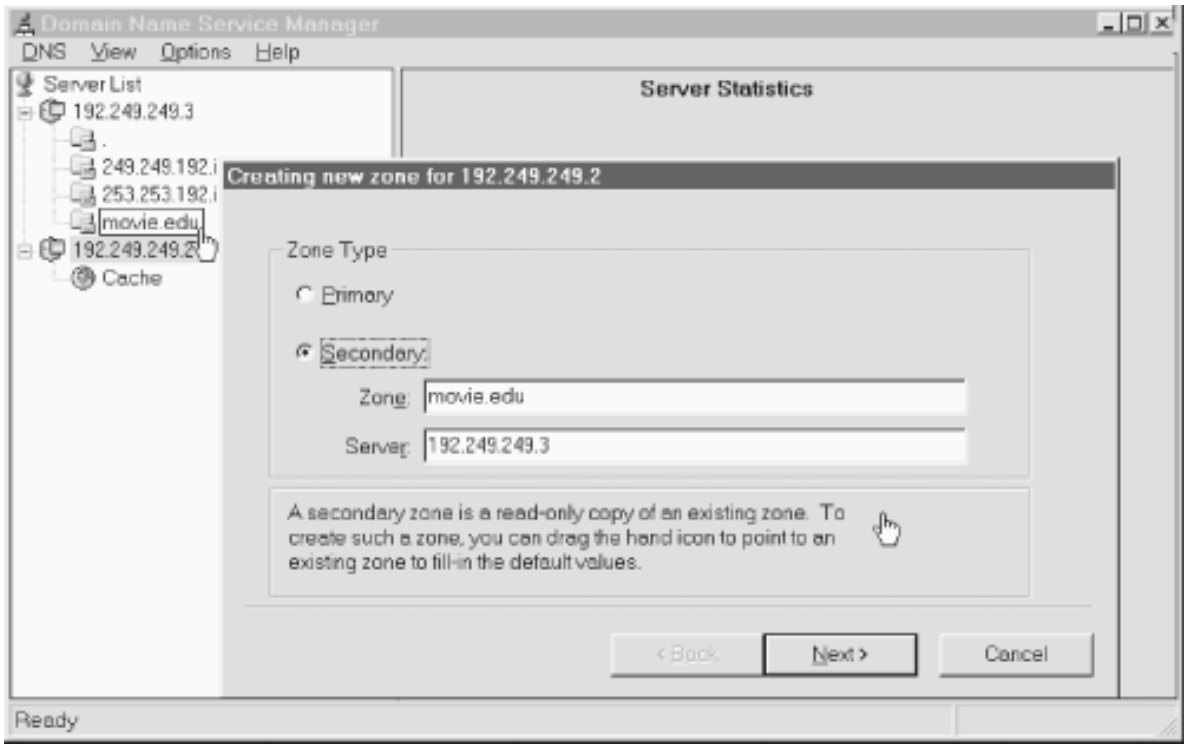
This new server will be a slave for every zone on the primary, so we'll have to go through the new zone process for each zone. Let's start with *movie.edu*. Select **DNS • New Zone**. This time, select **Secondary** (remember, this is synonymous with *slave*) in the resulting window, and you'll see something like Figure 4-18.

Figure 4-18. Creating a new secondary zone, first window, with Secondary selected



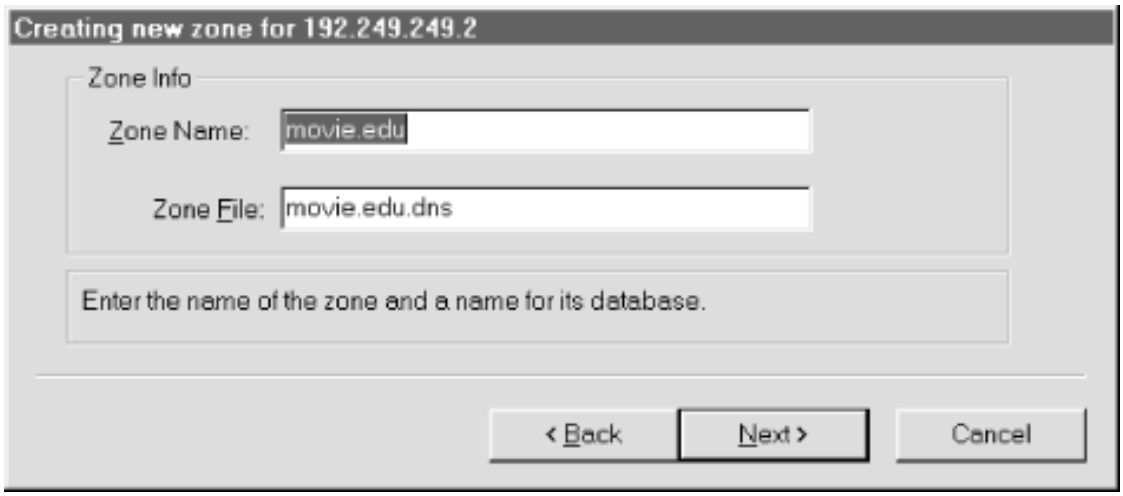
In the **Zone** field, enter the domain name of the zone to be a slave for (that is, *movie.edu*). In the **Server** field, enter the IP address of the primary master name server. You could type this information, or you can take advantage of a really slick shortcut offered by DNS Manager. Note the hand pointer in the window: you can specify the zone and server by dragging the hand to the zone on the primary that you want this server to be a slave for. Now you see why we wanted the primary's zones to be visible. Once the hand touches the zone, these fields are filled in. See Figure 4-19.

Figure 4-19. Creating a new secondary zone, first window, moving the hand pointer to the primary zone



Whether you enter the zone and server manually or use the hand pointer shortcut, click **Next** to get the next window as shown in Figure 4-20.

Figure 4-20. Creating a new secondary zone, second window



This is the same window you see when creating a primary zone. If you entered the zone and server manually in the previous window, only the zone is filled in here. You can still hit **Tab** to get the automatically generated zone filename conforming to the **.dns* convention. If you used the hand pointer in the previous window, both fields are filled in.

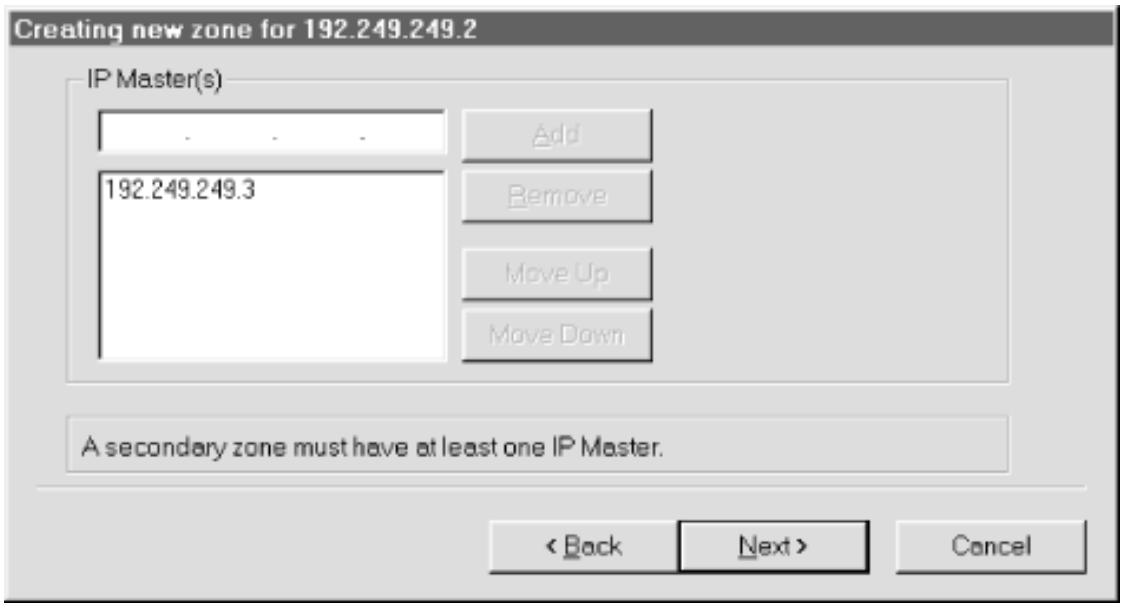
You're probably wondering why we're specifying a zone database file here--after all, this name server will be a slave for the *movie.edu* zone. It will load the zone from the primary, not from a file on its disk. On a slave, the zone file is not the definitive source of information for the zone as it is on the primary. Instead it's treated as a backup copy of the zone. After the slave does a zone transfer, it saves a copy of the zone in the backup file. The slave server reads the backup file on startup and later checks with the primary master server to see if the primary has a newer copy, instead of loading a new copy of the zone immediately. If the primary master server has a newer copy, the slave pulls it over and saves it in the backup file.

The backup file saves time and network bandwidth. When the slave starts up and the zone hasn't changed, it doesn't have to go through the time and expense of performing a zone transfer. Or suppose the primary server is down when the slave starts up. The slave would be unable to transfer the zone and therefore wouldn't function as a server for that zone until the primary server is up. With a backup copy, the slave has some data, although it might be slightly out of date. Since the slave does not rely on the master server always being up, the system is more robust.

The Microsoft DNS server requires slave zones to have a backup file. Those of you familiar with BIND know that backup files are optional, although we don't know anyone who doesn't use them.

When you're finished, click **Next**, and you'll see the window shown in Figure 4-21.

Figure 4-21. Creating a new secondary zone, third window



At this point, the process of creating a primary master zone and a slave zone really diverge. This is the screen where you specify where this name server will get the zone data from. In this example, we're making *wormhole* a slave for the *movie.edu* zone. We need to tell *wormhole* to load the zone from *terminator*, the primary master. "But wait," you say, "didn't we already do that two windows ago?" That's a valid question. For whatever reason, DNS Manager requires that you specify the primary's IP address in two places. In fact, on this screen you can specify multiple IP addresses. In advanced (and complicated) configurations, sometimes there are multiple primaries or multiple sources for a slave to get the zone information. DNS Manager supports those configurations. You could also just specify the IP address of another slave after that of the primary: in case the primary is down, this slave can load from another slave. Of course, Movie U. doesn't have another slave (yet).

For now, we just specify *terminator's* IP address, 192.249.249.3. Then click **Next**. The final window in the process is the same as when creating a primary zone: it just tells you that you're done now and asks you to click **Finish**. We'll omit showing it to you.

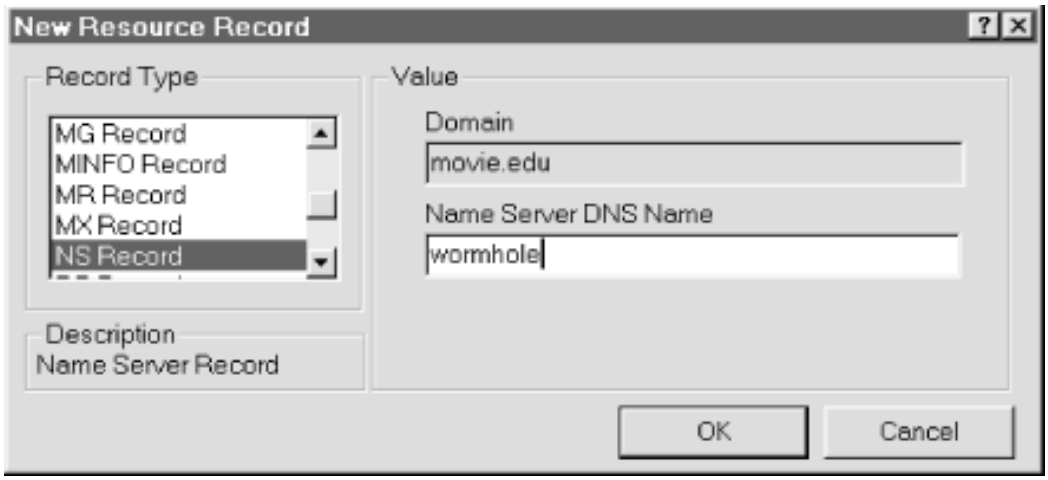
When you're done, the new slave immediately initiates a zone transfer to the primary to download the zone. Within a few seconds you should be able to double-click the slave's icon for the zone and see the records in the zone.

Add an NS Record for the New Slave Name Server

Your new slave won't be much good if the rest of the world doesn't know about it. As a general rule, when you add another name server for a zone, you also need to add an NS record for it. (We'll discuss the exceptions to this in Chapter 8, *Growing Your Domain*.)

You need to add an NS record on the zone's primary. (Remember that all changes to a zone are made on the primary and propagate automatically to the slaves. Don't get confused by the fact that DNS Manager lets you see all your name servers--you make the changes only to the zone's primary.) In our case, we need to add an NS record for *wormhole* to the *movie.edu* zone. So we highlight *movie.edu* under *terminator*, and select **DNS • New Record** as shown in Figure 4-22.

Figure 4-22. Creating an NS record



Don't Forget the in-addr.arpa Zones!

Now repeat this slave zone creation process with the *249.249.192.in-addr.arpa* and *253.253.192.in-addr.arpa* zones.

SOA Values

Remember this SOA record for the *movie.edu* zone?

```
@    IN     SOA terminator.movie.edu.administrator.movie.edu.(
                        1           ; serial number
                        3600        ; refresh
                        600         ; retry
                        86400       ; expire
                        3600      ) ; minimum TTL
```

We never explained what the values in between the parentheses were for.

The serial number applies to all the data within the zone. We chose to start our serial number at 1, a logical place to start. DNS Manager automatically increments the serial number in a zone's SOA record whenever you make a change to the zone. If you've maintained zone database files by hand, you might have encoded the date in the serial number--like 1997102301. This format is YYYYMMDDNN, where Y is the year, M is the month, D is the day, and NN is a count of how many times the zone data were modified that day. Unforunately, you can't use that convention when also using DNS Manager. It just updates the serial number by one each time a change is made and doesn't understand the date encoding.

When a slave name server contacts a primary master server for zone data, it first asks for the serial number on the data. If the slave's serial number is lower than the primary's, the slave's zone data are out of date. In this case, the slave pulls a new copy of the zone. As you might guess, if you ever modify the zone database files on the primary master by hand, you must increment the serial number, too. Updating zone database files is covered in Chapter 7.

The next four fields specify various time intervals in seconds:

refresh

The refresh interval tells the slave how often to check that its data are up to date. To give you an idea of the system load this feature causes, a slave will make one SOA query per zone per refresh interval. The default value generated by DNS Manager when the zone was created, one hour, is reasonably aggressive. Most users will tolerate a delay of half of a working day for things like name server data to propagate, when they are waiting for their new workstation to be operational. If you provide one-day service for your site, consider raising this value to eight hours. If your data don't change very often, or if all of your slaves

are spread over long distances (as the root name servers are), consider a value that is even longer: 24 hours.

retry

If a slave fails to reach the primary name server (s) after the refresh period (the hosts or hosts could be down), then it starts trying to connect every *retry* seconds. The retry interval is usually shorter than the refresh interval, but it doesn't have to be.

expire

If a slave fails to contact the primary server (s) for *expire* seconds, the slave expires its data. Expiring the data means the slave stops giving out answers about the data because the data are too old to be useful. Essentially, this field says: at some point, the data are so old that having *no* data is better than having stale data. We think Microsoft's default expire time of 86400 seconds (24 hours) is awfully short. Expire times on the order of a week are common--longer (up to a month) if you frequently have problems reaching your updating source. The expiration time should always be much larger than the retry and refresh intervals; if the expire time is smaller than the refresh interval, your slaves will expire their data before trying to load new data.

minimum TTL

TTL stands for *time to live*. This value applies to all the resource records in the zone database file. The name server supplies this TTL in query responses, allowing other servers to cache the data for the TTL interval. If your data don't change much, you might consider using a minimum TTL of several days. One week is about the longest value that makes sense. Again, the default value of 3600 seconds (one hour) is very short, which we don't recommend because of the amount of DNS traffic it causes.

What values you choose for your SOA record will depend upon the needs of your site. In general, longer times cause less loading on your systems and lengthen the propagation of changes; shorter times increase the load on your systems and speed up the propagation of changes. We find the following values work well for most sites; they're also a good starting point if you're not sure what values to use:

```
  10800 ;  Refresh         3 hours
   3600 ;  Retry           1 hours
2592000 ;  Expire         30 days
  86400 ;  Minimum TTL     1 day
```

Adding More Domains

Now that you have your name servers running, you might want to handle more zones. What needs to be done? Nothing special, really. Just use DNS Manager to select the appropriate server in the left pane, and then choose **DNS • New Zone**. Follow the instructions earlier in this chapter according to whether you are creating a primary or a slave (secondary) zone.

At this point, it's useful to repeat something we said in an earlier chapter. Calling a *given* name server a primary master name server or a slave name server is a little silly. Name servers can be authoritative for more than one zone. A name server can be a primary master for one zone and a slave for another. Most name servers, however, are either primary master for most of the zones they load or slave for most of the zones they load. So if we call a particular name server a primary master or a slave, we mean that it's the primary master or a slave for *most* of the zones it loads.

DNS • Properties

Let's finish this chapter with an explanation of the **DNS • Properties** selection. The **Properties** selection on the DNS menu is context sensitive. When selected, DNS Manager displays the properties of the resource record, zone, or server that is highlighted.

Resource Record Properties

Select a resource record on the right by single-clicking it. Then choose **DNS • Properties**. The window should look familiar: it's the same one you used to add the record. You can get the same effect by simply double-clicking the record, too.

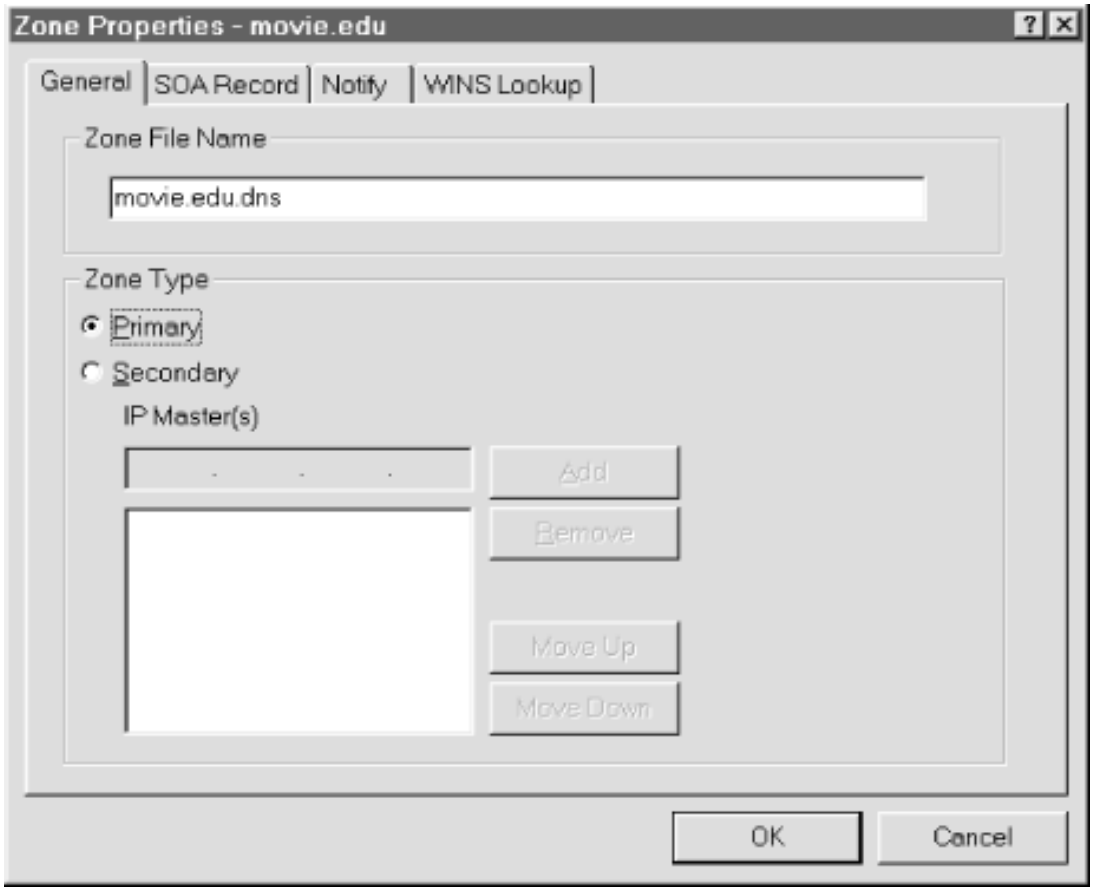
Zone Properties

The zone properties window is viewed by selecting a zone on the left and choosing **DNS • Properties**. Unlike resource record properties, some zone information can be changed only from this window. It has four tabs:

General

This window shows the name of the zone's database file as well as indicating if it's a primary or slave (secondary) zone. The type of the zone can be changed from primary to slave or vice versa. The window for the *movie.edu* zone is shown in Figure 4-23.

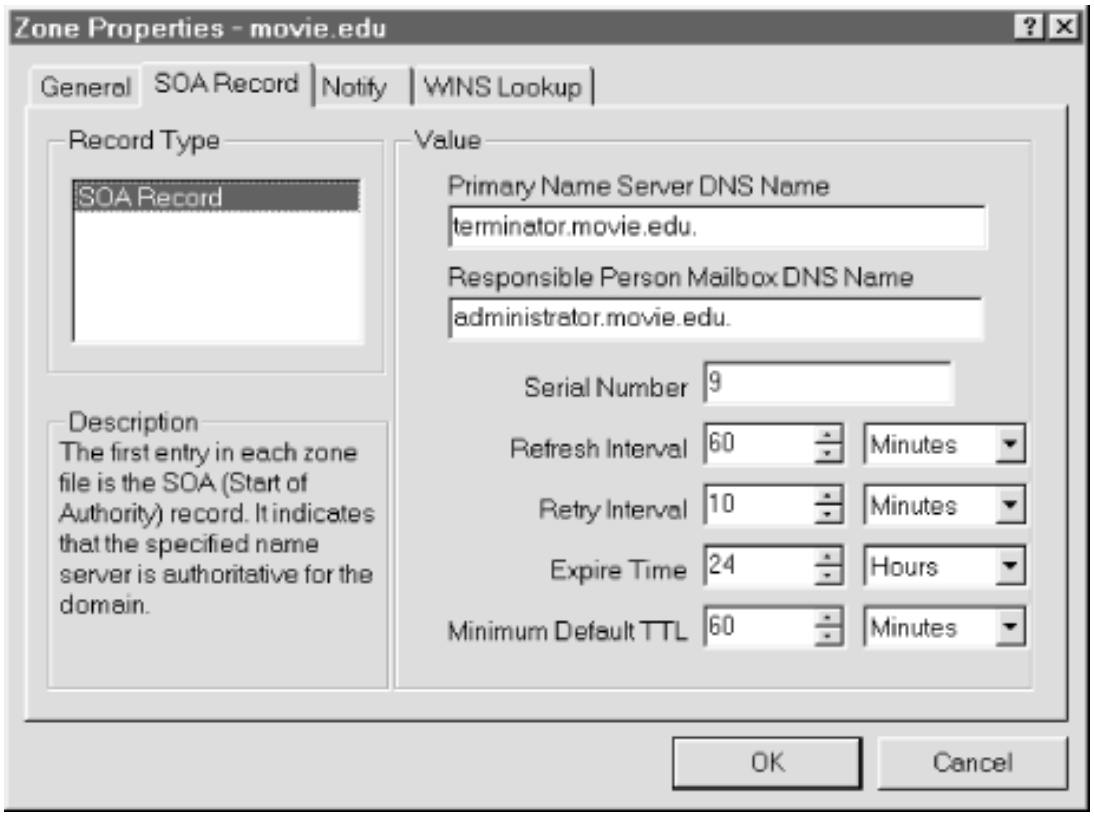
Figure 4-23. Zone Properties • General



SOA Record

This window shows the zone's SOA record. The display, shown in Figure 4-24, is no different than if you double-click the SOA record in the right panel.

Figure 4-24. Zone Properties • SOA Record



Notify

The Notify tab is covered in Chapter 10, *Advanced Features and Security*.

WINS Lookup

The WINS Lookup tab is also covered in Chapter 10.

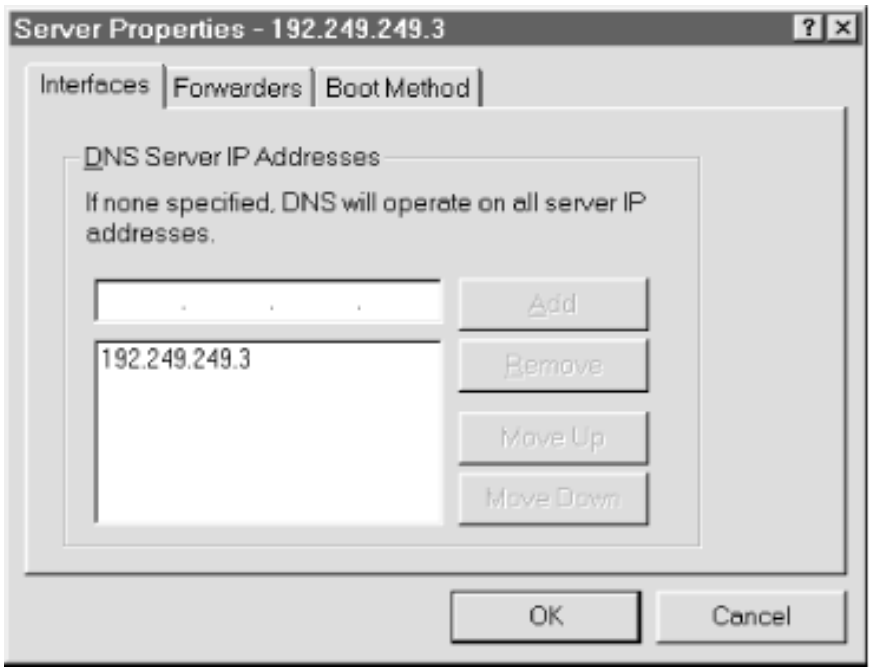
Server Properties

You can view the server properties by selecting a server on the left and choosing **DNS • Properties**. It has three tabs:

Interfaces

This window allows you to specify which interfaces the server will listen on for queries. If you have multiple interfaces (like for virtual web hosting), you might not need them all listed here. The default behavior is for the server to listen on all interfaces. The window is shown in Figure 4-25.

Figure 4-25. Server Properties • Interfaces



Forwarders

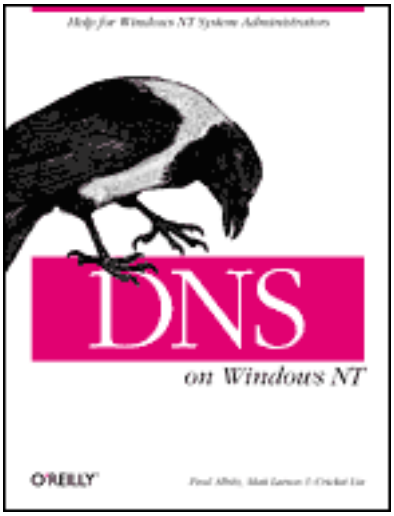
This tab is covered in Chapter 10.

Boot Method

This window displays where the name server is obtaining its configuration information (also referred to as its *boot method*). The default location of the configuration information is the Registry, but the server can also be configured to boot from a BIND Version 4 boot file. This window is "read only" and just displays the current boot method--changing the boot method means changing a Registry setting. More information about this topic is found in Appendix C, *Converting from BIND to the Microsoft DNS Server*.

What Next?

In this chapter, we showed you how to set up a primary master and a slave name server. There is more work to do to complete setting up your local domain: you need to modify your DNS data for email, configure the other hosts in your domain to use name servers, and you may need to start up more name servers. These topics are covered in the next few chapters.



DNS on Windows NT

By Paul Albitz, Matt Larson & Cricket Liu
1st Edition October 1998
1-56592-511-4, Order Number: 5114
348 pages, $34.95

Sample Chapter 10: Advanced Features and Security

In this chapter:

"What's the use of their having names," the Gnat said, "if they won't answer to them?"

In this chapter, we'll cover some of the Microsoft DNS Server's more advanced features and suggest how they might come in handy in your DNS infrastructure. (We do save some of the hard-core firewall material 'till the last chapter, though.)

DNS NOTIFY (Zone Change Notification)

Traditionally, slaves have used a polling scheme to determine when they need a zone transfer. The polling interval is called the *refresh time*. Other parameters in the zone's SOA record govern other aspects of the polling mechanism.

Wouldn't it be nice if the primary master name server could *tell* its slave servers when the information in a zone changed? After all, the primary master name server *knows* the data has changed: every time a zone is changed with DNS Manager, DNS Manager notifies the server, which immediately changes the zone in its memory. The primary's notification could come soon after the actual modification, instead of waiting for the refresh interval to expire.

RFC 1996 proposed a mechanism that would allow primary master servers to notify their slaves of changes to a zone's data. The Microsoft DNS Server implements this scheme, called DNS NOTIFY for short.

DNS NOTIFY works like this: when a primary master name server notices a change to data in a zone, it sends a special notification message to all of the slave servers for that zone. It uses the list of NS records in the zone to build the list of slave servers for the zone. The primary master removes the NS record corresponding to the name server listed in the first field in the zone's SOA record (which by convention lists the name of the primary master name server for the zone), as well as the local host. Removing those name servers prevents the primary master from sending a notification message to itself.

The special NOTIFY request is identified by its opcode in the query header. The opcode for most queries is QUERY. NOTIFY messages have a special opcode, NOTIFY. Other than that, the request looks very much like a query for the SOA record for the zone: it specifies the zone's domain name, class, and a type of SOA.

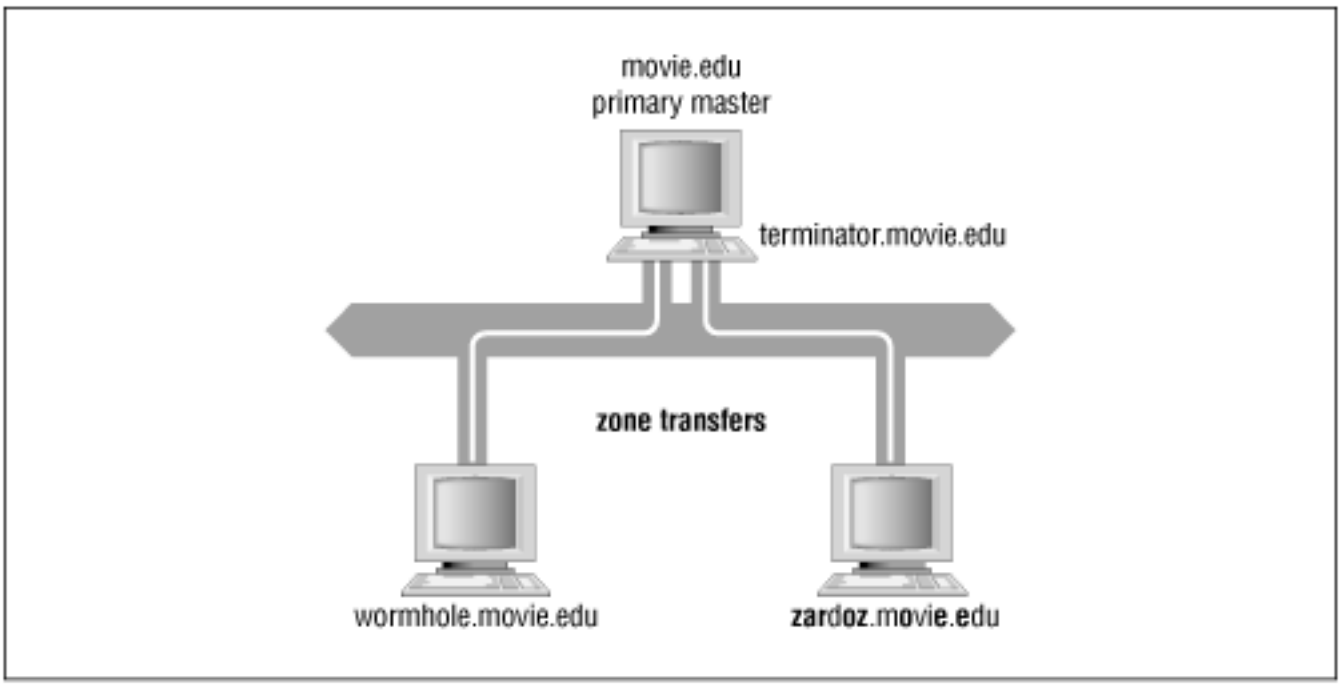
When a slave receives a NOTIFY request for a zone from one of its configured master name servers, it sends a NOTIFY response. The response tells the master that the slave received the NOTIFY request, so that it can stop sending NOTIFY messages for the zone. Then the slave proceeds just as if the refresh timer had expired: it queries the master server for the SOA record for the zone that the master claimed had changed. If the serial number is higher, the slave transfers the zone.

Why doesn't the slave simply take the master's word that the zone has changed? It's possible that a miscreant could forge NOTIFY requests to our slaves, causing lots of unnecessary zone transfers, amounting to a denial of service attack against our master server.

If the slave actually transfers the zone, RFC 1996 says that it should issue its own NOTIFY requests to the other authoritative name servers for the zone. The idea is that the primary master may not be able to notify all of the slave servers for the zone itself, since it's possible some slaves can't communicate directly with the primary master and so use another slave as their master. However, the Microsoft DNS Server doesn't implement this, and Microsoft DNS Server slaves don't send NOTIFY messages unless explicitly configured to do so.

Here's how that works in practice: on our network, *terminator.movie.edu* is the primary master for *movie.edu*, and *wormhole.movie.edu* and *zardoz.movie.edu* are slaves. See Figure 10-1.

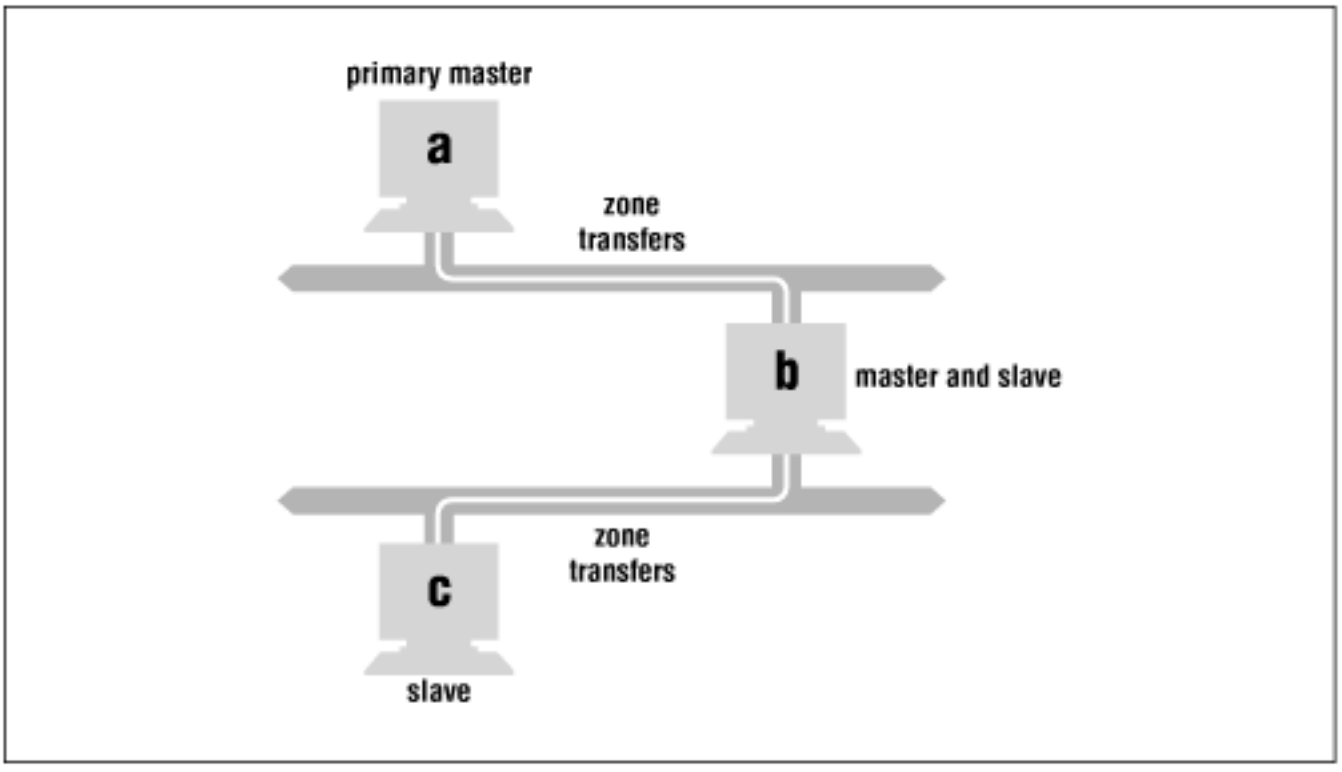
Figure 10-1. movie.edu zone transfer example



When we update *movie.edu* on *terminator*, *terminator* sends NOTIFY messages to *wormhole* and *zardoz*. Both slaves check to see whether *movie.edu*'s serial number has been incremented and, when they find it has, perform a zone transfer.

Let's also look at a more complicated zone transfer scheme. In Figure 10-2, *a* is the primary master name server for the zone and *b*'s master server, but *b* is *c* 's master server. Moreover, *b* has two network interfaces.

Figure 10-2. Complex zone transfer example



In this scenario, *a* notifies both *b* and *c* after the zone is updated. *b* checks to see whether the zone's serial number has been incremented and initiates a zone transfer. However, *c* ignores *a* 's NOTIFY message, because *a* is not *c* 's configured master name server (*b* is). If *b* is explicitly configured to notify *c*, then after *b*'s zone transfer completes, it sends *c* a NOTIFY message, which prompts *c* to check the serial number *b* holds for the zone.

Older BIND slave name servers, and other name servers that don't support NOTIFY, will respond with a "Not Implemented" (NOTIMP) error, wait until their refresh timers expire, and transfer the zone then. The Microsoft DNS Server just ignores the NOTIMP error.

NOTIFY is turned off by default and has to be enabled on a zone-by-zone basis. The controls for NOTIFY are accessed via the **Notify** tab on the **Zone Properties** window.

Figure 10-3. Zone Properties • Notify for movie.edu

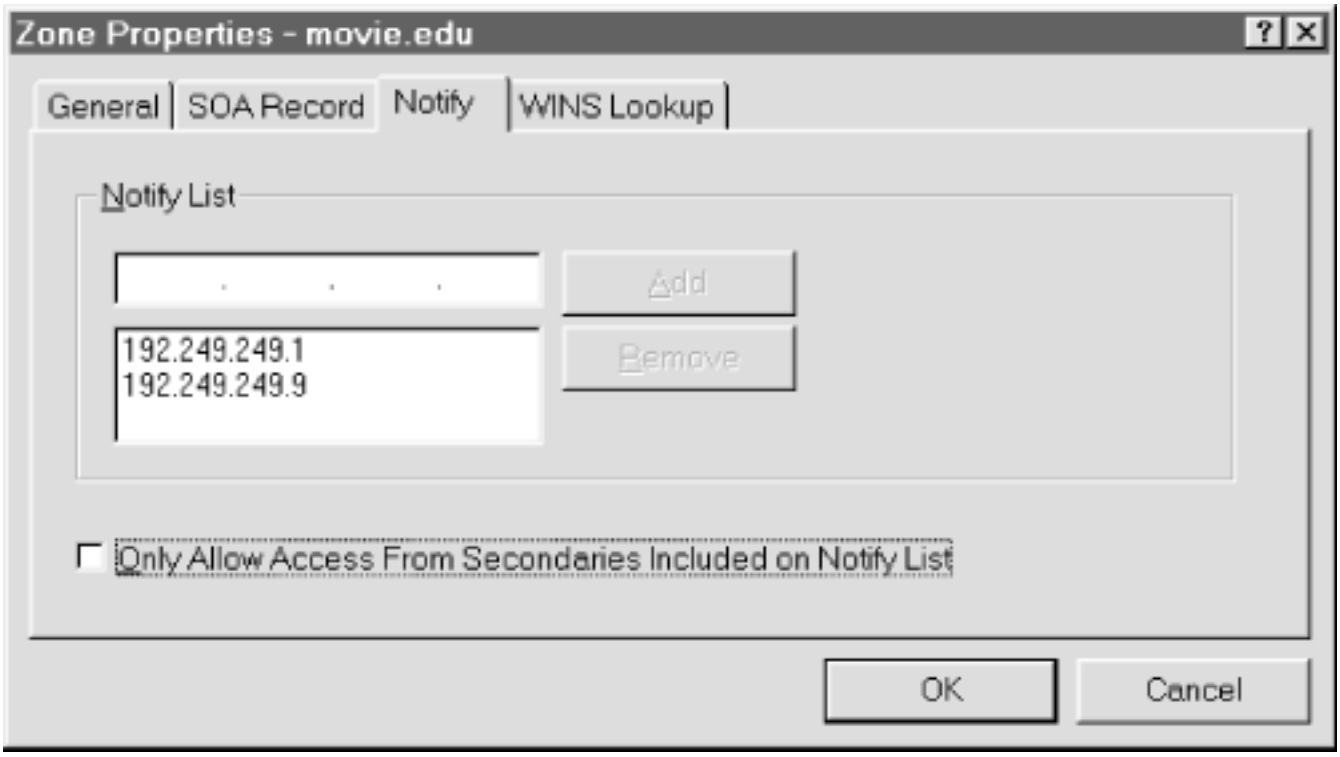


Figure 10-3 illustrates the Notify configuration for our old friend the *movie.edu* zone on the zone's primary, *terminator*. NOTIFY is enabled simply by listing at least one IP address in the **Notify List**. The addresses listed, 192.249.249.1 and 192.249.249.9, are for the slaves, *wormhole* and *zardoz*. You have to list the IP address of every slave you want to receive a NOTIFY message. Unlike the BIND name server, the Microsoft DNS Server does not use the zone's NS records to determine whom to send a NOTIFY message. You can also enable NOTIFY on a slave if other slaves perform zone transfers from it (a la the complex zone transfer scheme described previously).

We'll talk about the **Only Allow Access** checkbox in the "Securing Your Name Server" section, later in this chapter.

WINS Linkage

Our next topic requires a short detour into the world of Microsoft networking. Networks based on NetBT (NetBIOS over TCP) need to perform name resolution, too: hosts need a way to map NetBIOS names [1] to IP addresses. The way this name resolution works has evolved over time. In the early days, hosts would broadcast a query on the LAN to resolve a NetBIOS name. This forced all hosts to listen to every broadcast. Since broadcasts don't leave the local LAN, this method didn't allow name resolution beyond the local subnet. The next evolutionary step was the *LMHOSTS* file, which is just a list of NetBIOS names and IP addresses. Every host needed an *LMHOSTS* file to resolve names beyond the local subnet. This model didn't scale very well, either: it was tough to keep the *LMHOSTS* files up to date and distribute them. The introduction of DHCP essentially made basing a network's NetBIOS name resolution on *LMHOSTS* files impossible.

DHCP stands for Dynamic Host Configuration Protocol. A detailed description is beyond the scope of this book, [2] but suffice it to say that DHCP eliminates the requirement of configuring a static IP address on every one of your hosts. If those hosts support DHCP, they can contact a DHCP server when they boot to obtain an IP address and other configuration parameters, like the IP addresses of the default router, name servers, and WINS servers.

WINS, which stands for Windows Internet Naming Service, is a Microsoft invention introduced in Windows NT 3.5. The server component of WINS is an implementation of a NetBIOS Name Server as described in RFCs 1001 and 1002. The idea is nothing new--the RFCs date from early 1987. The function of a NetBIOS Name Server is simple: it maps NetBIOS names to IP addresses.

The name and IP address information in a WINS server comes from the various hosts on the network. Once a host sets its IP address using the value sent by a DHCP server, the host registers its name with the WINS server the DHCP server told it about. Actually, any modern NetBT host registers its name with a WINS server, regardless of how it obtained its IP address (dynamically from a DHCP server or statically from a user-input configuration). Modern NetBT hosts also know to contact a WINS server for NetBIOS name resolution, rather than relying on broadcasting or an *LMHOSTS* file.

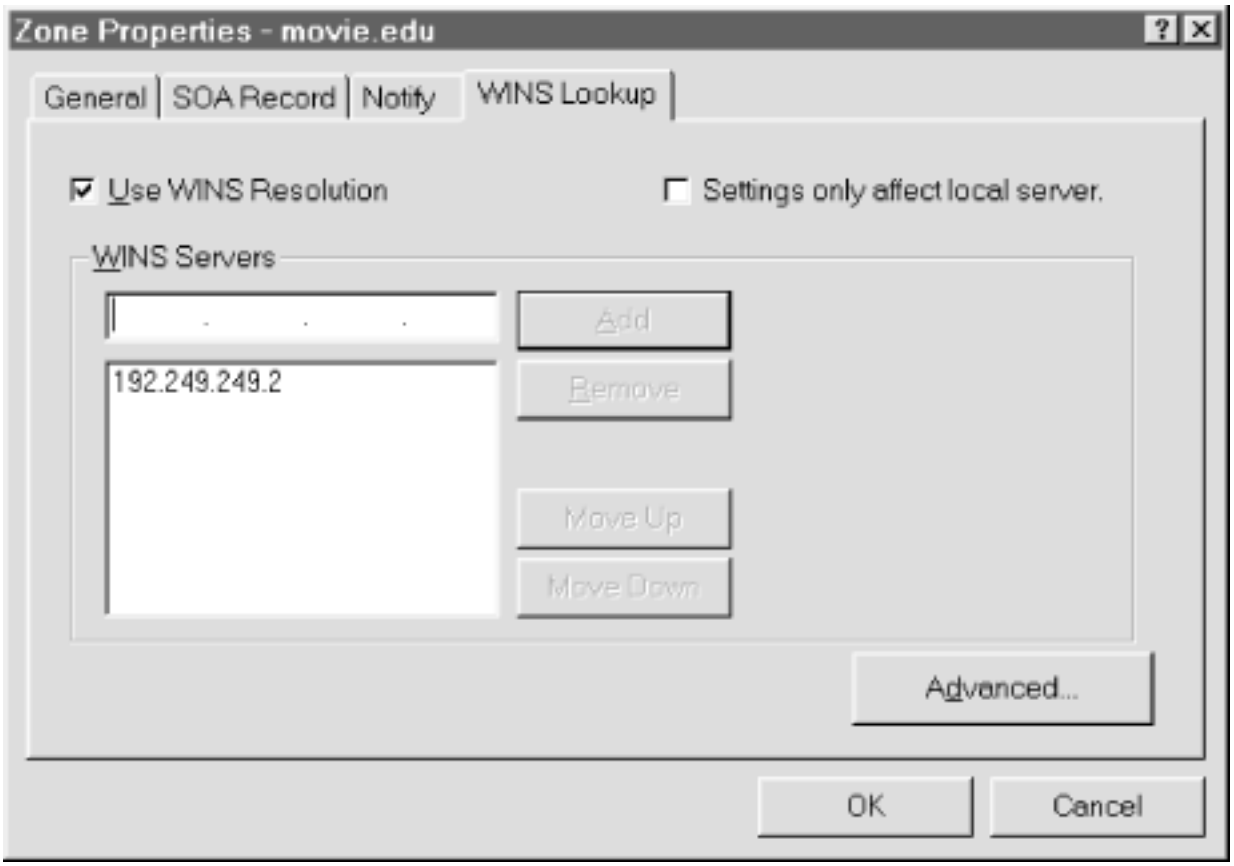
So where does DNS fit in to all this? The new name-to-IP address mappings generated when the DHCP server makes IP address assignments aren't visible to DNS. (The traditional DNS update model isn't suited to making incremental changes to zones on the fly, anyway, so it would be difficult to get this information into DNS.) Microsoft realized there would be some value to enabling a DNS Server to query a WINS server, which does know about names for dynamically assigned IP addresses. After all, a NetBIOS name in the WINS server is usually the same as a machine's host name (the first label of its fully qualified domain name in DNS), which is what would be in the DNS Server if there were an easy way to get it there. So a Microsoft DNS Server can be configured to ask a WINS server when it receives a query for a domain name that's not in its zone database.

You may be thinking that a name server contacting a WINS server is kind of silly--isn't there a way for name servers to know what the DHCP servers are doing directly? Yes, and it's coming in NT Version 5. DHCP servers will update name servers after every assignment using the new DNS dynamic update protocol. The importance of WINS in NT 5 is greatly reduced, too. NT 5 hosts will resolve NetBIOS names with DNS rather than WINS. WINS will still be required for a while to support older, legacy clients. But we digress.

Configuring WINS Lookup

WINS Lookup, as it's called, is enabled on a zone-by-zone basis using the **WINS Lookup** tab of the **Zone Properties** window. The DNS Server will forward queries for names it doesn't know about to a WINS server for those zones with WINS Lookup enabled. The WINS Lookup configuration for the *movie.edu* zone on the zone's primary, *terminator*, is shown in Figure 10-4.

Figure 10-4. Zone Properties • WINS Lookup for movie.edu

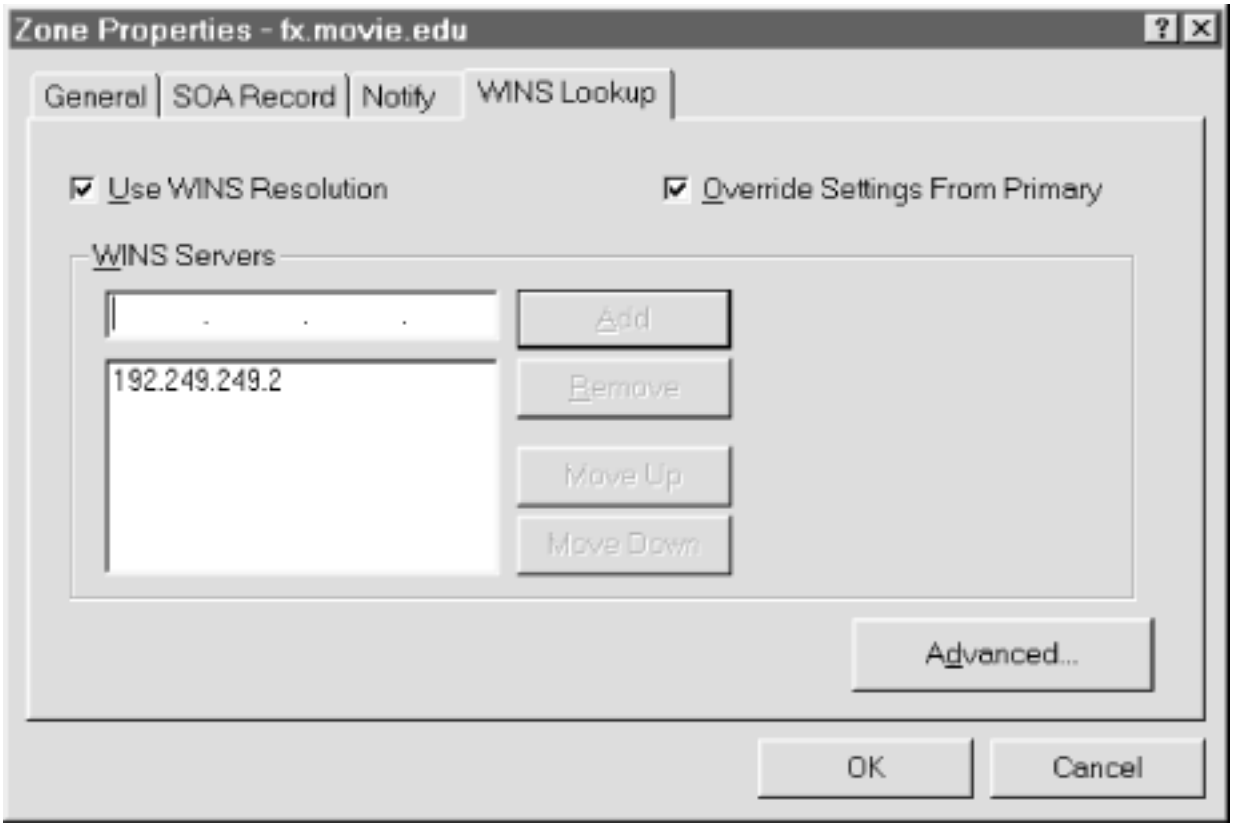


WINS Lookup is enabled by checking the **Use WINS Resolution** box. The IP addresses of up to five WINS servers go in the **WINS Servers** field. The DNS Server will try them in the order listed.

By default, the WINS Lookup configuration you establish on the primary master takes effect on the slaves as well. The primary master inserts a special WINS record that gets transferred with the rest of the zone to the slaves. If the slaves are Microsoft DNS Servers, they understand the WINS record and perform WINS lookups accordingly. If the slaves are BIND name servers, they complain about the unknown WINS record. You can suppress sending this WINS record to the slaves by checking **Settings only affect local server**.

The WINS Lookup window looks slightly different on the slave, as shown in Figure 10-5.

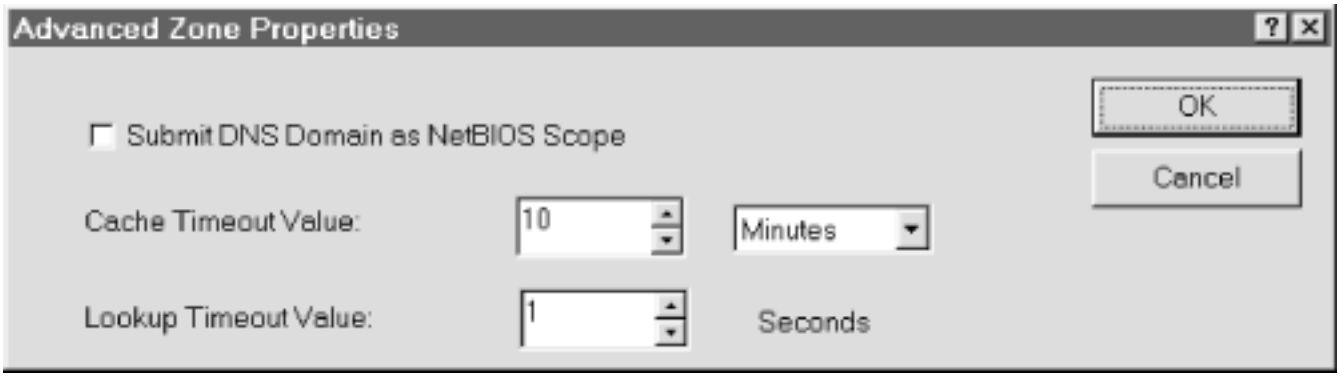
Figure 10-5. Zone Properties • WINS Lookup for movie.edu on a slave



Settings only affect local server has changed to **Override Settings From Primary**. Checking this box does just what you'd think: if the slave has received WINS lookup configuration from the primary, you can override it by checking this box.

The **Advanced** button produces the same window, shown in Figure 10-6, for either the primary master or a slave.

Figure 10-6. Zone Properties • WINS Lookup • Advanced



Submit DNS Domain as NetBIOS Scope is applicable only if you're using NetBIOS scope in your network. If you are, and if your scope names correspond to DNS domain names, you can pass along the DNS domain name (for example, *movie.edu*) along with the host name to the WINS server. Your NetBIOS scope names have to be consistent across your network for this to work. Another potential pitfall is that NetBIOS scope names are case sensitive, but DNS domain names are case insensitive.

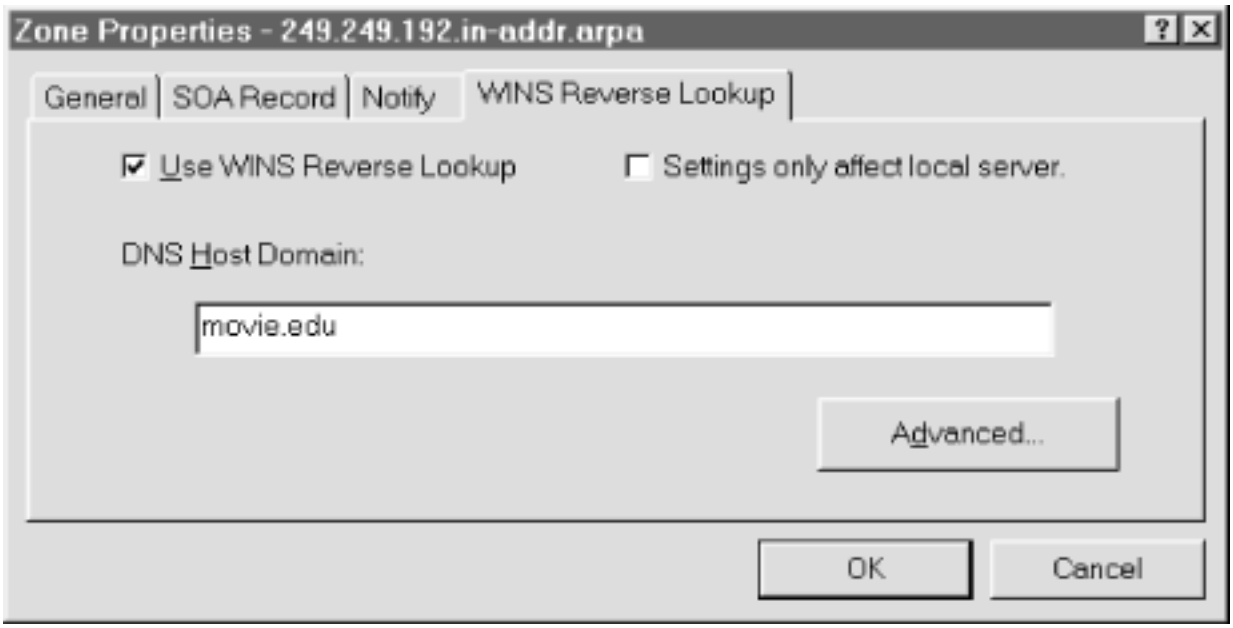
The next two values control timeouts. **Cache Timeout Value** controls how long the DNS Server will cache information it receives from the WINS server. The default value is 10 minutes. That value may

seem small, but it's a good choice: information in the WINS server is transient by nature, so you don't want the DNS Server to hold on to it for a long time. If it needs a name again, the DNS Server can just ask the WINS server for it. **Lookup Timeout Value** controls how long the DNS Server will wait after querying a WINS server. The default is one second.

You can enable WINS Lookup on *in-addr.arpa* zones, too. It's called WINS Reverse Lookup. It's implemented differently than plain WINS Lookup. When the name server receives a PTR query it can't answer and WINS Reverse Lookup is enabled for the zone, it sends a NetBIOS Adapter Status request directly to the IP address referenced by the PTR record. In other words, the name server asks the host directly what its name is. The name server can't ask a WINS server because lookups based on IP address aren't supported: you can't give a WINS server an IP address and get the corresponding NetBIOS name back. WINS servers have obviously never heard of *Jeopardy!* ("The host with IP address 192.249.249.3." "What is *terminator* ?")

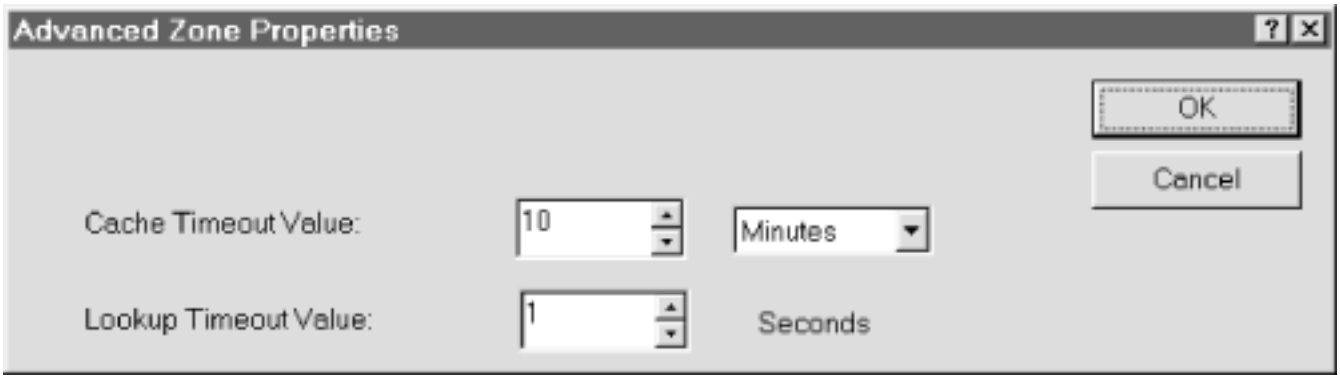
WINS Reverse Lookup is configured similarly to WINS Lookup--select the **WINS Reverse Lookup** tab of the **Zone Properties** window of any *in-addr.arpa* zone. The WINS Reverse Lookup configuration for the *249.249.192.in-addr.arpa* zone on the zone's primary, *terminator*, is shown in Figure 10-7.

Figure 10-7. Zone Properties • WINS Reverse Lookup for 249.249.192.in-addr.arpa



Use WINS Reverse Lookup enables the NetBIOS Adapter Status requests for unknown PTR records in this zone. **Settings only affect local server** has the same effect as its WINS Lookup counterpart. If you look in an *in-addr.arpa* zone database file, though, you'll see a WINS-R record instead of a WINS record. The **DNS Host Domain** field takes a DNS domain name that will be appended to the NetBIOS name returned by the host to form a fully qualified domain name. The **Advanced** button is very similar to the one for WINS Lookup (see Figure 10-8).

Figure 10-8. Zone Properties • WINS Reverse Lookup • Advanced



There's no reference to NetBIOS scope since it's not applicable here. **Cache Timeout Value** controls how long the DNS Server will cache the name received from the host. The default value is 10 minutes, just like WINS Lookup. **Lookup Timeout Value** controls how long the DNS Server will wait on the NetBIOS Adapter Status request. The default is one second.

Using WINS Lookup and WINS Reverse Lookup

What's WINS Lookup good for? In most networks, not a lot. The names that get resolved the most are the servers, and they usually have fixed IP addresses and thus static DNS entries. They're resolved directly in DNS, not via the WINS Lookup detour. Most networks don't have much peer-to-peer networking--your average desktop host usually doesn't offer network services, like a web server, name server, and so on. It's the need to reach those kind of network services that requires DNS name resolution to work for every host. (Sure, there's a lot of NetBIOS-based file and print sharing among desktop hosts, but that uses WINS natively.)

If you do need to support WINS Lookup in your network, a big problem with it is that the standard BIND name server doesn't support it. [3] Many people find they need WINS Lookup after they have a DNS infrastructure in place using BIND name servers. One option is to replace all those name servers with the Microsoft DNS Server and enable WINS Lookup. That's not realistic for most people. A better, but not perfect, option is to create a new subdomain for DHCP clients resolvable via WINS Lookup and delegate the subdomain to a set of Microsoft DNS Servers.

For example, let's say the folks running the domain *acmebw.com* suddenly find themselves with dozens of PCs doing peer-to-peer networking with DHCP-assigned IP addresses. Since they've already got a BIND infrastructure in place, they decide to create the domain *pcs.acmebw.com* for these PCs. (The domain name could be anything: *dhcp.acmebw.com*, *wins.acmebw.com*, *projectx.acmebw.com*, whatever.) They configure a couple Microsoft DNS Servers for this zone and enable WINS Lookup. Finally, they delegate to the *pcs.acmebw.com* zone from the *acmebw.com* zone.

In practice, we find WINS Reverse Lookup is much more useful. It's really nice to have complete reverse mapping information for your network in DNS. Network management applications can report names rather than IP addresses. Web servers can log usage statistics by name and make named-based authorization decisions, like only giving access to hosts in the *movie.edu* domain. Troubleshooting is easier, as well. Without WINS Reverse Lookup, the name server can't reverse-map dynamically assigned IP addresses. Of course, to use WINS Reverse Lookup, all the name servers for your *in-addr.arpa* zones need to support WINS Reverse Lookup.

System Tuning

While the default configuration values will work fine for most sites, yours may be one of the rare sites that needs some further tuning. The following tuning requires changes to the Registry. All DNS parameters referenced in this section are values of this Registry key:

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\DNS\Parameters

A complete listing of Registry settings for the DNS Server can be found in Appendix G, *Microsoft DNS Server Registry Settings*.

More Efficient Zone Transfers

A zone transfer, we said earlier, comprises many DNS messages sent end to end over a TCP connection. Traditional zone transfers put only a single resource record in each DNS message. That's a waste of space: you need a full header on each DNS message, even though you're only carrying a single record. It's like being the only person in a Chevy Suburban. A DNS message could carry many more records.

The Microsoft DNS Server understands a new zone transfer format that puts as many records as possible into a single DNS message. The resulting "many answers" zone transfer takes less bandwidth, because there's less overhead, and less CPU time, because less time is spent unmarshaling DNS messages.

The DNS Server uses the "many answers" format by default, which is fine if all your slaves can understand it. Older BIND name servers (prior to Version 4.9.4) can't cope with this format and require the traditional one. Fortunately, you can tell the DNS Server to use the traditional method by changing the *BindSecondaries* Registry value. When set to one, the server sends traditional zone transfers to satisfy older BIND servers. The default value is one, but that doesn't affect zone transfers between two Microsoft DNS Servers. They recognize each other, and the master uses the "many answers" format to the slave.

You should change this value only if you have no BIND slaves, or if all your BIND slaves are running Version 4.9.4 or later.

Cleaning Up the Database

The Registry value *CleanupInterval* controls how often the DNS Server performs housekeeping on its memory database of cached records. The default value is 900 seconds (15 minutes), and valid values range from 600 seconds (10 minutes) to 86,400 seconds (24 hours). At every cleanup interval, the name server examines its cache for expired records and removes them. (Expired records are simply ones that have been kept longer than the time to live value set by the name server they came from.) It also verifies that the root name servers are reachable at this interval. If it can't reach any, it reloads the cache file and follows the usual algorithm to obtain a list of root name servers.

We recommend leaving this value at the default. You might want to increase it, though, if your name server is especially busy and running the cleanup interval too often bogs it down. You might also raise it if you want records to persist in the cache longer than their original TTL. Remember, though, that the DNS administrators set the TTL according to their local circumstances, so there might be a very good

reason not to keep data longer than the TTL.

One more note about *CleanupInterval* : if the server receives a response containing an NS record (or its corresponding A record) that's already in the cache, it checks the TTL of the cached record. If the TTL is less than CleanupInterval, the server updates the cached record's TTL with the TTL from the record it just received, essentially refreshing the cached record.

Name Server Address Sorting

When you are contacting a host that has multiple network interfaces, using a particular interface may give you better performance. If the multihomed host is local and shares a network (or subnet) with your host, one of the multihomed host's addresses is "closer."

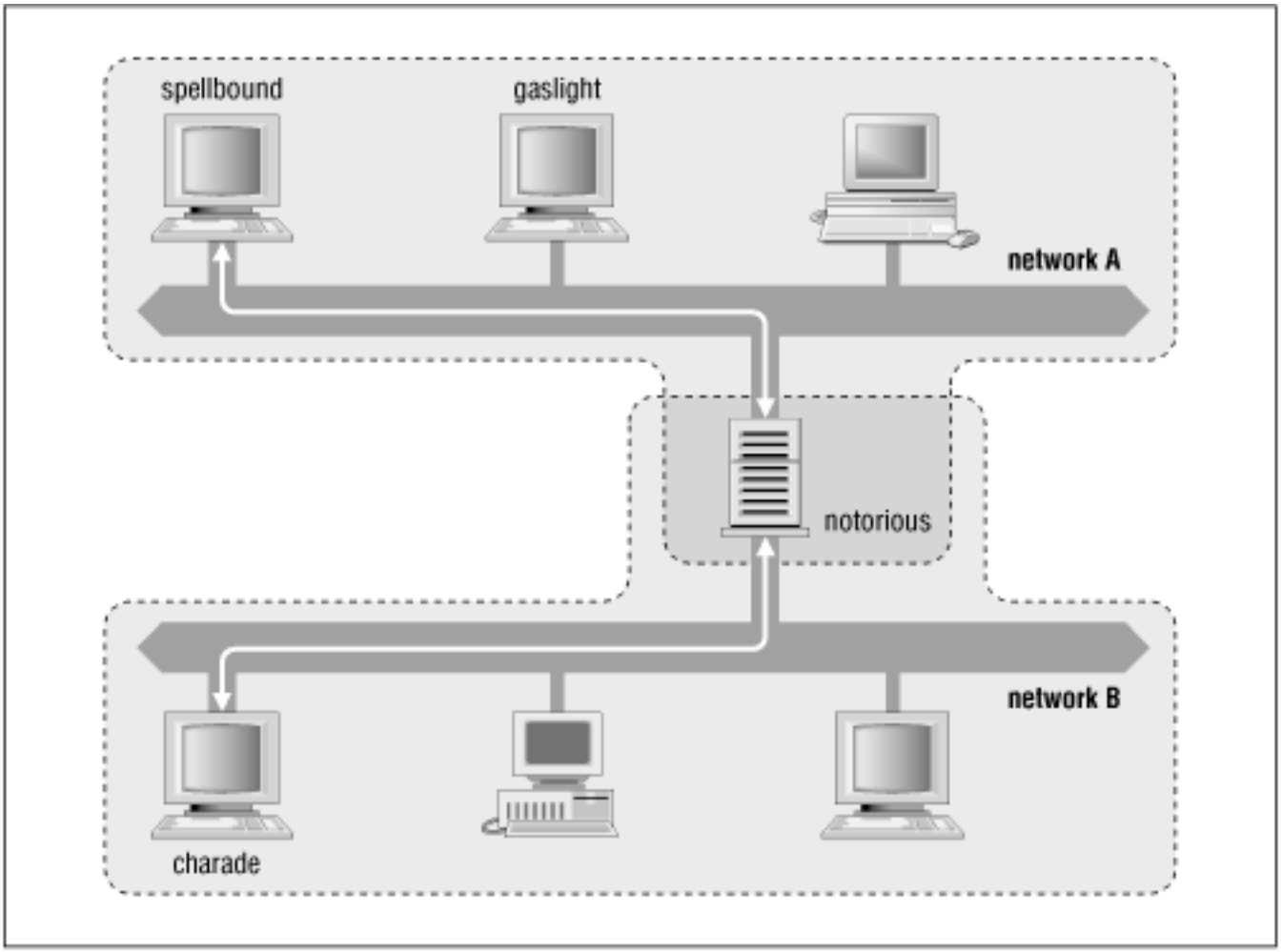
Suppose you have an FTP server on two networks, cleverly called network A and network B, and hosts on both networks access the server often. Hosts on network A will experience better performance if they use the server host's interface to network A. Likewise, hosts on network B would benefit from using the server host's interface to network B as the address for their FTP client.

In Chapter4, *Setting Up The Microsoft DNS Server*, we mentioned that the Microsoft DNS Server returns all the addresses for a multihomed host. There was no guarantee of the order in which a DNS server would return the addresses, so we assigned aliases (*wh249* and *wh253* for *wormhole*) to the individual interfaces. If one interface was preferable, you (or more realistically, a DNS client) could use an appropriate alias to get the correct address. You *can* use aliases to choose the "closer" interface, but because of address sorting, they are not always necessary.

The post-Service Pack 3 hotfixed version of the Microsoft DNS Server supports address sorting; earlier versions do not. The server sorts addresses, by default, if one condition holds: if the host that sent the query to the name server *shares* a network with the name server host (for example, both are on network A), then the DNS Server sorts the addresses in the response. How does the DNS Server know when it shares a network with the querier? When it starts up, it finds all the interface addresses of the host it is running on. The DNS Server extracts the network numbers from these addresses to create the default sort list. When a query is received, the DNS Server checks if the sender's address is on a network in the default sort list. If it is, then the query is local, and it sorts the addresses in the response.

In Figure 10-9, assume that a Microsoft DNS Server is on *notorious*. The name server's default sort list would contain network A and network B. When *spellbound* sends a query to *notorious* looking up the addresses of *notorious*, it will get an answer back with *notorious* 's network A address first. That's because *notorious* and *spellbound* share network A. When *charade* looks up the addresses of *notorious*, it will get an answer back with *notorious* 's network B address first. Both hosts are on network B. In both of these cases, the name server sorts the addresses in the response because the hosts share a network with the name server host. The sorted address list has the "closer" interface first.

Figure 10-9. Communicating with a local multihomed host



Let's change the situation slightly. Suppose the name server is running on *gaslight*. When *spellbound* queries *gaslight* for *notorious* 's address, *spellbound* will see the same response as in the last case because *spellbound* and *gaslight* share network A, which means that the name server will sort the response. However, *charade* may see a differently ordered response, since it does not share a network with *gaslight*. The closer address for *notorious* may still be first in the response to *charade*, but only because of luck, not name server address sorting. In this case, you'd have to run an additional name server on network B for *charade* to benefit from the DNS Server's default address sorting.

As you can see, you benefit by running a name server on each network; not only is your name server available if your router goes down, it also sorts addresses of multihomed hosts. Because the name server sorts addresses, you do not need to specify aliases to get the best response.

There's a small catch with the DNS Server's address sorting: it disables round robin. (Round robin is explained in the section "Load Sharing Between Mirrored Servers *," later in this chapter.*) In the post-SP3 hotfixed server, address sorting is enabled by default and round robin is disabled. If you want round robin and can live without address sorting (unfortunately they're mutually exclusive), you can disable address sorting with the *LocalNetPriority* value. Set it to zero to disable address sorting and enable round robin. Note, though, that this value doesn't exist in the Registry by default. You'll need to add it before you can change its value to zero.

Building Up a Large Sitewide Cache with Forwarders

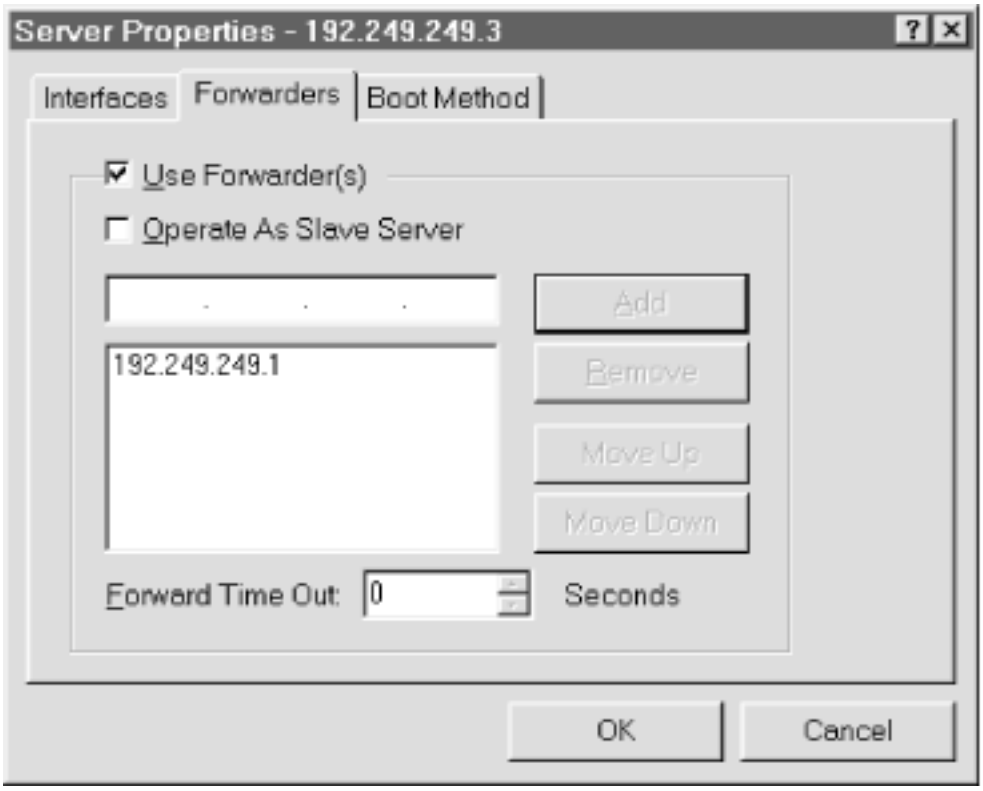
Certain network connections discourage sending large volumes of traffic off site, either because the network connection is pay-per-packet or because the network connection is a slow link with a high delay, like a remote office's satellite connection to the company's network. In these situations, you'll want to limit the offsite DNS traffic to the bare minimum. The Microsoft DNS Server has a feature to do this called *forwarding*.

If you designate one or more servers at your site as forwarders, all the offsite queries are sent to the forwarders first. The idea is that the forwarders handle all the offsite queries generated at the site, building up a rich cache of information. For any given query in a remote domain, there is a high probability that the forwarder can answer the query from its cache, avoiding the need for the other servers to send packets off site. Nothing special is done to these servers to make them forwarders; you modify all the *other servers* at your site to direct their queries through the forwarders.

A primary master or slave name server's mode of operation changes slightly when it is directed to use a forwarder. If the requested information is already in its database of authoritative data and cache data, it answers with this information; this part of the operation hasn't changed. However, if the information is not in its database, the name server will send the query to a forwarder and wait a short period for an answer before resuming normal operation and contacting the remote servers itself. What the name server is doing that's different is sending a *recursive* query to the forwarder, expecting it to find the answer. At all other times, the name server sends out *nonrecursive* queries to other name servers and deals with responses that only refer to other name servers.

Forwarding is serverwide, not zone specific: a server is either forwarding or it isn't. It's configured by selecting the **Forwarders** tab on the **Server Properties** window. Figure 10-10 shows how a *movie.edu* name server would be configured to use forwarders, assuming *wormhole* and *terminator* are the site forwarders. (Remember, forwarding is configured on every name server *except* the forwarders themselves-- *wormhole* and *terminator* in this case.)

Figure 10-10. Server Properties • Forwarders



Use Forwarder (s) enables forwarding on this name server. You can specify up to five forwarders. This name server will forward to them in the order they're listed, using a default timeout of five seconds per forwarder--that is, if the first forwarder doesn't respond within five seconds, try the next, wait five more seconds, try the next, and so on. The forwarding timeout can be changed with--surprise!--the **Forward Time Out** field. This value is stored in a Registry value, *ForwardingTimeout*, which you can also change. (The list of forwarders is stored in the *Forwarders* value.) We'll talk about **Operate As Slave Server** in the next section.

When you use forwarders, try to keep your site configuration simple. You *can* end up with configurations that are really twisted.

- Avoid having "mid-level" servers forward packets (that is, avoid configuring forwarding on your mid-level name servers). Mid-level servers mostly refer name servers to subdomain name servers. If they have been configured to forward packets, do they refer to subdomain name servers, or do they contact the subdomain name server to find out the answer? Whichever way it works, you're probably making your site configuration too hard for mere mortals (and subdomain administrators) to understand.
- Avoid chaining your forwarders. Don't configure server *a* to forward to server *b*, and configure server *b* to forward to server *c* (or worse yet, back to server *a*).

A More Restricted Name Server

You may want to restrict your name servers even further--stopping them from even *trying* to contact an offsite server if their forwarder is down or doesn't respond. You can do this by making the server a *forwarding server in slave mode* or just *slave server* for short. The terminology gets really confusing, unfortunately: this slave has nothing to do with zone transfers and primary master servers. A name server

can be a primary master, slave, or caching-only server and still be forwarding in slave mode. Later versions of the BIND name server eliminate the confusion by calling this a *forward only* server.

A slave server is a variation on a server that forwards. It still answers queries from its authoritative data and cache data. However, it relies *completely* on its forwarders; it *doesn't* try to contact other servers to find out information if the forwarders don't give it an answer. You can turn a forwarding server into a slave server by checking the **Operate As Slave Server** box in the **Forwarders** configuration window (see Figure 10-10).

The slave server contacts each forwarder only once, and it waits a short time for the forwarder to respond. Listing the forwarders multiple times directs the forward-only server to *retransmit* queries to the forwarders, and increases the overall length of time that the forward-only name server will wait for an answer from forwarders. With a slave server, you might want to consider listing the forwarders' IP addresses more than once for redundancy: If the first query to a forwarder is lost, the second might still get through and get answered.

However, you must ask yourself if it *ever* makes sense to use a slave server. A slave server is completely dependent on the forwarders. You can achieve much the same configuration (and dependence) by not running a slave server at all; instead, configure your hosts' resolvers to point to the forwarders you were using. Thus, you are still relying on the forwarders, but now your applications are querying the forwarders directly instead of having a slave name server query them for the applications. You lose the local caching that the forward only server would have done, and the address sorting, but you reduce the overall complexity of your site configuration by running fewer "restricted" name servers.

A Nonrecursive Name Server

By default, resolvers send recursive queries, and name servers do the work required to answer recursive queries. (If you don't remember how recursion works, refer to Chapter 2, *How Does DNS Work?*) In the process of finding the answer to recursive queries, the name server builds up a cache of nonauthoritative information about other zones.

In some circumstances, it is *undesirable* for name servers to do the extra work required to answer a recursive query or to build up a cache of data. The root name servers are an example of these circumstances. The root name servers are so busy that they should not be spending the extra effort to recursively find the answer to a request. Instead, they send a response based only on the authoritative data they have. The response may contain the answer, but it is more likely that the response contains a referral to other name servers. And since the root servers do not support recursive queries, they do not build up a cache of nonauthoritative data, which is good because their cache would be huge. [4]

You can induce the Microsoft DNS Server to run as a nonrecursive name server by setting the *RecursionNo* Registry value to true. By default the name server supports recursion, and this value is false.

If you choose to make one of your servers nonrecursive, do not configure any of your hosts' resolvers to use it. While you can make your name server nonrecursive, there is no corresponding option to make your resolver work with a nonrecursive name server. [5]

You can list a nonrecursive name server as one of the servers authoritative for your zone data (that is, you can tell a parent name server to refer queries about your zone to this server). This works because name servers send nonrecursive queries between themselves.

Do not list a nonrecursive name server as a forwarder. When a name server is using another server as a forwarder, it sends the query to the forwarder as a recursive query instead of a nonrecursive query.

Securing Your Name Server

Compared to a modern BIND name server, the Microsoft DNS Server is short on security features, but you do have some options. We'll discuss how to prevent unauthorized zone transfers from your servers and how to "lock down" a name server directly connected to the Internet.

Preventing Unauthorized Zone Transfers

It's important to ensure that only your real slave name servers can transfer zones from your name server. Users on remote hosts that can query your name server's zone data can only look up data (for example, addresses) for hosts whose domain names they already know, one at a time. Users who can start zone transfers from your server can list all of the hosts in your zones. It's the difference between letting random folks call your company's switchboard and ask for John Q. Cubicle's phone number and sending them a copy of your corporate phone directory.

Remember that list of name servers on the **Notify** window of the **Zone Properties** window (Figure 10-3)? It has two uses. The first tells the name server which slaves to send NOTIFY messages to. The second is security related. If you check the **Only Allow Access From Secondaries Included on Notify List** box, the server limits which servers can perform zone transfers: only authorized slaves on the Notify List can initiate a zone transfer.

For a primary master name server accessible from the Internet, you definitely want to limit zone transfers to just your slave name servers. You probably don't need to restrict zone transfers on name servers inside your firewall, unless you're worried about your own employees listing your zone data.

Delegated Name Server Configuration

Some of your name servers answer nonrecursive queries from other name servers on the Internet because your name servers appear in NS records delegating your zones to them. We'll call these name servers *delegated* name servers.

You can take special measures to secure your delegated name servers. But first, you should make sure that these servers don't receive any recursive queries (that is, you don't have any resolvers configured to use these servers, and no name servers use them as forwarders). Some of the precautions we'll take--like disabling recursive queries--preclude your resolvers from using these servers.

Once you know your name server only answers queries from other name servers, you can turn off recursion. This eliminates a major vector of attack: the most common spoofing attacks involve inducing the target name server to query name servers under the hacker's control by sending the target a recursive query for a domain name in a zone served by the hacker's servers. Disabling recursion is described in the

section "A Nonrecursive Name Server," earlier in this chapter. You should also restrict zone transfers of your zones to known slave servers, as described in the previous section "Preventing Unauthorized Zone Transfers."

Load Sharing Between Mirrored Servers

The Microsoft DNS Server has a feature called *round robin*, named after the equivalent feature in the BIND name server. The server rotates address records for the same domain name between responses. For example, if the domain name *foo.bar.baz* has three address records for IP addresses 192.1.1.1, 192.1.1.2, and 192.1.1.3, the round robin feature causes the name server to give them out first in the order

```
192.1.1.1 192.1.1.2 192.1.1.3
```

then in the order

```
192.1.1.2 192.1.1.3 192.1.1.1
```

and then in the order

```
192.1.1.3 192.1.1.1 192.1.1.2
```

before starting over again with the first order, and repeating the rotation *ad infinitum*.

The functionality is enormously useful if you have a number of equivalent network resources, like mirrored FTP servers, web servers, or terminal servers, and you'd like to spread the load among them. You establish one domain name that refers to the group of resources, configure clients to access that domain name, and the name server inverse-multiplexes the accesses between the IP addresses you list.

It's a good idea to reduce the records' time to live, too. This ensures that, if the addresses are cached on an intermediate name server that doesn't support round robin, they'll time out of the cache quickly. If the intermediate name server looks up the name again, your authoritative name server can round robin the addresses again.

Note that this is really load sharing, not load balancing, since the name server gives out the addresses in a completely deterministic way, without regard to the actual load or capacity of the servers servicing the requests. In our example, the server at address 192.1.1.3 could be a 486DX33 running Linux, and the other two servers HP9000 K420s, and the Linux box would still get a third of the queries.

---

1. A host's NetBIOS name is simply the name it's known by for NetBT networking purposes. NetBIOS names are limited to one label of up to 15 characters (that is, no multiple label names like DNS domain names). On NT systems the NetBIOS name is set on the **Network Control Panel** window, **Identification** tab. A host's NetBIOS name need not be the same as the host name portion of its fully qualified domain name in DNS.

2. But see another book from O'Reilly & Associates, *TCP/IP Network Administration* by Craig Hunt.

3. MetaInfo has ported BIND to NT and added WINS Lookup and WINS Reverse Lookup. See *http://www.metainfo.com*.

4. Note that a root name server wouldn't normally receive recursive queries, unless a name server's

administrator configured it to use a root server as a forwarder, or a host's administrator configured its resolver to use the root server as a name server, or a user pointed *nslookup* at the root server.

5. In general. Clearly, programs designed to send nonrecursive queries, or ones that can be configured to send nonrecursive queries, like *nslookup*, would still work.
