*OpenStep Journal*, Spring 1995 (Volume 1, Issue 1).


# The Village Smythy

*Written by* **Alan M. Marcum**

*Under the spreading chestnut-tree,*
*The village smithy stands¼*
*ÐHenry Wadsworth Longfellow*

## THE BIG PICTURE

When you start a UNIX-based computer, lots of ªstuffº has to be done. The kernel has to be found and loaded, kernel structures allocated, some drivers loaded, devices located and initialized, and file systems mounted. Then you're ready to run ªsingle-user,º a mode in which normal system and network services aren't available, the NEXTSTEP Window Server hasn't been started, and you must use an ancient, line-oriented terminal interface.

*What happens during the low-level portions of network startup under NEXTSTEP? How does automatic address configuration work? Why does automatic host name configuration sometimes not work? What happens on the network during startup? And what exactly is a* configuration server*? In each issue, this column explores such details of network startup.*

But if you really want to *use* the computer, yet more work has to be done. More drivers must be loaded, the network interface configured, various system servicesÐfrom a ªdirectory assistanceº service to network management to data cachesÐrun, more file systems mounted, and yet more system services run. And then, finally, for a machine running NEXTSTEP, a login window appears. (For other OpenStep-compatible systems, you first somehow log in and then start OpenStep.)

That's the big-picture view of startup. And it's good enough for many people. But, for those who want more details, let's get out the magnifying glasses and the microscopes. We'll focus on what it takes to get the networking portions of the system up and running, from the time the network interface is configured (**rc.net** and **ifconfig**) until the time that network administrative information is available to user-level processes (through **lookupd**).

In this column, we'll cover the configuration of the network interface, including the Internet address, network mask, and broadcast address, and the determination of a host name. Future columns will examine the rest of the process.

*These steps are specific to NEXTSTEP Release 3.3. Startup for non-NeXT OpenStep implementations may be very different, and startup of other NEXTSTEP releases may be slightly different.*

# THE STEPS

Let's first examine the steps we'll follow, as specified in **/etc/rc**. If we wade through the comments and the conditional statements, the first three things of relevance done in **/etc/rc** are these:

```
/usr/etc/nmserver -nonet
```

```
/usr/etc/driverLoader a
sh /etc/rc.net -h
```

***KYBBLES AND BYTS***

***Here's the key to the abbreviations used in this article:***

- *ARPÐaddress resolution protocol*
- *DLCÐdata link control*
- *DNSÐdomain name system*
- *ICMPÐInternet control message protocol*
- *IPÐInternet protocol*
- *IPCÐinterprocess communication*
- *NISÐnetwork information system*
- *TCPÐtransmission control protocol*
- *UDPÐuser datagram protocol*

*For additional background information on Internet addresses or the Internet protocols discussed here, see the entries for Comer, Majka, and Stevens in the references at the end of this article.*

Note that **driverLoader** is run only on Driver Kit±based systems; NEXTSTEP on Motorola 680x0-based NeXT computers does not use Driver Kit.

First, **nmserver**, the Mach network server, is started. The **nmserver** encapsulates Mach IPC for transmission over the network within an IP packet, using either UDP or TCP. It also provides a simple service for registering names of available services; this registry can be queried by processes on the same computer and other computers on the network.

When invoked with **-nonet**, **nmserver** doesn't attempt to configure the network interface. It allows its services to be available for loading drivers, for example, but it does not assume there's a network interface. (The network portion will be initialized later in the startup sequence.)

Next, all device drivers specified in the configuration table are loaded and configured if the system uses Driver Kit.

Now, we're ready to turn on the network, using **/etc/rc.net.** (For the rest of this column, you might want a copy of **/etc/rc.net** handy.)

# rc.net

The **rc.net** shell script is invoked with **-h** to set the host name; without **-h**, the host name won't be set. When **rc.net** runs, it looks for **/etc/iftab**, the network interface table. If it's not there, **rc.net** uses the information in **/etc/hostconfig**. In addition, the default **iftab** just uses **hostconfig**. Let's assume for now that the default **iftab** is being used:

```
-1-    inet -HOSTCONFIG-
*      inet -AUTOMATIC-netmask -AUTOMATIC- -trailers up
```

*Incidentally, the* **-d** *flag can be used to debug* **rc.net***: It causes commands to be printed instead of executed.*

There are extensive comments in **/etc/iftab** describing the features available. For example, an arbitrary command can be run when an interface is to be configured, rather than just running **ifconfig**. (More on this in a future column perhaps.)

The **iftab** shown above specifies that **hostconfig** should be used to configure the primary network interface (the line beginning **-1-**) and to configure any other interfaces using automatic address and automatic network mask configuration (the line beginning *****).

Since only one network interface is supported under NEXTSTEP, we'll assume that there's only one network interface configured.

**rc.net** is written with a fair number of internal subroutine functions. The **hostconfig()** function, for example, parses **/etc/hostconfig**, setting various local variables for later use in actually configuring the network. There are three relevant parameters in **/etc/hostconfig** for configuring the interface: **INETADDR**, **IPNETMASK**, and **IPBROADCAST**. These correspond to the ªInternet Address,º ªNetmask,º and ªBroadcast Addressº fields in HostManager, respectively. If values other than **-AUTOMATIC-** are specified (ªAutoconfigure,º ªAutomatic,º and ªDefault,º respectively, in HostManager), then those specific values will be used for the Internet address, the network mask, and the local broadcast address. Otherwise, automatic determination of these will be done.

Following is a sample of **/etc/hostconfig**:

```
HOSTNAME=-AUTOMATIC-
INETADDR=-AUTOMATIC-
ROUTER=-ROUTED-
IPNETMASK=-AUTOMATIC-
IPBROADCAST=-AUTOMATIC-
NETMASTER=-NO-
YPDOMAIN=-NO-
TIME=-AUTOMATIC-
```

Normally, except for machines providing networkwide NetInfo services or configuration services, all three of these parameters should be set to **-AUTOMATIC-**; the rest of this discussion assumes this, since soldering in explicit values, while often appropriate, is pretty straightforwardÐand doesn't do anything interesting on the network! Similarly for automatic host name configuration: Let's assume that **HOSTNAME** is set to **-AUTOMATIC-** for this

discussion (in HostManager, this will be the ªHostnameº parameter's ªAutoconfigureº choice) and see what happens out on the network.

The commands that will be run for fully automatic configuration of an Ethernet interface are these:

```
ifconfig en0 inet -AUTOMATIC- netmask -AUTOMATIC- -trailers up
hostname -AUTOMATIC-
```

By default, the broadcast address is automatically determined, so no broadcast clause is needed.

Let's see what each portion of these commands does.

### Simple things first

Let's take care of the last two arguments on the **ifconfig** command line: **-trailers** and **up**. The **-trailers** argument disables *trailer link-level encapsulation*. (See the entries for Comer, Leffler,
and Stevens in the references section at the end of this article and Internet RFC 893 for information on trailer encapsulation.) Trailer encapsulation has been deprecated; NEXTSTEP does not use it.

The **up** argument simply configures the interface as up, which means ªactive,º as opposed to down.

### Automatic address configuration

The first thing to occur in the interface configuration is automatic determination of the

interface address, the processing of **inet -AUTOMATIC-**. This causes a BOOTP request to be sent to the network. The request takes the form of a UDP-based broadcast (from UDP port 68 to UDP port 67), specifying the interface's hardware address, requesting the interface's protocol address. In the case of our example, the hardware address is the Ethernet address; the protocol address is the Internet address.

This packet will make its way to all the **bootpd** server processes running on the machines on the local network (or subnet). Normally, NEXTSTEP will run a **bootpd** only when the **NETMASTER** parameter in **/etc/hostconfig** is **-YES-**; **NETMASTER** is set when a computer runs a networkwide NetInfo server, and thus is assumed also to be an appropriate machine to run the configuration servers **bootpd** and **rpc.bootparamd**.

The **bootpd** command is run about two-thirds of the way into **/etc/rc** during system startup. **bootpd**, which is listening on UDP port 67, will check its database when it receives the request;
if the requester's hardware address is found, **bootpd** will reply with the client's Internet address, along with other information, per the BOOTP spec. **bootpd** contacts **lookupd** to get the information in its database; **lookupd** obtains the information solely from NetInfo. Specifically, **bootpd** asks **lookupd** to search for an entry in **/machines** with an **en_address** property whose value is that specified in the request packet.

If the requester isn't found, *and* if automatic host addition is enabled, *and* if the **bootpd** is running on the same machine as the master server for the second-level NetInfo domain, then a dialog for automatic host addition begins. (This dialog will be described in a future column.) Note that an automatically added host will be added only to the appropriate second-level domain, not to any higher-level domains, if such exist.

Even though automatic host addition occurs only in the second-level domain, **bootpd**'s database is obtained from the entire (vertical view of the) NetInfo domain hierarchy.

What if multiple **bootpd**s are running on the local network and more than one respond to the request? The first ªpositiveº response to reach the requester is used. A positive response is one that specifies an explicit answer rather than inviting automatic host addition. To handle the possibility that one of these responders might search a domain that has the host and Ethernet address, and another might not find it because it's in a different domain hierarchy, any
ªnonpositiveº responses are ignored for about 10 seconds, after which time it's assumed that there really is no positive answer available.

### *THE BOOTP BOOTSTRAP PROBLEM*

*How can an IP packet be sent when the address of the sending interface isn't known? After all, one of the required fields in the IP header is the source address! Further, how does the client even receive the reply when it can't compare its IP address with the reply's destination IP address?*

*The BOOTP client code uses 0.0.0.0Ðthis hostÐas the source address for the BOOTP request and the
generic local broadcast addressÐ255.255.255.255Ðas the destination address. (Note that 0.0.0.0 is valid only as a source address and should be used only when the source's actual IP address is unknown.) The server replies to the client's actual IP address, encapsulating the IP packet in an appropriate DLC frame (Ethernet, token ring, and so on). The destination address in the DLC frame is the client's hardware address.*

*On receipt, the client's interface sees that the frame contains an IP packet and forwards the frame to the IP stack in the kernel. The protocol stack handles IP packets containing UDP-based BOOTP replies in a special manner when the interface's address is being configured. They're passed along up the protocol stack and the interface's IP address is set.*

*Once the system is done configuring its IP address, it broadcasts its address to other systems using ARP, so*

*those other systems can load their ARP tables, if desired.Ð*amm

## No response?

If no reply at all is received to the BOOTP request after about 20 seconds (four attempts, waiting 5 seconds after each attempt), the following message is sent to the console:

```
No response from network configuration server.
Type Control-C to start up computer without a network
connection.
```

In this case, the ªconfiguration serverº means the BOOTP server. If the machine in question shouldn't be using automatic address configuration, type Control-C, modify the local network configuration using HostManager (or by editing **/etc/hostconfig** carefully), and reboot. If the machine should be using automatic address configuration, verify that there's some computer on the same local network with a BOOTP server that has the correct information available.

## Automatic netmask configuration

The network mask is used to determine which of the bits in an Internet address are part of the network portion and which are part of the host portion. When **ifconfig** processes the **netmask -AUTOMATIC-** arguments, it sends an ICMP Netmask Request packet to the generic local network broadcast address (255.255.255.255). This packet is received by all the network interfaces on the local network. The IP stack itself in the kernel, rather than a separate user-level daemon, handles this request.

*To find out more about the significance of the difference between network and host portions, see the references on the IP protocols cited at the end of the column.*

If the network mask on the machine receiving the request was set explicitlyÐthat is, was not itself determined through automatic netmask configurationÐan ICMP Netmask Reply packet is sent to the requester. As with automatic address configuration, the first reply received is used.

***TIMEOUTS AND RETRIES***

*What happens when something requiring a response from ªthe networkºÐthat is, from some process running on some anonymous remote computerÐdoesn't get a response? What does it even mean to ªnot get a responseº?*

*Network communication in general uses timeouts to determine when ªno responseº has been received. What if a response arrives just after the timeout expires? The answer depends on the software: The late response might be ignored, or it might be used. If an identical retry is made, for example, the late response just might fulfill the outstanding request (now on its second iteration), in which case the response to the second request will be ignored.*

*What happens when a timeout occurs? Again, it depends on the software: The requester might request the information again, log a warning and keep trying, log an error and abort the attempt to acquire the information, or silently abort the attempt. Typically, when the requester tries again, it'll delay some amount of time before the retry; successive attempts will typically involve increasing delays. (The increasing delaysÐbackoffsÐare to minimize congestion. In an ideal environment, the timeout would be based on an expectation of the reasonable delays for the specific implementation, and the expiration of the timeout would indicate that the service is unavailable, not that it's busy.)*

*In the case of automatic network configuration, address and host name configuration will keep trying until receiving a response or being interrupted by user input. Netmask configuration will abort after the fifth retry, over a period of about a minute, with the error message* **ioctl (SIOCAUTONETMASK) timed out** *being sent to the console.*Ðamm

Normally, the network mask should be hardwired on machines providing networkwide NetInfo services or configuration services and on any appropriate networking equipment (such as routers). On all other NEXTSTEP computers and most other non-NEXTSTEP computers supporting this, it should be left to automatically configure the netmask. In NEXTSTEP Release 3.3, automatic netmask configuration is the default; prior to 3.3, the default was to use the netmask inherent in the address class of the interface. For example, an interface with a Class C address used a mask of 255.255.255.0Ðthe high-order 24 bits are in the network portion of the address, the low-order 8 bits are in the host portion.

### Automatic broadcast address calculation

Under almost all circumstances, the broadcast address should be automatically derivedÐ**IPBROADCAST** should be set to **-AUTOMATIC-** (or ªBroadcast Addressº set to ªDefaultº in HostManager). This will allow the system to determine the specific local broadcast address using the configured Internet address and network mask. The calculation turns on all the bits in the host portion of the broadcast address and leaves the network portion of the address set to that of the interface being configured.

Here's a table of Internet addresses, network masks, and default broadcast addresses, taken from the article, ªBefore NetInfo Starts,º in the Summer 1993 *NEXTSTEP in Focus.*

| Internet address | Netmask | Default broadcast address |
|---|---|---|
| 129.18.1.2 | 255.255.0.0 | 129.18.255.255 |
| 129.18.1.2 | 255.255.255.0 | 129.18.1.255 |
| 129.18.1.2 | 255.255.255.240 | 129.18.1.15 |
| 129.18.1.195 | 255.255.255.240 | 129.18.1.207 |

It's sometimes easier to understand unusual netmasks like the last two in the table if you see

them in hexadecimal format, so here are the same addresses in hexadecimal.

| Internet address | Netmask | Default broadcast address |
|---|---|---|
| 0x81120102 | 0xffff0000 | 0x8112ffff |
| 0x81120102 | 0xffffff00 | 0x811201ff |
| 0x81120102 | 0xfffffff0 | 0x8112010f |
| 0x811001c3 | 0xfffffff0 | 0x811201cf |

*For details about the port mapper, see the entries for the books by Bloomer, Comer, and Stevens and the second and third articles by Marcum in the references at the end of this article.*

Among the first three example addresses, the Internet address remains the same and the netmask changes, masking different bits. The default broadcast address changes accordingly. For the fourth example, the low-order byte (octet) in the Internet address changed from 02 to c3. The netmask indicates that the low-order nybble (4 bits) are in the host number portion. The default broadcast address is calculated appropriately.

The formula for computing a broadcast address is:

```
broadcast_address = Internet_address | (~netmask)
```

The broadcast address is the Internet address logically ORed with the logical NOT of the netmask.

### Automatic host name configuration

There's one last piece to the automatic network configuration puzzle: automatic hostname configuration. This service is provided by another process in the suite called the *configuration server*, **rpc.bootparamd**.

To enable automatic host name configuration, set the **HOSTNAME** parameter to **-AUTOMATIC-**, corresponding to the ªHostnameº parameter in HostManager being set to ªAutoconfigure.º This causes **rc.net** to execute the command **hostname -AUTOMATIC-** as noted above. The **hostname** command will send a packet to the network, requesting its host name. This request is made using the BootParams protocol, which is built atop SunRPC.

How are the **rpc.bootparamd**s contacted? A UDP broadcast packet is sent to UDP port 111, which is the port mapper's port number. The packet is a SunRPC Portmap request, PMAPPROC_CALLIT, an indirect invocation of some procedure in a SunRPC program. In this case, that SunRPC program is **bootparam**; the procedure is BOOTPARAMPROC_WHOAMI, requesting the host name for the given protocol (Internet) address.

When **rpc.bootparamd** receives a BOOTPARAMPROC_WHOAMI request, it looks for the requesting host in the normal network administration information sources using **gethostbyaddr()**. If the address is knownÐfor example, if there's a host name available for that addressÐ**rpc.bootparamd** looks for that host by name in NetInfo. (If NetInfo's not running, the **/etc/bootparams** file is consulted.) If the requesting host's name isn't found in NetInfo, NIS is checked; specifically, the **bootparams** map is searched, by requesting host's name. The DNS is not checked here, though the **gethostbyaddr()** call might contact the DNS.

***Questions?*** *Send your questions about system and network internals to* **smythy@NeXT.COM**.
*In future columns we'll address those that are most interesting.*

***WHY USE AUTOMATIC CONFIGURATION?***

*One might ask, ªWhy use automatic configuration? Why not just solder everything in, and use specific*

*values for all these parameters, like we're used to doing?º*

*We could use ªmanualº network configuration if we wanted to. But, we'd then lose a great deal of the flexibility and power available through our network administration and NetInfo, and a large network would require many more people to deploy and administer.*

*What are the specific benefits of automatic configuration? The keys are centralization, uniformity, and reliability. By centralizing all the necessary information on a very few computers, we can focus on ensuring that just those computers are always availableÐand making just a few computers highly available is pretty easy. Centralizing the data also makes it easier to manage: we know exactly where to go to change a given piece of information when necessary, and very few places require changing.*

*But uniformity is perhaps the most important benefit. Uniformity means that every ªclientº computer on the networkÐevery computer not providing networkwide services, and this is most of the computers on a large networkÐis the same as every other computer. Anything unique about that computer, its location, or its environmentÐthings like its address, its name, the names and locations of resources like file systems and nearby printersÐcan be obtained from the network. Because of centralization, modifying the information provided by ªthe networkº is fast and easy.*

*With each client computer identical to every other client computer, only one configuration is needed. Computers can be configured, their software loaded, from a fixed template, with no local configuration required. This allows for preconfiguring of spare computers, for example, or for replacing one person's failed computer with that of someone on vacation and not worrying even about moving the ªcorrectº computer back to the person on vacation.*

*Operationally, all that's required to replace a broken computer is to modify the* **en_address** *property of the appropriate subdirectory of /***machines** *in the correct NetInfo domain and boot the machine. Absolutely no specific reconfiguration of the new computer's software is necessary. Besides decreasing the number of people required to administer and maintain a large network, it also means that people who haven't gone through the extensive training required to administer a very large network can take care of things like hardware replacement essentially with no training.*

*Complete uniformity even allows people to use a computer in a different office, anywhere on a potentially worldwide network, and see the same view as from his or her ªownº computer.*

*There is a detrimental effect to using automatic configuration: Network traffic is increased during startup. This can be a particular problem if many computers are starting or restarting at the same time, such as might happen when the power is restored following a power failure. There are also potential security issues.*Ðamm

Now, what does it mean for the host's name to be found in NetInfo? In fact, two things are required. First, somewhere in the NetInfo domain hierarchy, there must be a **/machines** subdirectory whose **name** property has, as one of its values, the host's name. Second, in the case of the **bootparams** search, there must be a **bootparams** property in that directory (in the same domain). The **bootparams** property need not necessarily have a value, but it must be present; else, the host's name ªwon't be found.º

It's very important to remember that automatic host name configuration requires all three of the following things:

**1** The host must be known by addressÐit must be possible to translate the host's (interface's) Internet address into a name using **gethostbyaddr()**.

**2** There must be a **/machines** subdirectory for the host name found in Step 1 somewhere in the NetInfo domain hierarchy.

**3** In the directory and domain found in Step 2, there must be a **bootparams** property. Note that for automatic host name configuration, the **en_address** property is unused, just as the **bootparams** property isn't used for automatic address configuration.

If any of the above conditions is not met, no response will be provided by **rpc.bootparamd**. In this case, after about 45 seconds the **hostname** command will print the following message on the console:

```
Configuration server not responding to request for hostname
Do nothing to keep waiting or press 'c' to continue.
```

On the other hand, if the requesting host's name is found, a reply containing the host name, along with other information that is ignored, is sent to the requester.

Finally, back on the requesting computer, the **hostname** command receives the reply, sets its host's name, prints the information about its having received a name from a server on a particular computer, and returns.

# Last Bits of rc.net

After setting the host name, **rc.net** sends a **SIGUSR2** to **nmserver**. This tells **nmserver** to initialize its network portion, because a network interface is now configured and available. **rc.net** then returns to **/etc/rc**.

# MORE TO COME¼

Believe it or not, we've not gone terribly far in **/etc/rc**. We've only covered about 10 to 15 percent of the code in **/etc/rc**, in fact. But, we have a network driver loaded, an interface configured, and a host name set. In future editions of this column, we'll cover the rest of **/etc/rc**, with continuing focus on the network-related aspects of startup. n

*Alan M. Marcum is a member of NeXT's Premium Support Team and specializes in the management of large networks.*

**References**


Bloomer, John. *Power Programming with RPC*. Palo Alto, CA: O'Reilly & Associates, 1991.

Comer, Douglas. *Internetworking with TCP/IP, Volume I: Principles, Protocols, and Architecture*. Englewood Cliffs, NJ: Prentice Hall, 1991.

Leffler, Samuel J., Marshall Kirk McKusick, Michael J. Karels, and John S. Quarterman. *The Design and Implementation of the 4.3BSD UNIX Operating System*. Palo Alto, CA: Addison-Wesley, 1989.

Majka, Marc. "Behind the Scenes of NeXT Networking," *Support Bulletin*, Summer 1992. Redwood City, CA: NeXT Computer, 1992.

Marcum, Alan M. "Before NetInfo Starts," *NEXTSTEP in Focus*, Summer 1993. Redwood City, CA: NeXT Computer, 1993.

Marcum, Alan M. "NetInfo Binding and Connecting," *NEXTSTEP in Focus*, Summer 1993. Redwood City, CA: NeXT Computer, 1993.

Marcum, Alan M. "The Tough Stuff," *NEXTSTEP in Focus*, Summer 1993. Redwood City, CA: NeXT Computer, 1993.

Stevens, W. Richard. *TCP/IP Illustrated, Volume 1: The Protocols*. Palo Alto, CA: Addison-Wesley, 1994.

**Next Article** NeXTanswer #1988 **appDidInit:**
**Previous article** NeXTanswer #1992 **Realities of Distributed Objects**
**Table of contents** http://www.next.com/HotNews/Journal/OSJ/SpringContents95.html