IP Address Management: Past, Present and Future

By Cricket Liu and Matt Larson
Acme Byte & Wire

Introduction

IP address management is critical to reliable TCP/IP network operation. It's a task that's especially challenging in today's corporate networking environments, with the increasing use of Internet-based technologies. Modern networks have grown both in scope and in services, and older methods and tools for managing IP addresses aren't always up to the demands. The challenges include multiple DNS zones, hundreds—even thousands—of nodes, roaming and remote-access users, and the need to preferentially allocate bandwidth to the most critical traffic.

This paper traces the development of IP address management methods. It provides a historical background so the reader can better appreciate how the earliest schemes have evolved into today's tools. We'll review the features of currently available commercial solutions, and discuss possible trends in the future of IP address management.

What is IP Address Management?

In its most basic sense, IP address management refers to the maintenance of all the information associated with a network's IP address space. At a high level, the network administrator needs to know:

- How much free address space to I have? (In the world before variable-length subnet masks this question was, What free subnets do I have?)
- What subnets are in use?
  - How large are they?
  - Who is using them?
  - Where are they?
  - How many addresses are free in each of them?

At a lower level, the administrator needs to know information about each allocated IP address:

- Is the IP address permanent or temporary?
- What IP-specific configuration parameters does this device have? (What is its default router? What name servers does it use? Etc.)
- What host name is associated with this IP address?
- What hardware address is associated with this IP address?
- Where is the device physically located?
- What is its serial number, company asset number or other similar information?

Whatever the size of the network, these fundamental questions have remained unchanged from the earliest days of TCP/IP. What has changed is the scope and sophistication of the tools used to answer these questions, keep track of the answers and automate the administration process.

The Early Days

The ARPANET—the Department of Defense network that is the ancestor of today's Internet—was built in the late 1960's and used a proprietary protocol suite. This first protocol proved to have shortcomings in linking with other networks, which led to the development of TCP/IP.

TCP/IP uses 32-bit address numbers. These soon proved to be unwieldy for most users, even when expressed in the less daunting format of four eight-bit decimal numbers, delimited by periods. The obvious solution was a scheme for addressing computers by name: people relate much better to names and find it much easier to remember the computer in the corner as *frodo* rather than *192.168.1.2*.

The First Generation: Host Tables

The desire to refer to machines by name instead of number led to the first IP address management scheme: the host table. The host table is a file that contains all the IP addresses in use on a network, along with their names. The host table provides a mapping from a host's name to its IP address, as well as the reverse mapping: given a host's IP address, one can look up its name.

For a host table to be most useful, it must contain the names of all the hosts that a given host might want to communicate with. For the ARPANET, that meant that the host table had to contain the names and IP addresses of every host on the network. Such a file was maintained by the Network Information Center ("the NIC"), the central organization responsible for managing the ARPANET. The file was called *HOSTS.TXT* and was similar in format to the UNIX */etc/hosts* file. Network administrators all over the network emailed host table changes to the NIC every time they added or deleted a host or changed a host's IP address. The NIC made the changes to its master host table, which it made available via FTP. Administrators periodically downloaded the latest version of the host table to stay current.

As the ARPANET exploded in size to eventually become the Internet, the rapid growth stressed the centralized host table scheme. The first problem was consistency: the file changed too quickly for everyone to have a current copy. Just as printed phone books are out of date as soon as they arrive, so too was the host table's information outdated after it was downloaded. A second problem was the lack of a hierarchical naming space for host names. The host table required short, single-label names for hosts. As the population of hosts grew, so did contention for popular names and it became harder and harder to avoid duplicates. Worse, operator error at the NIC sometimes led to duplicates slipping into the file and wreaking havoc. Finally, the traffic and load just from downloading the file began to represent a significant portion of the ARPANET's total traffic. This model just didn't scale well.

From an IP address management perspective, the host table has both advantages and disadvantages:

- *Centralization.* The centralized aspect of host tables is appealing from an IP address management standpoint: A host table provides a central location for an entire network's IP address information.
- *Simple format.* The host table format (both the old *HOSTS.TXT* and the modern UNIX */etc/hosts*) is simple and easy to edit. After all, the file contains only host names and IP addresses. It is easy for an administrator to make changes and make them correctly.
- *Limited contents.* The simple format is also a drawback, since it limits the amount of additional information that can be stored easily, such as host location information, serial numbers, etc.

The Second Generation: DNS "by Hand"

The problems with the centralized host table scheme led directly to the development of the Domain Name System (DNS) in the early 1980s. The two main design goals of DNS grew out of the biggest problems with the host table: distributed administration and a hierarchical naming scheme.

DNS defines a name space composed of hierarchical domains. Each organization is assigned a domain, such as *metainfo.com*, under which it can place hosts and create still more hierarchy by creating subdomains, such as *sales.metainfo.com*. Domains are broken into units called *zones* for administrative purposes. Thus administration is no longer centralized but divided along zone lines: Individual organizations administer the information corresponding to their zones. DNS is really just a distributed database with data maintained locally but available globally.

Programs called *name servers* are the workhorses of DNS. They store information about particular zones, loading this information from *zone database files* on disk. Name servers then answer questions from other name servers about these zones. Name servers also know how to contact other name servers to find information about other zones.

DNS was good for the quickly growing Internet. It removed the central bottleneck at the NIC and the now-familiar domain-based names paved the way for a truly vast number of hosts with unique names. But from a network administrator's IP address management standpoint, it made some things more difficult.

- *Zone database file complexity.* The format of DNS zone database files is more complicated than the simple host table format, since DNS was designed to store more than just names and IP addresses. (For example, SMTP mail routing information is also stored in DNS, something the host table is not capable of handling.) The more complicated format lends itself to making mistakes when editing the zone database files by hand.
- *Decentralization.* With DNS, the information that was formerly centralized in the host table may now be spread among multiple name servers. Depending on how an organization creates zones, a network administrator looking for IP address information might have to consult multiple files on multiple name servers.

- *Limited contents*. Like the host table, DNS lacks the ability to easily store additional host information, such as host location information, serial numbers, etc. While this information can be stored as simple comments, it is difficult to search and any reporting would require add-on software.

The Third Generation: DNS With Homemade Tools

Editing DNS zone database files by hand tends to be a workable solution only for smaller organizations:

- Smaller organizations have fewer hosts, and therefore a smaller and more comprehensible amount of DNS information.
- They often have fewer host moves, adds and changes, resulting in less frequent changes to DNS information.
- They also usually have only a few people making these changes, and therefore fewer people who need to understand the somewhat complicated zone database file syntax.
- Smaller organizations also tend to be more tolerant of the inevitable problems that arise from editing DNS information by hand.

Larger organizations usually decide that DNS by itself is not enough. Before the advent of commercially available IP address management software (discussed in the next section), organizations seeking to avoid DNS hand editing were forced to use freely available software or write their own tools.

Freely available software to simplify IP address management tasks had several disadvantages:

- *Functionality*. Software tended to focus on the automation of one particular function, rather than the larger feature set supported by later commercial IP address management software.
- *Support*. No support is usually available for freely available software, which is unacceptable for most companies.
- *Quality*. The quality of freely available software can be an issue, since anyone can write and distribute it. Commercial software is generally of higher quality.

An example of software in this category is *h2n*, a small Perl program distributed with the book *DNS and BIND*. *h2n* converts a host table into DNS zone database files. While designed to be used once during an organization's conversion from the use of host tables to DNS, it can also be used on a day-to-day basis: The host table becomes the canonical listing of IP addresses and host names and *h2n* converts the information to DNS format. This offers the administrator the advantages of the host table—editing one file in a simple format—while using DNS, a necessity in today's Internet. But *h2n* is a simple tool focused on one task. It merely eliminates the need to hand-edit DNS zone database files.

Most large organizations made the decision to write some kind of customized application to avoid editing DNS files by hand and implement some rudimentary IP address management features. This course of action also had disadvantages:

- *Development cost*. Software development is almost always expensive, and many companies ambitiously began developing packages with a large feature set. Such development ended up taking a significant amount of time and money.
- *Support cost*. Once developed, the software must be supported and maintained in-house, an ongoing burden.
- *Technical knowledge*. This is difficult software to develop, potentially exceeding the capability of a company's internal development staff.

Of course, the tremendous advantage to developing one's own software is that it has exactly the features required.

We are aware of a Fortune 25 company that had no less than a dozen custom IP address management packages. Some were very simple and not much more complicated than *h2n*, while others performed some of the tasks later seen in commercial packages, such as tracking additional host information like serial and asset numbers. The cost to the company of maintaining all of these packages, however, was staggering.

The Advent of DHCP

The early 1990s saw the development and widespread implementation of the Dynamic Host Control Protocol (DHCP), which somewhat paradoxically simplified the lives of network administrators but increased the complexity of IP address management.

DHCP evolved from the Bootstrap Protocol (BOOTP), which allowed devices without permanent storage, like printers, X terminals and routers, to obtain an IP address and other configuration parameters at startup. The network administrator maintained a table of device hardware addresses and corresponding IP addresses. A device's hardware address had to be known and entered on the BOOTP server, along with the IP address that would always be assigned to that device.

A DHCP server also assigns IP addresses and configuration parameters to devices, but with a significant difference: a DHCP server does not need to know a device's hardware address ahead of time. DHCP servers own *lease pools*, groups of IP addresses that may be dynamically assigned to DHCP clients on a first-come, first-served basis. The IP address assignment is called a *lease*. It may be permanent, but as the name suggests, it is more often valid for an interval called the *lease time*. Devices with non-volatile storage—the norm today—attempt to *renew* their leases with the DHCP server upon startup.

DHCP represented a tremendous time savings for network administrators, who had previously been required to visit each device to configure its IP address and other parameters, such as default router, name servers, etc. DHCP allows these devices to obtain this information out of the box, without any prior configuration—DHCP need only be enabled.

From an IP address management perspective, however, DHCP servers represented a whole new class of servers to configure and manage, along with name servers. Their configuration is similar. For example, DHCP lease pools are often entire subnets, which

must also be configured on name servers as *in-addr.arpa* subdomains (the DNS construct that allows reverse mapping of IP address to host name).

Unfortunately, early DHCP servers didn't communicate with name servers, a situation still true today for almost all DHCP servers. Thus the DHCP server generates new name-to-IP address mappings as it leases IP addresses, but this information is not reflected in DNS. Admittedly, the traditional DNS update mechanism is not conducive to this one-at-a-time-style update. The name server is designed for bulk updates and had to transfer an entire zone, even if only one piece of information had changed.

The DNS community recognized this deficiency, particularly in light of the growing popularity of DHCP and the desire for better integration between the two protocols. The result was a set of DNS protocol extensions developed in the mid-1990s and commonly referred to as *dynamic DNS*. These extensions allow a DHCP server (or any authorized updater) to make changes to a running name server, which in turns notifies its peers immediately and propagates only the changed information.

By the mid-1990s, DHCP was in common use, which both increased and decreased the IP address management workload of network administrators. On the one hand, they no longer had to visit each host to configure its IP address and other parameters. On the other hand, they now had DCHP servers to maintain along with DNS servers. And while the protocols had been developed to allow DNS synchronization with DHCP lease activity, DHCP vendors did not implement them immediately. Finally, DNS and DHCP alone cannot track the additional physical information about a device associated with its IP address. The stage was set for software products to integrate these protocols and fill in the gaps.

The Fourth Generation: Commercial IP Address Management Software

By the mid-1990s, commercial IP address management software appeared from several vendors. It offered the following advantages and features:

- *Centralized management*. The ability to centrally manage an entire organization's DNS and DHCP servers is probably the single largest advantage of commercial IP address management software. From one location, the software generates configuration and data files for all DNS and DHCP servers, and transfers the files to the appropriate servers and even restarts the server processes so the changes take effect. A relatively recent feature is cross-platform support: Early packages focused on UNIX, but the growing popularity of Windows NT has generated support for it as well.
- *Graphical user interface*. Many commercial packages support a graphical user interface (GUI), which is a great improvement over editing configuration files by hand on a terminal. A GUI simplifies the presentation of information, making it easier to interpret more information at once. It also simplifies data input and hides the underlying technical configuration, allowing more users than just the network administrator to make changes. Several commercial packages employ Java-based

interfaces that can be accessed through a web browser. These allow authorized users to perform management functions from anywhere on the network, without having to install separate software.

- *Consistency*. All packages have a central data store from which configuration and data files for DNS and DHCP servers are created. In early packages, the data store is either a full-blown relational database from a third party or another less complicated commercial database. This central data store, combined with the ability to do error and syntax checking in the GUI, increases the consistency and correctness of the data. For example, editing DNS files by hand requires two entries, one for the name-to-IP address mapping and another for the reverse. A common error is to forget one or the other, but this is easily avoided with IP address management software, along with all other syntax errors that stem from hand editing.
- *Security*. Most packages also implement access control lists, allowing the creation of different classes of users, from administrators with wide-ranging powers to operators who can only add and change host information.
- *Dynamic DNS*. The commercial IP address management packages were the first to integrate DHCP and DNS information on a real-time basis. Early packages used proprietary protocols rather than the standards-based dynamic DNS, but packages available today offer DHCP servers modified to provide true dynamic DNS and the ability to update DNS servers in real time. This synchronization allows the DNS to provide an up-to-date view of the network at all times.
- *Extensible and customizable*. Most packages allow the administrator to store considerable additional information along with each IP address, such as serial number, asset number, physical location, etc. Often these fields are customizable and additional fields may be added, allowing the administrator to track the desired information.
- *Reporting*. Powerful reporting capabilities are a feature of most packages, allowing administrators to produce a report on just about any view of the IP address data: free subnets, allocated subnets, including percent used—the list goes on.

The Future

The future of IP address management products is, of course, still being written. Based on our experience in this field, we foresee the following features and trends:

- *Ubiquity*. Eventually IP address management products will be as common as the DNS and DHCP servers they currently manage. Software to manage these services effectively will be as necessary as the services themselves.
- *Directory-enabled applications*. The entire industry is in the middle of a push to directory-enabled applications, and IP address management software is no exception. There are currently several packages available that use an LDAP-based directory server for their central data store, rather than a relational database. We expect this trend to continue. The directory-enabled model offers several advantages:
  - *Simplicity*. A directory server is considerably less complicated than a relational database.
  - *Content suitability*. Directory servers handle information in the format of attribute-value pairs. IP address management information is easily represented in this format and is thus an ideal choice for storage in a directory server.

- *Quantity.* The total amount of IP address information of most organizations does not require the nearly infinite scalability of relational databases, but is well suited to the capacity of currently available directory servers.
- *Shared data store.* Multiple applications, including IP address management software, can share an organization's single directory server. In addition, the IP address management data in the directory server is easily accessible to all directory enabled applications.

- *Integration with other protocols*. It is inevitable that IP address management products will grow in scope to encompass other protocols. A good candidate is RADIUS, the authorization protocol used by terminal servers and other devices. Some packages integrate with and manage RADIUS servers today.
- *User information.* User-address mapping, which relates user information in the central data store to the currently assigned IP address, is a logical next step. This user information could be the basis for:
  - *Authentication.* RADIUS servers could obtain users' information from IP address management software. This capability could also be used by any other device making authentication decisions, such as a firewall.
  - *Authorization.* Software and devices making authorization decisions—whether or not a user is allowed access to a particular resource—could also use information in the central data store.
  - *Usage tracking.* User activity could be correlated to other network information, such as DHCP lease activity, to obtain a picture of usage traits for different users.
- *Policy-based management.* The ability to relate IP addresses to user information enables policy-based management, for controlling access to network resources on a per-user basis. While password schemes can allow or deny access to individual servers, this type of policy-based management would be built directly into the network's infrastructure including bridges, routers and firewalls. Uses include:
  - *Remote access.* Remote users could be authenticated by RADIUS, then granted or denied access to all or part of the network according to their user profile.
  - *Virtual private networking.* Firewalls could be configured to permit or deny access on a per-user basis, even though their IP addresses might be dynamically assigned.
  - *Bandwidth provisioning.* In an actively managed network, packets destined for the CEO or for servers performing key business functions could be given priority over streaming video.
- *Management platform.* Because network routers, bridges, firewalls and other devices depend largely on IP addressing, address management software can provide a platform for actively managing the network. Just as current packages store configuration information centrally and download files to DNS and DHCP servers, future packages could do the same with network hardware. A dominant software developer could create a *de facto* standard mechanism for this integration.

IP address management is key to maintaining a secure, reliable, robust network that works for its users. The new generations of management software will allow IT professionals to meet the demand for more TCP/IP services, extend services to remote sites, and efficiently manage the traffic on their growing networks.

Cricket Liu is the co-author of the O'Reilly & Associates Nutshell Handbook *DNS and BIND*, the only book devoted to the Domain Name System, and is the principal author of O'Reilly's *Managing Internet Information Services*, which was one of the first books to describe web server configuration and management. He is also the co-founder of HP's Internet consulting business and author of the program's business plan. As a Senior Technical Consultant in HP's Professional Services Organization, Cricket designed DNS architectures and built Internet firewalls for some of HP's largest customers, including Caterpillar, Ford, GTE, Imation, Lockheed Martin, the Securities Industry Automation Corporation, Sprint, and U S WEST.

Before joining the PSO, Cricket worked for five and a half years at HP's Corporate Network Services, where he ran hp.com, one of the largest corporate domains in the world, and helped design the HP Internet's security architecture.

Matt Larson started working with Internet technology at Northwestern University. From 1990 to 1992, he managed the TCP/IP backbone network of Northwestern's engineering school, administered its DNS environment, and provided UNIX and network support to various departments.

Matt went to work for Hewlett-Packard's Corporate Network Services group in 1992, which is responsible for designing and running the HP Internet, one of the world's largest private TCP/IP networks. During his three years supporting and engineering HP's network, Matt specialized in DNS and Internet security issues. He succeeded Cricket and spent two years as hp.com hostmaster, the person ultimately responsible for DNS throughout HP. He also designed solutions for HP business divisions that allowed them to communicate with their customers and suppliers without compromising HP's network security.

In 1995, Matt became a consultant in the financial services practice of HP's Professional Services Organization and was one of the practice's Internet specialists. He applied Internet technology to financial applications, such as Internet banking and electronic commerce, and focused on Internet security and DNS. His accounts included the Securities Industry Automation Corporation and KeyBank.