2

INTRODUCTION TO THE DOMAIN NAME SYSTEM (DNS)

DNS OVERVIEW – DOMAINS AND RESOLUTION

As we introduced in Chapter 1, DNS is a foundational element of IP communications. DNS provides the means for improved usability of IP applications, such as insulating end users from typing IP addresses directly into applications like web browsers and enabling web servers to serve web pages compromised of diverse linked content. To communicate over an IP network, an IP device needs to send IP packets to the intended destination; and each IP packet header requires both source and destination IP addresses. DNS provides the translation from a user-entered named destination, for example, web site www address, to its IP address such that the sending device may populate the destination IP address with the address corresponding to the entered domain name.

DNS is not only useful for Internet users but also for network administrators. By publishing name-to-IP address mappings in DNS, the administrator gains the freedom to change IP addresses as needed for network maintenance, service provider changes, or general renumbering without affecting how end users connect. I can map my www address to 192.0.2.55 today and change it to 198.51.100.23 tomorrow without affecting how users reach my website. Of course, I'd need to keep both addresses active for some time given the caching of the former mapping in various recursive servers that had queried for my address before I made the change.

DNS Security Management, First Edition. Michael Dooley and Timothy Rooney.

This level of indirection is also useful when configuring Internet "things" such as home appliances, smart cars, surveillance cameras, etc. that can connect to the Internet independently of user-initiated commands. Such connections may be initiated first by a DNS lookup of its configured web address, to which it connects to communicate a status update, alert, or other information. The thing's home website IP address may be changed as needed by changing its DNS entry in a manner similar to that just described.

As a network service, DNS has evolved from this simple domain name-to-IP address lookup utility to enabling very sophisticated "lookup" applications supporting voice, data, multimedia, and even security applications as we shall discuss in Chapter 13. DNS has proven extremely scalable and reliable for such lookup functions. We'll discuss how this lookup process works after first introducing how this information is organized.

Domain Hierarchy

The global domain name system is effectively a distributed hierarchical database. Each "dot" in a domain name indicates a boundary between tiers in the hierarchy, with each name in between dots denoted as a *label*. The top of the hierarchy, the "." or *root* domain provides references to top-level domains, such as .com, .net, .us, .uk, which in turn reference respective subdomains. Each of these top-level domains or TLDs is a child domain of the root domain. Each TLD has several children domains as well, such as ipamworldwide.com with the ipamworldwide domain beneath the com domain. And these children may have child domains and so on.

As we read between the dots from right to left, we can identify a unique path to the entity we are seeking. The text left of the leftmost dot is generally* the host name, which is located within the domain indicated by the rest of the domain name. A *fully qualified domain name* (FQDN) refers to this unique full [absolute] path name to the node or host within the global DNS data hierarchy. Figure 2.1 illustrates a FQDN mapping to the tree-like structure of the DNS database. Note that the trailing dot after .com. explicitly denotes the root domain within the domain name, rendering it fully qualified. Keep in mind that without this explicit FQDN trailing dot notation, a given domain name may be ambiguously interpreted as either fully qualified or relative to the "current" domain. This is certainly a legal and easier shorthand notation, but just be aware of the potential ambiguity.

NAME RESOLUTION

To illustrate how domain information is organized and how a DNS server leverages this hierarchical data structure, let's take a look at an example name resolution. Let's

* Some environments allow dots within hostnames which is relatively uncommon though permissible.

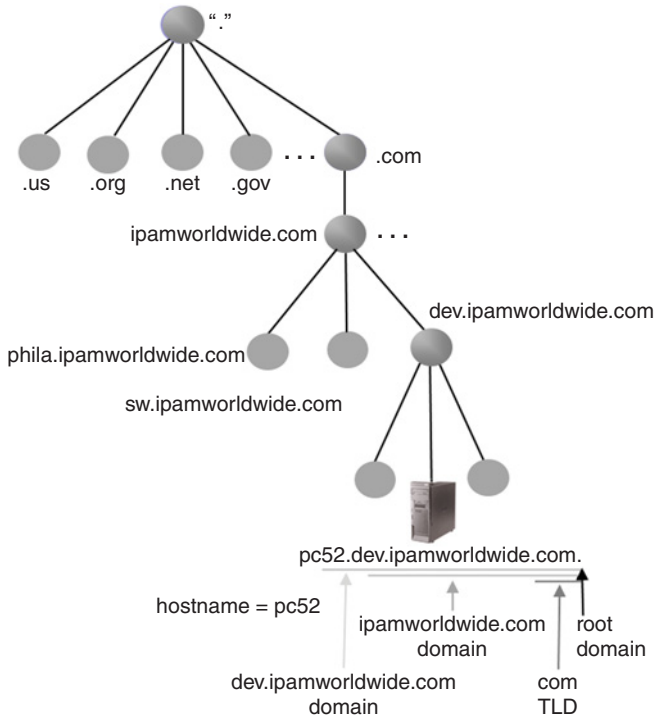

Figure 2.1. Domain Tree Mapping to a Fully Qualified Domain Name

assume you'd like to connect to a website, www.example.com. You'd enter this as your intended destination in my browser. The application (browser in this case) utilizes the sockets* application programming interface (API) to communicate with a portion of code within the TCP/IP stack called a *resolver*. The resolver's job in this instance is to translate the destination domain name you entered into an IP address that may be used to initiate IP communications. If the resolver is equipped with a cache and the answer resides in cache, the resolver responds to the application with the cached data and the process ends.

If not cached, the resolver issues a query for this domain name to the local recursive DNS server, requesting the server provide an answer. The IP address of this local DNS server is configured either manually or via DHCP using the domain server's option (option 6 in DHCP and option 23 in DHCPv6). This DNS server will then attempt to answer the query by looking in the following areas in the specified order and as illustrated in Figure 2.2. We recommend you specify at least two local DNS

* This API call is from the application to the TCP/IP layer of the protocol stack. The gethostbyname sockets/Winsock call initiates this particular process.

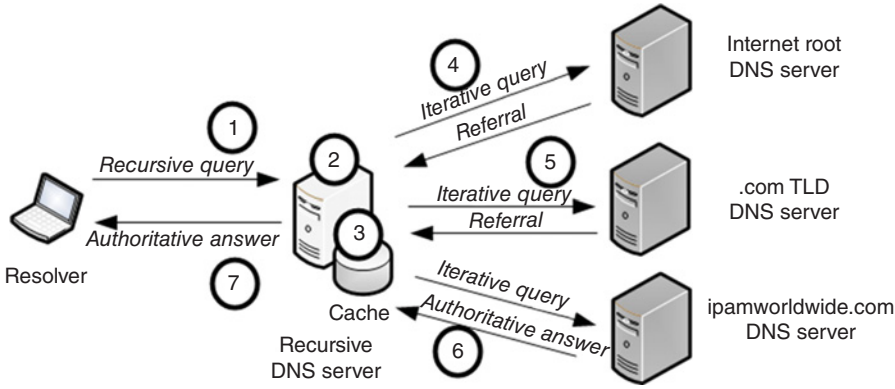

Figure 2.2. Recursive and Iterative Queries in Name Resolution

server addresses to provide resilient resolution services in the event of a single DNS server failure.

We refer to this DNS server to which the resolver issues its query as a *recursive server*. "Recursive" means that the resolver would like the DNS server to try to find the answer to its query even if it does not know itself. From the resolver's viewpoint, it issues one query and expects an answer. From the recursive DNS server's perspective, it attempts to locate the answer for the resolver. The recursive server is the resolver's "portal" into the global domain name system. The recursive server accepts recursive queries directly from client resolvers and performs the following steps to obtain the answer to the query on behalf of the resolver.

1. The resolver initiates a query to resolve `www.example.com` to the recursive DNS server. As mentioned, the resolver knows which DNS server to query based on configuration via manual entry or via DHCP.
2. The queried server will first search its configured data files. That is, the DNS server is typically configured with resource record information for which it is *authoritative*. This information is typically configured using text files, a Microsoft Windows interface, or an IP address management (IPAM) system. For example, your company's DNS servers are likely configured with resolution information for your company's IP devices. As such, this is authoritative information. If the answer is found, it is returned to the resolver and the process stops.
3. If the queried server is not authoritative for the queried domain and is configured to enable recursion, it will access its cache to determine if it recently received a response for the same or similar query from another DNS server during a prior resolution task. If the answer for `www.example.com` resides in

cache*, the DNS server will respond to the resolver with this non-authoritative information and the process stops. The fact that this is not an authoritative answer is generally of little consequence, but the server alerts the resolver to this fact in its response.

4. If the queried DNS server cannot locate the queried information in cache, it will then attempt to locate the information via another DNS server that has the information. There are three methods used to perform this "escalation."
   a. If the queried recursive server is configured to forward queries to another server, the server will forward the query as configured in its configuration or zone file and will await a response.
   b. If forwarding is not configured and the cache information referenced in step 3 indicates a partial answer to the query, it will attempt to contact the source of that information to locate the ultimate source and answer. For example, a prior query to another DNS server, server X, may have indicated that DNS server X is authoritative for the `example.com` domain. The recursive DNS server may then query DNS server X for information leading to resolution of `www.example.com`.
   c. If no information is found in cache, the server cannot identify a referral server, or forwarding did not provide a response† or is not configured, the DNS server will access its *hints* file. The hints file provides a list of *root name servers* to query in order to begin traversing down the domain hierarchy to a DNS server that can provide an answer to the query.

   Upon querying either a root server or a server further down the tree based on cached information, the queried server will either resolve the query by providing the requested IP address(es) for `www.example.com`. or will provide a referral to another DNS server further down the hierarchy "closer" to the sought-after FQDN. For example, upon querying a root server, you are guaranteed that you will not obtain a direct resolution answer for `www.example.com`. However, the root name server will *refer* the querying DNS server to the name servers that are authoritative for `com` as illustrated in Figure 2.2. The root servers are "delegation-only" servers and do not directly resolve queries, only answering with delegated name server information for the queried TLD.
5. The recursive server *iterates*‡ additional queries based on responses down the domain tree until the query can be answered. Continuing with our example,

* Cache entries are temporary and are removed by DNS servers based on user configuration settings as well as advertised lifetime ("time to live," TTL) of a resource record.

† If the `forward only` option is configured, the resolution attempt will cease if the forwarded query returns no results; if the `forward first` option is configured, the process outlined in this paragraph ensues, with escalation to a root server.

‡ These "point-to-point" queries are referred to as iterative queries.

the recursive server queries one of the `.com` name servers referred to by the root server. The answer received from that name server will also be a referral to the name server that is authoritative for `example.com.`, and so on down the tree.

Note that by issuing queries to other DNS servers to locate resolution information, the recursive server itself performs a resolver function to execute this lookup. The term *stub resolver* is commonly used to identify resolvers which do not iterate down the domain tree. Stub resolvers such as those within typical end user clients are configured only with which recursive name servers to query.

6. Upon receipt of the referral from the TLD server indicating two or more name servers and corresponding IP addresses that are delegated to example.com, the recursive server prepares its next iterative query to issue to one of these servers. The response includes the answer to the query in the form of one or more resource records matching the queried name, class, and resource record type.

   The recursive server generally updates its cache not only with the ultimate answer for the specific query, but with any additional information provided with the answer and referral messages received in the process. In this way, the recursive server caches the name server domain names and IP addresses for the .com and example.com domains. When the same or another stub resolver queries for another domain with the .com domain subtree, the recursive server can utilize its cache to query one of the .com name servers directly without needing to query the root server as mentioned in step 4 previously. If an answer cannot be found, the recursive server will also cache this "negative" information as well for use in responding to similar queries.

7. When the answer is received, the recursive DNS server will provide the answer to the stub resolver as per Figure 2.2 which can update its cache and the process ends. It might seem that this process is laborious and time consuming, but as you've probably experienced, it all happens very quickly. Root and TLD servers are "delegation-only" servers in that they only support referrals. This helps performance, as does caching by the recursive DNS server, which can leverage cached DNS information to streamline the resolution process.

   As mentioned earlier, the stub resolver may be configured to cache this information as well. In this case, only the answer is cached, not other domain tree nodes, because the stub resolver only queries its configured name servers. But for users frequently visiting common sites, the stub resolver cache can improve application response time.

We mentioned the use of DNS forwarders in step 4, where the server queried by stub resolvers forward all or some queries to other DNS servers for recursion. You might deploy this configuration if you have a consolidated set of internal caching servers

through which you funnel external-bound DNS queries or if you're using a third-party DNS resolver service. In both cases, you can configure your internal forwarding DNS servers to resolve internal name space and forward other queries to your caching servers or to the service providers' DNS servers, respectively. The forwarding DNS server generally caches resolution answers like recursive servers to minimize DNS message flow and improve resolution performance.

In summary, the resolution process entails (a) finding a set of name servers with authoritative information to resolve the query in question, and (b) querying an authoritative server for the desired information. In our example, the desired information is the IP address corresponding to the host domain name `www.example.com`. This "translation" information, mapping the queried domain name to an IP address, is stored in the DNS server in the form of a resource record. Different types of resource records are defined for different types of lookups. Each resource record contains a "key" or lookup value and a corresponding resolution or answer value. In some cases, a given lookup value for a given type may have multiple entries in the DNS server configuration. In this case, the authoritative DNS server will respond with the entire set of resource records, or *resource record set* (*RRSet)*, matching the queried value (name), class, and type.

The organization of DNS data is such that DNS servers are configured at all levels of the domain tree as authoritative for their respective domain information, including where to direct queriers further down the domain tree. In most cases, these servers at different levels are administered by different organizations. Not every level or node in the domain tree requires a different set of DNS servers as an organization may serve multiple domain levels within a common set of DNS servers.

While the top three layers of the domain tree typically utilize three sets of DNS servers under differing administrative authority, the support of multiple levels or domains on a single set of DNS servers is a deployment decision. This decision hinges primarily on whether administrative delegation is required or desired. For example, the DNS administrators for the ipamworldwide.com domain may desire to retain administrative control of the sw.ipamworldwide.com. domain, but to delegate dev.ipamworldwide.com to a different set of administrators and name servers. This leads us to a discussion regarding the distinction between zones and domains.

ZONES AND DOMAINS

The term *zone* is used to differentiate the level of administrative control with respect to the domain hierarchy. In our example, the ipamworldwide.com zone contains authority for the ipamworldwide.com, sw.ipamworldwide.com, and phila.ipamworldwide.com domains, while the dev.ipamworldwide.com zone retains authority for the dev.ipamworldwide.com domain and its subdomains as illustrated in Figure 2.3.

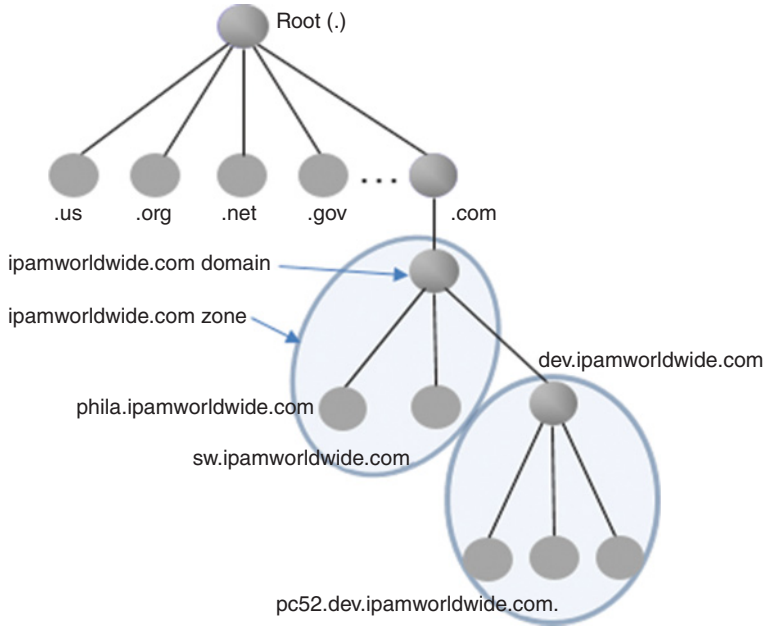

Figure 2.3. Zones as Delegated Domains

By delegating authority for dev.ipamworldwide.com, the DNS administrators for ipamworldwide.com agree to pass all resolutions for dev.ipamworldwide.com (and below in the domain tree for any subdomains of dev.ipamworldwide.com) to DNS servers administered by personnel operating the dev.ipamworldwide.com zone. These dev.ipamworldwide.com administrators can manage their domain and resource records and any children autonomously; they just need to inform the parent domain administrators (for ipamworldwide.com) where to direct queries they receive as resolvers or other DNS servers attempt to traverse down the domain tree seeking resolutions.

Thus, administrators for the ipamworldwide.com zone must configure all resource records and configuration attributes for the ipamworldwide.com zone, including subdomains within the ipamworldwide.com zone such as the sw.ipamworldwide.com domain. At the same time, ipamworldwide.com administrators must provide a delegation linkage to any child zones, such as dev.ipamworldwide.com. This delegation linkage is supported by entering name server (NS) resource records within the ipamworldwide.com zone file, which indicate which name servers are authoritative for the dev.ipamworldwide.com delegated zone. These NS records provide the continuity to delegated child zones by referring resolvers or other name servers further down the domain tree. Corresponding A or AAAA records called *glue records* are also typically defined to glue the resolved NS host domain name to an IP address to enable direct addressing of further queries.

The encircled domain nodes in Figure 2.3 indicate that the ipamworldwide.com zone contains the ipamworldwide.com domain plus two of its children, phila and sw. The ipamworldwide.com DNS administrators are responsible for maintaining all DNS configuration information for the ipamworldwide.com zone including subdomains managed within the ipamworldwide.com zone, as well as referrals to DNS servers serving delegated child zones, such as dev.ipamworldwide.com as shown in Figure 2.3. Thus, when other DNS servers around the world are attempting to resolve any domain names ending in ipamworldwide.com on behalf of their clients, their queries will require traversal of the ipamworldwide.com DNS servers and perhaps other DNS servers, such as those serving the dev.ipamworldwide.com zone.

The process of delegation of the name space enables autonomy of DNS configuration while providing linkages via NS record referrals within the global DNS database. As you can imagine, if the name servers referenced by these NS records are unavailable, the domain tree will be broken at that point, inhibiting resolution of names at that point or below in the domain tree. If the dev.ipamworldwide.com DNS servers are down, authoritative resolution for dev.ipamworldwide.com *and its children* will fail. This illustrates the requirement that each zone must have at least two authoritative DNS servers for redundancy.

In summary, the administrators for the ipamworldwide.com zone will configure their DNS servers with configuration and resource record information for the ipamworldwide.com, phila.ipamworldwide.com, and sw.ipamworldwide.com domains. They will also need to configure their servers with just the names and addresses of DNS servers serving delegated child zones, particularly dev.ipamworldwide.com in this case. They need know nothing further about these delegated domains; just where to refer the querying recursive DNS server during the resolution process.

Dissemination of Zone Information

Given the criticality of the DNS service in resolving authoritatively and maintaining domain tree linkages, DNS server redundancy is a must. DNS server configuration information consists of server configuration parameters and declarations of all zones for which the server is authoritative. This information can be defined on each server that is authoritative for a given set of zones. Additions, changes, and deletions of resource records, the discrete resolution information within each zone configuration file, can be entered once on a *master* server, or more correctly, the server that is configured as master for the respective zone. The other servers that are likewise authoritative for this information can be configured as *slaves* or *secondaries*, and they obtain zone updates by the process of *zone transfers*. Zone transfers enable a slave server to obtain the latest copy of its authoritative zone information from the master server. Microsoft Active Directory-integrated DNS servers support zone transfers for compatibility with this standard process, but also enable DNS data replication using native Active Directory replication processes.

Versions of zone files are tracked by a zone serial number which must be updated every time a change is applied to the zone. Slaves are configured to periodically check

the zone serial number set on the master server; if the serial number is larger than its own value defined for the zone, it will conclude that it has outdated information and will initiate a zone transfer. Additionally, the server that is master for the zone can be configured to *notify* its slaves that a change has been made, stimulating the slaves to immediately check the serial number and perform a zone transfer to obtain the updates more quickly than awaiting the normal periodic update check.

Zone transfers may consist of the entire zone configuration file, called an absolute zone transfer (AXFR) or of the incremental updates only, called an incremental zone transfer (IXFR). In cases where zone information is relatively static and updated from a single source, for example, an administrator, the serial number checking with AXFRs as needed works well. These so-called *static zones* are much simpler to administer than their counterpart: *dynamic zones*. Dynamic zones, as the name implies, accept dynamic updates, for example, from DHCP servers updating DNS with newly assigned IP addresses and corresponding host domain names. Updates for dynamic zones can utilize IXFR mechanisms to maintain synchronization among the master and multiple slave servers.

The popular BIND DNS reference implementation utilizes journal files on each server to provide an efficient means to track dynamic updates to zone information. These journal files are temporary appendages to corresponding zone files and enables tracking of dynamic updates until the server writes these journal entries into the zone file and reloads the zone. Many server implementations load the zone file information into memory along with incremental zone updates, which are also loaded into memory for fast resolution. Other approaches to storing zone information include use of a database such as is the case with the PowerDNS and Knot DNS implementations.

Additional Zones

Root Hints We mentioned a *hints file* during the overview of the resolution process. This file should provide a list of DNS server names and addresses (in the form of NS, A, and AAAA resource records) that the server should query if the resolver query cannot be resolved via authoritative, forwarded, or cached data. The hints file will typically list the Internet root servers, which are authoritative for the Internet root (.) of the domain tree. Querying a root server enables the recursive server to start at the top to begin the traversal down the domain tree in order to locate an authoritative server to resolve the query. The contents of the hints file for Internet root servers may be obtained from www.internic.net/zones/named.root, though all major DNS server implementations include this file with their distributions.

Some environments may require use of an internal set of root servers, where Internet access is restricted by organizational policy. In such cases, an internal version of the hints file can be used, listing names and addresses of internal root servers instead of the Internet root servers as we'll discuss in Chapter 5. The organization itself would need to maintain the listing of internal root servers, as well as their requisite root zone configurations.

Localhost Zones Another zone file that proves essential is the localhost zone. The localhost zone enables one to resolve "localhost" as a hostname on the given server. A corresponding in-addr.arpa. zone file resolves the 127.0.0.1 loopback address. A single entry within the 0.0.127.in-addr.arpa zone maps this loopback address to the host itself. This zone is required as there is no upstream authority for the 127.in-addr.arpa domain or subdomains. Likewise, the IPv6 equivalents need to be defined for the corresponding IPv6 loopback address,::1. The localhost zone simply maps the localhost hostname to its 127.0.0.1 or::1 IP address using an A and AAAA record, respectively. Please refer to *IP Address Management Principles and Practice* (8) for a full discussion of this and reverse zones in general.

RESOLVER CONFIGURATION

DNS does require some basic client configuration prior to use. This initial configuration may be performed manually or by obtaining this information from a DHCP server. The client configuration consists of configuring resolver software regarding which DNS server(s) to query for resolution.

Figure 2.4 illustrates the configuration of a Microsoft Windows resolver in terms of manually defining the DNS server to query or the use of DHCP to obtain DNS server addresses automatically. Though your screen shots may look a bit different, these screens for Windows have looked nearly identically over the last several versions of Windows.

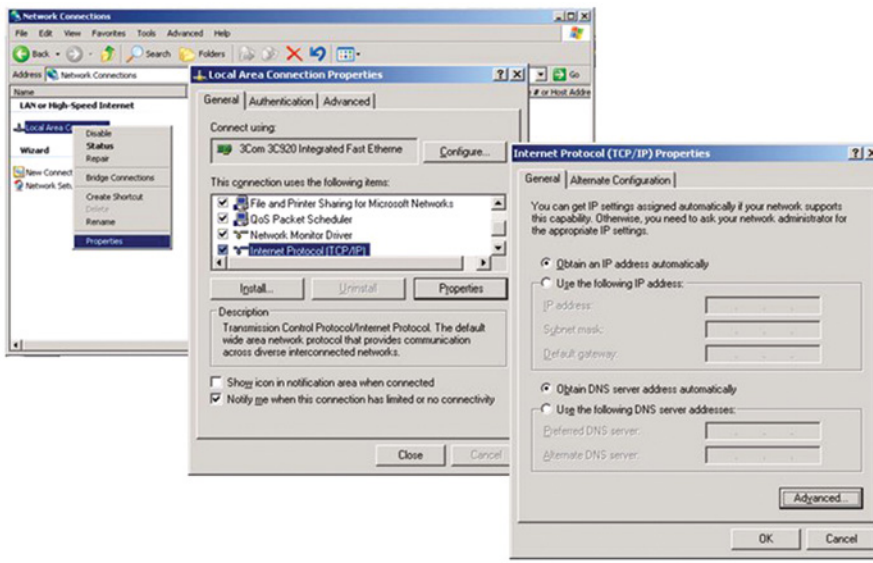

Figure 2.4. Microsoft Windows Configuration of IP Address DNS Servers to Query

Microsoft Windows enables entry of multiple DNS servers to query within its graphical interface. Notice there are two entries in the "brute force" method shown on the screen on the right of Figure 2.4, one for preferred and another for alternate. Clicking the Advanced tab enables entry of more than two and in particular order. We recommend having *at least* two DNS servers configured for the resolver in the event one DNS server is unavailable, the resolver will automatically query the alternative server. If the "Obtain DNS server address automatically" radio button is selected, as shown in Figure 2.4, the resolver will obtain a list of DNS servers typically via DHCP.

On Unix or Linux based systems, the `/etc/resolv.conf` file can be edited to configure the resolver. The key parameter in this file is one or more `nameserver` statements pointing to DNS servers, but a number of options and additional directives enable further configuration refinement as described below. The italicized text should be replaced by actual data referenced; for example, *`domain`* should be replaced with a DNS domain name.

- `nameserver` *`IP_address`* – the IP address of a recursive DNS server to query for name resolution; multiple `nameserver` entries are allowed and encouraged. The `nameserver` entry instructs the resolver where to direct DNS queries.
- `domain` *`domain`* – the DNS domain where this host (on which this resolver is installed) resides. This is used when resolving relative hostnames, as opposed to fully qualified host domain names.
- `search` *`domain(s)`* – the search list of up to six domains in which to search the entered hostname for resolution. Thus if we type in `www` for resolution, the resolver will successively append domains configured in this parameter in an attempt to resolve the query. If the entry `search ipamworldwide.com.` exists in resolv.conf, entry of `www` will result in a resolution attempt for www.ipamworldwide.com.
- `sortlist` *`address/mask list`* – enables sorting of resolved IP addresses in accordance with the specified list of address/mask combinations. This enables the resolver to choose a "closer" destination if multiple IP addresses are returned for a query.
- `options` – keyword preceding the following which enables specification of corresponding resolver parameters including the following:
  - `debug` – turns on debugging
  - `ndots` *`n`* – defines a threshold for the number of dots within the entered name required before the resolver will consider the entered name simply a hostname or a qualified domain name. When considered a hostname, the hostname will be queried as appended with domain names specified within the `domain` or `search` parameter.
  - `timeout` *`n`* – number of seconds to wait before timing out a query to a DNS server.

- `attempts` *n* – number of query attempts before considering the query a failure.
- `rotate` – enables round robin querying among DNS servers configured within the `nameserver` directives. Queries will be sent to a different server each time and cycled through.
- `no-check-names` – turns off name checking of entered host names for resolution. Normally, underscore characters are not permitted, for example, so setting this option enables query processing to proceed without validation of the entered hostname.
- `inet6` – causes the resolver to issue a query for a AAAA record to resolve the entered hostname before attempting an A record query.

`search` and `options` settings can also be overridden on a per process basis via corresponding environment variable settings.

SUMMARY

This chapter described the organization of DNS data in a tree structure where each node of the tree may be managed by independent entities. We also discussed the various formats for looking up information as well as the basic types of zone files generally configured on DNS servers. Lastly we provided a brief overview of device resolver configuration. This background serves as a foundation for understanding the structure of "data at rest" on DNS servers and resolvers, as well as the process for locating DNS information within the global DNS tree. Now onto the next chapter to review DNS data in motion, the DNS protocol.