9

THE DOMAIN NAME SYSTEM (DNS) PROTOCOL

9.1 DNS OVERVIEW—DOMAINS AND RESOLUTION*

DNS is the third cornerstone of IPAM and a foundational element of IP communications. DNS provides the means for improved usability of IP applications, insulating end users from typing IP addresses directly into applications like web browsers. Certainly, to communicate over an IP network, an IP device needs to send IP packets to the intended destination IP device; and as we have seen, the IP packet headers require source and destination IP addresses. DNS provides the translation from a user-entered named destination, for example, web site address, to its IP address.

As a network service, DNS has evolved from simple host name-to-IP address lookup utility to enabling very sophisticated "lookup" applications supporting voice, data, multimedia, and security applications. DNS has proven extremely scalable and reliable

* Initial sections of this chapter are based on Chapter 4 of Ref. 11.

IP Address Management: Principles and Practice, by Timothy Rooney

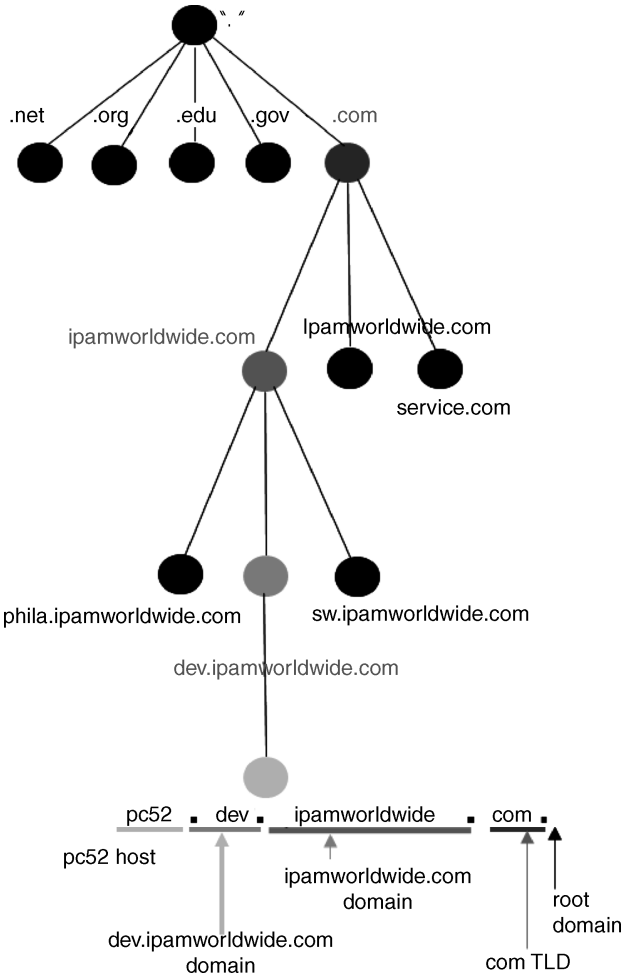

Figure 9.1. Domain tree mapping to a fully qualified domain name (11)

for such lookup functions. We'll discuss how this lookup process works after first introducing how this information is organized.

9.1.1 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 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 children domains and so on.

As we read between the dots from right to left, we can identify a unique path to the host we are seeking. The text left of the leftmost dot is generally* the hostname, 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 9.1 illustrates a fully qualified domain name 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 legal and easier shorthand notation, but just be aware of the potential ambiguity.

9.2 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 assume I'd like to connect to a device named `pc52` per the example in Figure 9.1. Thus I enter the host domain name, `pc52.dev.ipamworldwide.com.` as my intended destination. The application into which I type this domain name (e.g., email client and web browser) 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 web server name I typed into an IP address that may be used to initiate IP communications.

The resolver issues a query for this hostname to my local 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 servers 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 the Figure 9.2.

We often 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 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

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

† 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.

‡ We'll review how to perform this manual configuration a little later in this chapter.

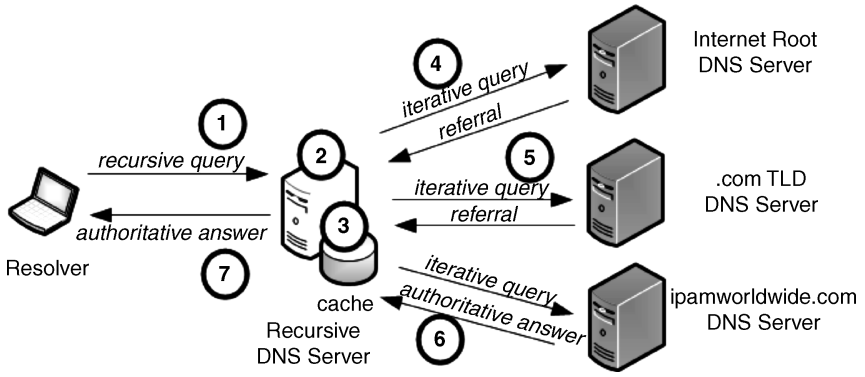

Figure 9.2. Recursive and iterative queries in name resolution (11)

queries directly from client resolvers and performs the steps outlined below to obtain the answer to the query on behalf of the resolver.

1. The resolver initiates a query to the recursive DNS server. 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 configuration and resource record information for which it is *authoritative*. This information is typically configured using text files, a Windows interface or an 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, 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 `pc52.dev.ipamworldwide.com.` resides in cache*, the DNS server will respond to the resolver with this nonauthoritative 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 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

* Cache entries are temporary and are removed by DNS servers based on user configuration settings as well as advertised lifetime of a resource record.

server, server A, may have indicated that DNS server A is authoritative for the `ipamworldwide.com` domain. The initially queried DNS server may then query DNS server A for information leading to resolution of `pc52.dev.ipamworldwide.com`.

b. If the cache does not provide relevant information, and the queried recursive server is configured to forward, the server will forward the query as configured in its configuration or zone file. We'll cover the details on this configuration later.

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.

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, like those within end user clients, that are configured only with which recursive name servers (NSs) to query.

5. 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 IP address(es) for `pc52.dev.ipamworldwide.com.` or will provide a referral to another DNS server further down the hierarchy "closer" to the sought fully qualified domain name. For example, upon querying a root server, you are guaranteed that you will not obtain a direct resolution answer for `pc52.dev.ipamworldwide.com.` However, the root name server will refer the querying DNS server to the name servers that are authoritative for `com.` The root servers are "delegation-only" servers and do not directly resolve queries, only answering with delegated name server information for the queried TLD.
6. The recursive server *iterates*† additional queries based on responses down the domain tree until the query can be answered. Continuing with our example, upon querying the name server that is authoritative for `com.`, the answer received will be a referral to the name server that is authoritative for `ipamworldwide.com.`, and so on down the tree. Ultimately, a DNS server that is authoritative for the zone of relevance to the query should be located. The authoritative DNS server will read the corresponding zone information for a *resource record* of the type being queried. The server will pass the resource record(s) to the querying (recursive) DNS server.

* 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.

7. When the answer is received, the recursive DNS server will provide the answer to the resolver and also update its cache and the process ends. If an answer cannot be found, the recursive server will also cache this "negative" information as well for use in responding to similar queries.

In summary, the resolution process entails (a) finding a name server with authoritative information to resolve the query in question and (b) querying that server for the desired information. In our example, the desired information was the IP address corresponding to the domain name pc52.dev.ipamworldwide.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 *RRSet*, matching the queried value (name), class and type. We'll discuss resource records in detail in the next chapter.

The bottom line is that DNS servers are configured at all levels of the domain tree as authoritative for their respective domain information, as well as where to refer queriers further down the domain tree. In many 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 within an organization 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 `dev.ipamworldwide.com`. domain, but to delegate `eng.ipamworldwide.com` to a different set of administrators and name servers. This leads us to a discussion regarding the distinction between zones and domains.

9.3 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` and `dev.ipamworldwide.com` domains, while the `eng.ipamworldwide.com` zone retains authority for the `eng.ipamworldwide.com` domain as illustrated in Figure 9.3.

By delegating authority for `eng.ipamworldwide.com`, the DNS administrators for `ipamworldwide.com` are agreeing to pass all resolutions for `eng.ipamworldwide.com` (and below in the domain tree for any subdomains of `eng.ipamworldwide.com`) to DNS servers administered by personnel operating the `eng.ipamworldwide.com` zone. These `eng.ipamworldwide.com`

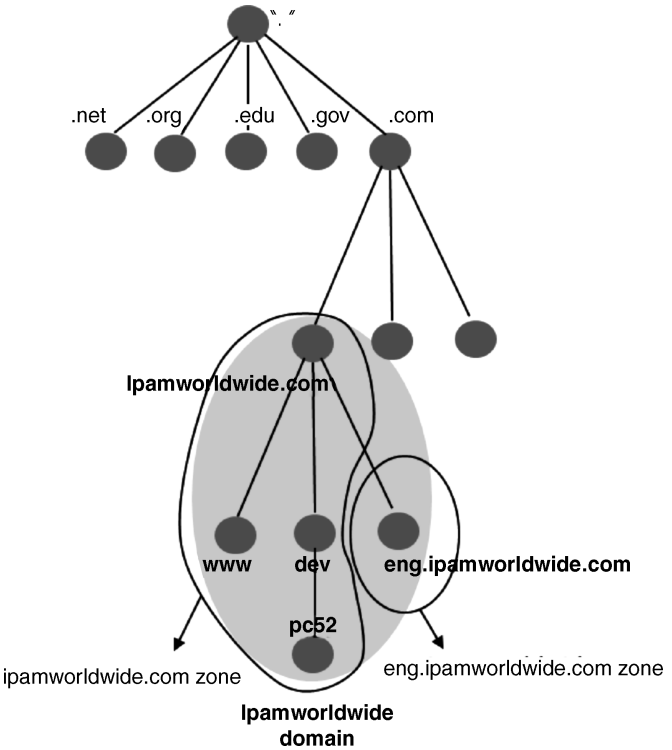

Figure 9.3. Zones as delegated domains (11).

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 `dev.ipamworldwide.com` domain. At the same time, `ipamworldwide.com` administrators must provide a delegation linkage to any child zones, such as `eng.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 `eng.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 usually defined to glue the resolved NS host domain name to an IP address to enable direct addressing of further queries.

The shading in Figure 9.3 indicates that the `ipamworldwide.com` domain contains the `ipamworldwide.com` node plus all of its children, highlighting this level

of responsibility for ipamworldwide.com and "below." The `ipamworldwide.com` DNS administrators are responsible for maintaining all DNS configuration information for the `ipamworldwide.com` zone as well as referrals to DNS servers serving delegated child zones. Thus, when other DNS servers around the world are attempting to resolve any name 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 `eng.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 `eng.ipamworldwide.com` DNS servers are down, authoritative resolution for `eng.ipamworldwide.com` *and its children* will fail. This illustrates the requirement that each zone must have at least two authoritative DNS servers for redundancy.

Thus, the administrators for the `ipamworldwide.com` zone will configure their DNS servers with configuration and resolution information for the `ipamworldwide.com` and `dev.ipamworldwide.com` domains; they will also configure their servers with the names and addresses of DNS servers serving delegated or child zones. They need know nothing further about these delegated zones; just who to contact so a referral can be sent to the querying recursive DNS server.

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.

9.3.1 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. Different DNS server vendors take different redundancy approaches. Microsoft replicates DNS information among a set of domain controllers when DNS information is integrated into Active Directory, an architectural foundation of the Windows Server products. The ISC BIND implementation supports DNS information replication through a hub-and-spoke model. Configuration changes are made to a *master* DNS server as mentioned above. Redundant DNS servers are 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 also support zone transfers for compatibility with this standard process.

Versions of zone files are tracked by a zone serial number that must be changed 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 domain names. Updates for dynamic zones can utilize IXFR mechanisms to maintain synchronization among the master and multiple slave servers.

With BIND 9, journal files on each server 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. We'll discuss more details about server and zone configuration later in the chapter, but first let's consider a different kind of domain tree structure.

9.3.2 Reverse Domains

Until now, we've introduced the common name-to-IP address resolution process, locating a DNS server authoritative for a name resolution, which then responds authoritatively to the query. Another popular form of query is for IP address-to-name resolution. This "reverse" form of resolution is commonly used as a security check when establishing virtual private network (VPN) connections or for general IP address-to-hostname lookups. Given an IP address, how does a DNS server traverse the domain tree to find a host domain name? Special top-level domains are defined for IP address-based domain trees within the *Address and Routing Parameter Area* (arpa) domain: `in-addr.arpa.` is defined for IPv4 address-to-name resolution and `ip6.arpa.` is for IPv6 address-to-name resolution[*].

[*] Technically, these are both children of the .arpa. TLD

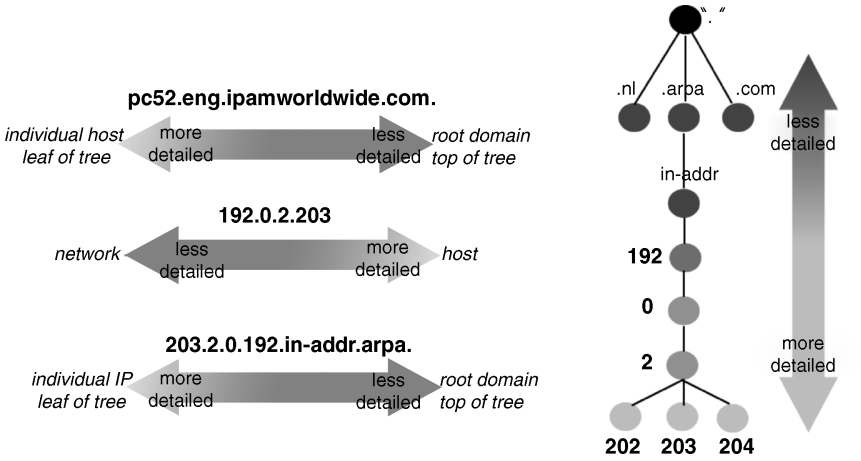

Figure 9.4. IP address (reverse) domain tree mapping (11).

The only wrinkle in organizing IP addresses within a domain tree results from mapping an IP address, which reads left-to-right as less detailed (network) to more detailed (IP host), while reading a domain name left-to-right reads more specific (specific host, domain) to less specific (root). Therefore, the IP address is reversed to enable representation within the domain hierarchy, reading left-to-right as more specific to less specific. This is illustrated in Figure 9.4.

You may notice that the mapping of dotted decimal notation enables mapping of reverse domains to octet-boundary-based network allocations. For example, if we've been allocated a "class C" network as our public space, 192.0.2.0/24, it is easy* to visualize the leaves of the `in-addr.arpa.` domain tree depicted above mapping to individual hosts. And like resolution of hostnames, traversal of the `in-addr.arpa.` domain tree follows a similar process to locate authoritative resolution of address-to-name queries. The pointer or PTR resource record provides a mapping from address to host, as we'll discuss in the next chapter.

But what if we had been allocated a subnet on non-octet boundaries? For example, if we had allocated a /23 instead of a /24, the network address might have been denoted 192.0.2.0/23. This /23 is in effect comprised of the 2 /24 networks: 192.0.2.0/24 and 192.0.3.0/24. The two reverse domains corresponding to these octet-normalized network addresses, `2.0.192.in-addr.arpa.` and `3.0.192.in-addr.arpa`, would need to be configured within DNS to allow reverse lookups of hosts within this /23 network.

If the allocated subnet was smaller than a class C network, a more complex representation and zone file configuration is required. Let's say for example that we allocate a subnet for a remote office as 192.0.2.0/25. If we try to represent the corresponding reverse domain as `2.0.192.in-addr.arpa`, this would encompass

* Of course "easy" is a relative term, but once you get accustomed to reverse domains, at least such classful networks are easily visualized as reverse domains.

the desired half but also the "other half" of the 192.0.2.0/24 network, namely the 192.0.2.128/25 network. But this other half could be allocated to a different organization having its own DNS authority. In that case, who would administer the classful reverse zone since it's split across two authorities? The solution is to indicate that portion of the fourth octet to which the subnet applies in the reverse zone name.

RFC 2317 (93) specifies the use of the CIDR notation within the `in-addr.arpa` zone name. Thus, literally reversing the numbers between the dots of the allocated subnet, we arrive at the following: for network 192.0.2.0/25, the corresponding reverse domain is `0/25.2.0.192.in-addr.arpa.`* The "other half" of this class C would be `128/25.2.0.192.in-addr.arpa.` Subnets of smaller sizes would follow a similar notation, using the fourth octet of the network address, followed by/<network size>, followed by the remaining three octets from the IP address, reversed, then appended with `in-addr.arpa.`

But when a resolver issues a query, it will be for a particular address (PTR record) in the form of `185.2.0.192.in-addr.arpa.`, so how do we map this query to the appropriate zone file, `128/25.2.0.192.in-addr.arpa` in this case? The solution calls for the use of canonical name (CNAME) records in the parent (`2.0.192.in-addr.arpa.`) zone to selectively point to the proper delegated zone, each of which may be administered by separate DNS administrators. A CNAME record serves as an alias for a given record, directing the querier to then query for the alias name. In this case a CNAME record for each individual IP address needs to be created to map to a corresponding RFC 2317 style reverse zone, enabling the delegation of subsets of records to different subzone administrators.

Let's look at how this would work in our example case. Within the parent zone file corresponding to this `2.0.192.in-addr.arpa.` zone, we would configure the following†:

```
2.0.192.in-addr.arpa. IN SOA dns.ipamworldwide.com.
admin.ipamworldwide.com. ( 1 2h 30m 1w 1d )

$ORIGIN 2.0.192.in-addr.arpa.          //implicit
0/25  IN NS dns.A1.ipamworldwide.com.  //authoritative servers
      IN NS dns.A2.ipamworldwide.com.  // for 0/25
1     IN CNAME 1.0/25.2.0.192.in-addr.arpa.
2     IN CNAME 2.0/25.2.0.192.in-addr.arpa.
3     IN CNAME 3.0/25.2.0.192.in-addr.arpa.
. . .
127    IN CNAME 127.0/25.2.0.192.in-addr.arpa.
```

* While RFC 2317 specifies slashes within these domain names, many DNS administrators substitute dashes in order to associate zone names with zone file names, which cannot contain slashes. Hence we could denote this zone as 0-25.2.0.192.in-addr.arpa defined in zone file db.0-25.2.0.192.in-addr.arpa. We'll stick to the RFC 2317 format here, but dashes work just as well.

† DNS configuration file examples in this book utilize BIND DNS format (144).

```
128/25 IN NS dns.B1.ipamworldwide.com.  //authoritative servers
       IN NS dns.B2.ipamworldwide.com.  // for 128/25
129    IN CNAME 129.128/25.2.0.192.in-addr.arpa.
130    IN CNAME 130.128/25.2.0.192.in-addr.arpa.
131    IN CNAME 131.128/25.2.0.192.in-addr.arpa.
. . .
254    IN CNAME 254.128/25.2.0.192.in-addr.arpa.
```

Based on standard domain tree traversal, when the querying name server queries the DNS server authoritative for the `2.0.192.in-addr.arpa.` zone, the file above on the corresponding DNS server provides not a resolution, but a next step, pointing the desired IP address answer to another FQDN via a CNAME record. So far in the process, a query for the hostname for IP address 192.0.2.185 would result in a CNAME pointing to `185.128/25.2.0.192.in-addr.arpa.` We also know who to ask to resolve this query because two NS records are listed as authoritative for the associated domain, `128/25.2.0.192.in-addr.arpa.`, namely `dns.B1.ipamworldwide.com` and `dns.B2.ipamworldwide.com`.

The corresponding `128/25.2.0.192.in-addr.arpa.` zone file on these servers would contain the following:

```
128/25.2.0.192.in-addr.arpa.  IN  SOA  dns.B1.ipamworldwide.com.
admin.ipamworldwide.com. ( 1 2h 30m 1w 1d )

128/25.2.0.192.in-addr.arpa. IN NS dns.B1.ipamworldwide.com.
128/25.2.0.192.in-addr.arpa. IN NS dns.B2.ipamworldwide.com.

129.128/25.2.0.192.in-addr.arpa.  IN  PTR  public1.ipamworldwide.
com.
130.128/25.2.0.192.in-addr.arpa.  IN  PTR  public2.ipamworldwide.
com.
131.128/25.2.0.192.in-addr.arpa. IN PTR www.ipamworldwide.com.
```

Or in abbreviated format using "relative" domain names:

```
@ IN SOA dns.B1.ipamworldwide.com. admin.ipamworldwide.com. ( 1 2h 30m
1w 1d )
// Implicit $ORIGIN 128/25.2.0.192.in-addr.arpa.

   IN NS dns.B1.ipamworldwide.com.
   IN NS dns.B2.ipamworldwide.com.
129  IN PTR public1.ipamworldwide.com.
130  IN PTR public2.ipamworldwide.com.
```

```
131  IN PTR www.ipamworldwide.com.

. . .
185  IN PTR server-x.ipamworldwide.com.
```

Querying this zone file for this referenced CNAME alias, to `185.128/25.2.0.192.in-addr.arpa.`, we find our PTR record pointing to the associated hostname server-x.ipamworldwide.com, completing the resolution.

For non-octet bounded networks larger than class C networks (i.e., /9 - /15 and /17 - /23), domain alias (DNAME) records can be used. For example, the 172.16.0.0/14 network could be allocated and delegated to an administrator in the engineering group. Reverse queries on this network can be referred to the engineering group's DNS server, dns[1-2].eng.ipamworldwide.com per the following example, configured within the `172.in-addr.arpa.` zone file:

```
16/14.172.in-addr.arpa. IN NS   dns1.eng.ipamworldwide.com
16/14.172.in-addr.arpa. IN NS   dns2.eng.ipamworldwide.com
16.172.in-addr.arpa.    IN DNAME 16.16/14.172.in-addr.arpa.
17.172.in-addr.arpa.    IN DNAME 17.16/14.172.in-addr.arpa.
18.172.in-addr.arpa.    IN DNAME 18.16/14.172.in-addr.arpa.
19.172.in-addr.arpa.    IN DNAME 19.16/14.172.in-addr.arpa.
```

These entries delegate the reverse lookups for all four /16 networks comprising the engineering group's /14 to the their DNS servers as indicated by the first two records shown above. The next four records map these four /16 reverse domains to this delegated 16/14.172.in-addr.arpa. domain.

We've essentially inserted an artificial layer in the reverse tree to serve as a consolidation point. Thus, to resolve the PTR record for a host with IP address 172.18.45.94, the resolving name server would traverse down the 172.in-addr.arpa. tree. The next node down, 18.172.in-addr.arpa., has a domain alias of 18.16/14.172.in-addr.arpa. by virtue of the DNAME lookup. Next, by querying the dns1.eng.ipamworldwide.com DNS server, which is authoritative for the 16/14.172.in-addr.arpa. zone, we resolve the corresponding PTR entry within this zone:

```
94.45.18.172.in-addr.arpa. IN PTR host.eng.ipamworldwide.com.
```

9.3.3 IPv6 Reverse Domains

IPv6 reverse domain mapping is a bit more cumbersome. As with IPv4, the IPv6 address must be reversed, maintaining its hexadecimal format. But the IPv6 address must first be "padded" to the full 32-hex digit representation; that is, the two forms of abbreviation discussed in Chapter 2 must be removed by including leading zeroes between colons and filling in double-colon-denoted implied zeroes. Figure 9.5 illustrates an example of the

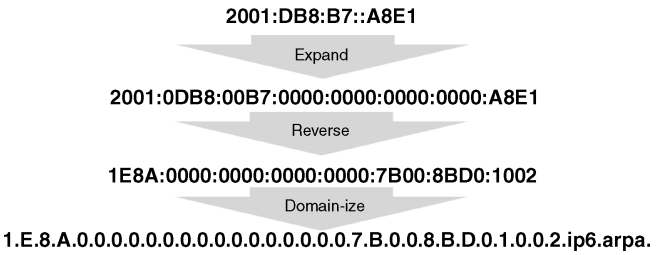

Figure 9.5. IPv6 address to reverse domain mapping.

process for the IPv6 address 2001:DB8:B7::A8E1. The address must be expanded or padded and the digits reversed. Then, this result must be "domain-ized" by removing the colons, inserting dots between each digit, and appending the ip6.arpa. upper level domains.

Figure 9.6 illustrates the logic in reversing the IPv6 address in order to be represented in a domain hierarchy as read left-to-right as more specific to less specific. This is directly analogous to Figure 9.4, which illustrates this concept for IPv4 addresses. The full 32-hex digit representation used in Figure 9.6 provides a unique, though lengthy, traversal down the `ip6.arpa.` domain tree (not shown).

Note that this example illustrates the reverse domain representation for a full 128-bit IPv6 address. Subnets can have corresponding reverse domain definitions as in IPv4. For a /64 allocation, only the first 64 bits (16-hex digits) would be included. Thus, for the host above, its /64 subnet reverse zone notation would be defined as

```
0.0.0.0.7.B.0.0.8.B.D.0.1.0.0.2.ip6.arpa.
```

Notation for reverse domains of IPv6 networks allocated on non-nibble boundaries was not formally addressed in RFC 2317; however, the same techniques specified in the RFC can be mapped to IPv6 reverse zones corresponding to non-nibble bounded IPv6 block allocations. Let's illustrate this by example. Say the North America team desires to allocate four /54 blocks from its 2001:db8:4af0:8000::/52 block, namely 2001:

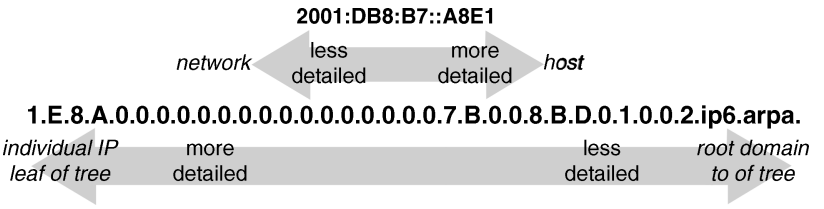

Figure 9.6. The IPv6 reverse domain notation.

db8:4af0:8000::/54, 2001:db8:4af0:8400::/54, 2001:db8:4af0:8800::/54, and 2001:db8:4af0:8c00::/54. Using CNAME resource records to refer queriers to servers responsible for these corresponding reverse zones, the 8.0.f.a.4.8.b.d.0.1.0.0.2.ip6.arpa zone file would look something like

```
8.0.f.a.4.8.b.d.0.1.0.0.2.ip6.arpa. IN SOA dns.ipamworldwide.com.
admin.ipamworldwide.com. ( 1 2h 30m 1w 1d )

$ORIGIN 8.0.f.a.4.8.b.d.0.1.0.0.2.ip6.arpa. //implicit
0/54  IN NS dns.A1.ipamworldwide.com.  //authoritative servers
      IN NS dns.A2.ipamworldwide.com.  // for 2001:db8:4af0:8000::
      /54
0     IN CNAME 0.0/54.8.0.f.a.4.8.b.d.0.1.0.0.2.ip6.arpa.
1     IN CNAME 1.0/54.8.0.f.a.4.8.b.d.0.1.0.0.2.ip6.arpa.
2     IN CNAME 2.0/54.8.0.f.a.4.8.b.d.0.1.0.0.2.ip6.arpa.
3     IN CNAME 3.0/54.8.0.f.a.4.8.b.d.0.1.0.0.2.ip6.arpa.

4/54  IN NS dns.B1.ipamworldwide.com.     //authoritative servers
      IN NS dns.B2.ipamworldwide.com.     // for 2001:db8:4af0:
      8400::/54
4     IN CNAME 4.4/54.8.0.f.a.4.8.b.d.0.1.0.0.2.ip6.arpa.
5     IN CNAME 5.4/54.8.0.f.a.4.8.b.d.0.1.0.0.2.ip6.arpa.
6     IN CNAME 6.4/54.8.0.f.a.4.8.b.d.0.1.0.0.2.ip6.arpa.
7     IN CNAME 7.4/54.8.0.f.a.4.8.b.d.0.1.0.0.2.ip6.arpa.

8/54  IN NS dns.C1.ipamworldwide.com.  //authoritative servers
      IN NS dns.C2.ipamworldwide.com. // for 2001:db8:4af0:8800::/54
8     IN CNAME 8.8/54.8.0.f.a.4.8.b.d.0.1.0.0.2.ip6.arpa.
9     IN CNAME 9.8/54.8.0.f.a.4.8.b.d.0.1.0.0.2.ip6.arpa.
a     IN CNAME a.8/54.8.0.f.a.4.8.b.d.0.1.0.0.2.ip6.arpa.
b     IN CNAME b.8/54.8.0.f.a.4.8.b.d.0.1.0.0.2.ip6.arpa.

c/54  IN NS dns.D1.ipamworldwide.com.  //authoritative servers
      IN NS dns.D2.ipamworldwide.com. // for 2001:db8:4af0:8c00::/54
c     IN CNAME c.c/54.8.0.f.a.4.8.b.d.0.1.0.0.2.ip6.arpa.
d     IN CNAME d.c/54.8.0.f.a.4.8.b.d.0.1.0.0.2.ip6.arpa.
e     IN CNAME e.c/54.8.0.f.a.4.8.b.d.0.1.0.0.2.ip6.arpa.
f     IN CNAME f.c/54.8.0.f.a.4.8.b.d.0.1.0.0.2.ip6.arpa.
```

Following standard domain tree traversal, when the querying name server queries the DNS server authoritative for the `8.0.f.a.4.8.b.d.0.1.0.0.2.ip6.arpa.` zone, the file above on the corresponding DNS server provides not a resolution, but a next step, pointing the desired IPv6 address answer to another FQDN via a CNAME record. So far in the process, a PTR query requesting the hostname for IP address 2001:db8:4af0:8d03::f6 results in a CNAME pointing to `d.c/54.8.0.f.a.4.8.b.d.0.1.0.0.2.ip6.arpa.` We also know who to ask to resolve this query because two NS records are listed as authoritative for this domain, namely `dns.D1.ipamworldwide.com` and `dns.D2.ipamworldwide.com`.

The corresponding `d.c/54.8.0.f.a.4.8.b.d.0.1.0.0.2.ip6.arpa.` zone file on these servers would contain the following:

```
c/54.8.0.f.a.4.8.b.d.0.1.0.0.2.ip6.arpa. IN SOA dns.D1.ipamworld-
wide.com. admin.ipamworldwide.com. ( 1 2h 30m 1w 1d )

   IN NS dns.D1.ipamworldwide.com.
   IN NS dns.D2.ipamworldwide.com.

1.0.b.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.3.0.c   IN PTR public1.ipamworld-
                                                wide.com.
0.2.0.a.4.0.0.0.0.0.0.0.0.0.0.0.0.0.0.3.0.c   IN PTR public2.ipamworld-
                                                wide.com.
f.c.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.3.0.d   IN PTR www.ipamworldwide.
                                                com.
. . .
6.f.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.3.0.d   IN PTR server-y.ipamworld-
                                                wide.com.
```

Querying this zone file for this referenced CNAME alias, that is, `6.f.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.3.0.d.c/54.8.0.f.a.4.8.b.d.0.1.0.0.2.ip6.arpa.`, we find our PTR record pointing to the associated hostname server-y.ipamworldwide.com, completing the resolution.

9.3.4 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 root (.) of the domain tree. Querying a root server enables the querying 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 BIND and Microsoft DNS server implementations include this file with their distributions.

As we'll discuss in Chapter 11, 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. 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 address 1 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.

9.4 RESOLVER CONFIGURATION

Like DHCP transactions, DNS resolution occurs behind the scenes and involves a client and server. Ideally, end users don't even know it happens; they type in a web address and connect! The resolver software must be configured regarding which DNS server(s) to query for resolution. Thus, unlike DHCP that requires no initial client configuration (since it simply broadcasts or multicasts to a well-known address), 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.

Figure 9.7 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.

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 9.7, 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 so should a DNS server be out of service, the resolver will automatically query an alternate server. If the "Obtain DNS server address automatically" radio button is selected, as shown in Figure 9.7, the resolver will obtain a list of DNS servers via DHCP.

On Unix or Linux-based systems, the `/etc/resolv.conf` file can be edited with 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

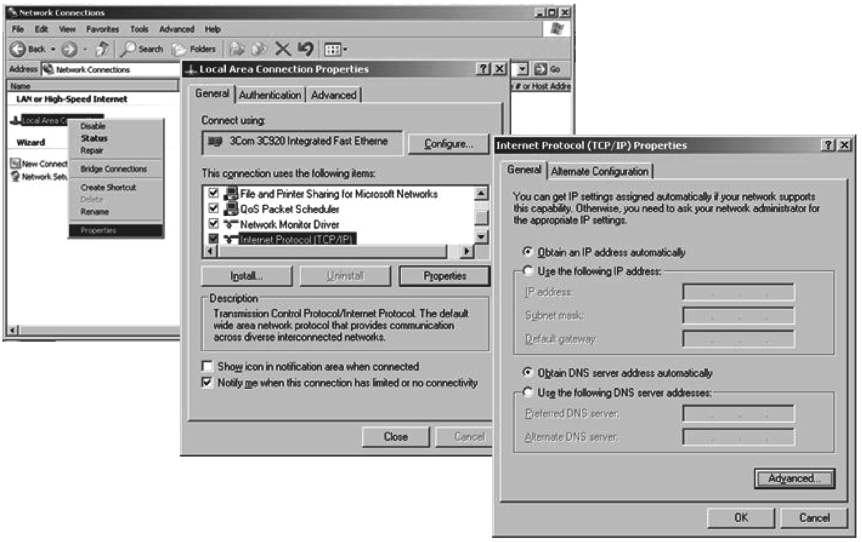

Figure 9.7. Microsoft Windows configuration of IP address DNS servers to query.

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 that 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 hostnames 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.

9.5 DNS MESSAGE FORMAT

9.5.1 Encoding of Domain Names

So far, we've discussed the organization of DNS information into a domain hierarchy as well as the basics of how a client or resolver performs resolution by issuing a recursive query to a DNS server that in turn iterates the query in accordance with the domain hierarchy (or forwarding) to obtain the answer to the query. Next we'll dig deeper into the DNS query and general message format, but first we'll first introduce the representation of domain names within DNS messages. Domain names are formatted as a series of *labels*. Labels consist of a one byte length field followed by that number of bytes/ASCII characters representing the label itself. This sequence of labels is terminated by a length field of zero indicating the root "`.`" domain. For example, the series of labels for `www.ipamworldwide.com`. would look like the following in ASCII format, where length bytes are highlighted in darker shading in Figure 9.8.

Starting at the upper left, the value "3" of the first length byte indicates that the following three bytes comprise first label, "`www`". The fifth or next byte after this is our next length byte, which has a value of "13" (0xD), which is the length of "`ipamworldwide`." After this label, the following byte of value "3" is the length of "`com`." Finally, the zero-value byte indicates the root "`.`" domain, fully qualifying the domain name. Note that the darker shaded bytes in the figure are encoded as length bytes to differentiate them from host or domain name characters containing numbers. The first

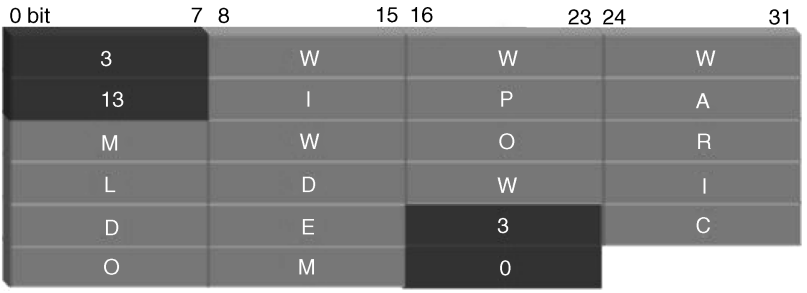

Figure 9.8. DNS labels.

byte in a name will almost always* be a length byte followed by that number of bytes representing the first label, immediately followed by another length byte to eliminate ambiguity.

9.5.2 Name Compression

A given DNS message may contain multiple domain names, and many of these may have repetitive information, the `ipamworldwide.com.` suffix for example. The DNS specification enables message compression in order to reduce repetitive information and thereby reduce the size of the DNS message. This works by using *pointers* to other locations within the DNS message that specify a common domain suffix. This domain suffix is then appended at the point of location referenced by the pointer.

Let's say for example, that our query for www.ipamworldwide.com. returns a pair of DNS servers that can be queried for more information: `ns1.ipamworldwide.com.`, and `ns2.isp.com.` The `ipamworldwide.com.` portion of these domain names is common to the query and one of the answers, while only the `.com` portion is common to the question, first answer and the second answer. Thus, the message is formulated by fully specifying the domain name `www.ipamworldwide.com.` as illustrated above in Figure 9.8. Then when specifying `ns1`, instead of fully specifying `ns1.ipamworldwide.com`, only `ns1` is specified, followed by a pointer to the `ipamworldwide.com.` suffix earlier in the message. When identifying `ns2.isp.com`, the `ns2.isp` labels are specified, followed by a pointer to the `.com` suffix within the message.

How do DNS resolvers and servers differentiate a pointer from a standard label length byte? The DNS standard stipulates that each label may be of length 0–63 bytes. In binary, this is 000000000 to 00111111. Thus, the first two bits, $[00]_2$ in this case, identify the byte as a standard length byte, indicating the length of the following label. A pointer is identified by setting the first two bits to $[11]_2$, and is comprised of two

* As we'll discuss next, the length byte may alternatively consist of a two-byte pointer or a DNS extensions label.

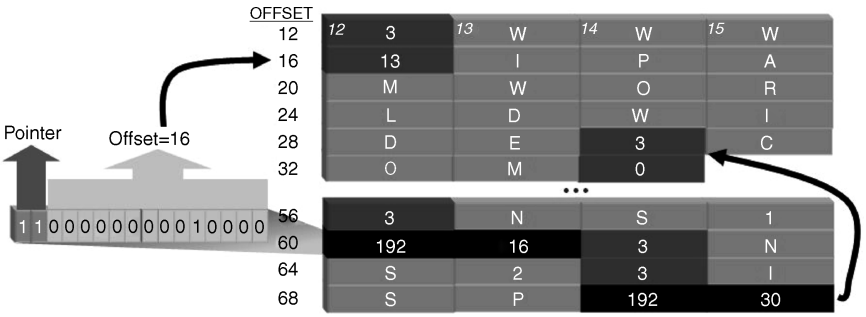

Figure 9.9. Name compression with pointers.

bytes, where the $[11]_2$ bits are followed by 14 bits identifying the offset in bytes from beginning of the DNS header. The first byte of the DNS message header is considered byte 0, and as the message is created, pointers are defined pointing to byte offsets from this point.

Let's look at how this maps out from our prior example. Let's say that beginning 12 bytes from the DNS header, we've included the domain name, `www.ipamworldwide.com`. Now, later in the message, beginning at byte 56 from the beginning of the header, we would like to encode responses `ns1.ipamworldwide.com` and `ns2.isp.com`.

Figure 9.9 indicates how this would look. The first portion is as we discussed earlier, with length bytes (dark shading) followed by the respective number of name bytes (light shading). At byte position 56 in our example, the `ns1` portion of the name is encoded normally, using a label length of "3", followed by `ns1`. However, the next byte is not a standard length byte but a pointer "double-byte" as it begins with $[11]_2$ and is shown as shaded black in the figure. The value encoded in the 14-bit offset field of the pointer is "16", indicating that the portion of the domain name starting at an offset of 16 bytes from the start of the DNS header should be appended to the `ns1` label already specified. The first row of bytes in the figure below enumerates the individual byte offsets (italics), and byte 16 is the length byte of value "13", followed by encoding for `ipamworldwide`, followed by a length byte of value "3", then `com`, then `.` (length byte of value "0"). Concatenating this together, we arrive at the result: `ns1.ipamworldwide.com.`

Returning to the next domain name after processing the pointer, we find encoding for `ns2.isp` followed by a pointer to byte offset 30*, which points to the length byte of "3", followed by `com.`, completing the domain name as `ns2.isp.com.` Considering just these three example domain names, the number of bytes in the message occupied by domain names can be compressed from 59 to 39 bytes.

* Note that pointer double bytes shown above in black are displayed with their byte-wise decimal number representation, which in our example conveniently displays the offset in decimal in the second byte. But just to state the obvious, don't rely on just this second byte when parsing a pointer, as a pointer value can range from 0 to 214 = 16,384, which at this maximum value the decimal representation would be 255-255.

9.5.3 International Domain Names

DNS resolvers and servers communicate hostname queries and responses in ASCII-formatted messages. Configuration information is stored in ASCII* text files. Unfortunately, while ASCII characters have been defined to effectively represent the English language, they do not enable formatting of characters from other languages, especially those using a non-Latin-based alphabet. This limitation certainly impacts the ease-of-use of IP applications in countries where people do not use the English language. RFC 3490 (94) is a standards track RFC that addresses this limitation†.

The RFC is entitled, Internationalizing Domain Names in Applications (IDNA). The "in applications" qualifier in the title insinuates the involvement of applications in this process. Indeed, the onus is placed on the application, such as a web browser or email client, to convert the user's native language entry into an ASCII-based string that can be communicated to a DNS server for resolution. This ingenious approach enables application level support of international character sets for end users without affecting the DNS protocol (or other ASCII-based IP protocols like SMTP). Existing DNS servers can be configured to resolve these ASCII-encoded domain names as they would for native ASCII-based domain names.

International character sets are encoded as Unicode characters. The Unicode standard "provides a unique number for every character, no matter what the platform, no matter what the program, no matter what the language," according to the Unicode Consortium web site (www.unicode.org). Every character is represented as a unique 2–3 byte hexadecimal number. RFC 3490, and its related RFCs 3491 (95), 3454 (96), and 3492 (97), describes the process of converting a Unicode-based domain name to an ASCII-formatted domain name. Note that technically, the domain labels are each converted, not the "domain name."

To resolve international domain names, a DNS server must be configured with resource records encoded in ASCII format, specifically Unicode-mapped ASCII characters referred to as *punycode*. The output of the punycode algorithm results in an ASCII string, which is then prefixed with the ASCII Compatible Encoding (ACE) header, `xn--`. Thus, within the DNS infrastructure, domains denoted as xn--<additional ASCII characters> are likely punycode representations of an international domain name. The application, for example, web browser, is responsible for converting the user-entered URL into Unicode format, then into punycode. The punycode domain name is passed to the resolver on the client for resolution via DNS using ASCII characters. The punycode algorithm is specified in RFC 3492 and several web sites are available for performing conversions for entry into DNS.

Consider an example (98): let's consider a web server host address in the `źdźbło.com` domain. as `www.źdźbło.com`. The domain name contains diacritics and has characters outside of the ASCII character set. The web browser in which this URL is entered would

* RFC 2673 (182), initially a standards track RFC, defined the use of binary data within DNS names but RFC 3363 (183) reverted RFC 2673 to experimental status.

† Note: RFC 3490 which defines "IDNA2003", has been updated by RFCs 5890-4 (184–188), referred to as "IDNA2008", each version denoted by the year specification work began. There are some differences between these versions but material in this section generally applies to both.

convert this to ASCII characters or punycode as `www.xn--dbo-iwa1zb.com`. A corresponding A or AAAA record entry in DNS for the `www.xn--dbo-iwa1zb.com.` host would enable the end user to enter a native language URL while utilizing the existing base of DNS servers deployed throughout the world to identify and connect via the IP address of the destination web server. The net result is that these DNS messages sent on the wire are encoded in ASCII characters.

9.5.4 DNS Message Format

Now let's look more closely at the format of DNS messages used to perform this overall resolution function, incorporating the label formatted domain names we discussed earlier. DNS messages are transmitted over UDP by default, using port 53. TCP can also be used on port 53. The basic format of a DNS message is illustrated in Figure 9.10.

- The message header contains fields that define the type of message and associated information, including the number of records for each of the following fields.
- The Question section specifies the information being sought via this message.
- The Answer section contains zero or more resource records that answer the query specified in the Question section.
- The Authority section contains zero or more resource records referring to name servers authoritative for the given answer or pointing to delegated name servers down the domain tree to which a successive iterative query may be issued.
- The Additional section contains zero or more resource records that contain supplemental information related to the question but are not strictly answers to the question.

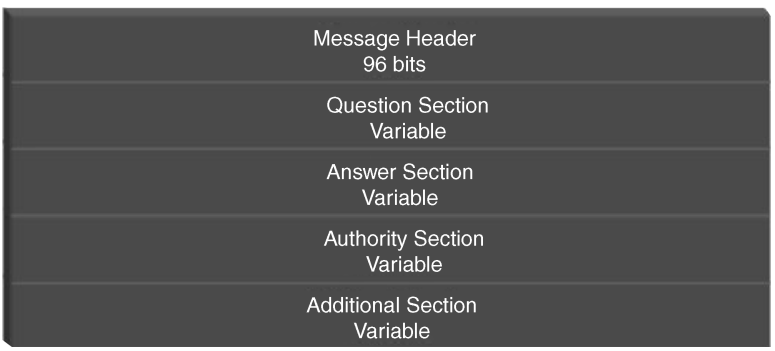

Figure 9.10. DNS message fields (99).

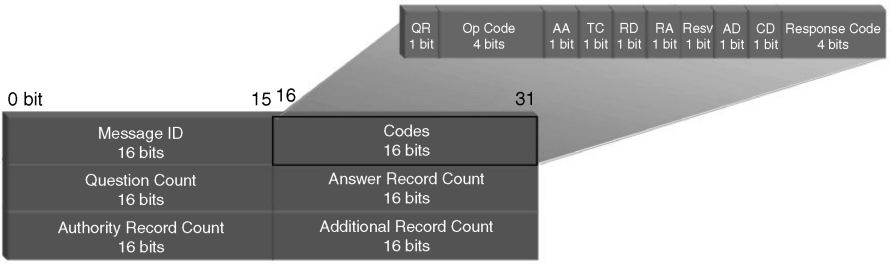

Figure 9.11. DNS message header (99).

Message Header. The DNS message header is included on every DNS message and conveys what type of message is enclosed as well as associated parameters as illustrated in Figure 9.11.

The message header is comprised of six 16-bit fields:

- *Message ID*. Also referred to as transaction ID, an identifier assigned by the resolver and copied in replies from the DNS server to enable resolver correlation of responses with queries.
- *Codes*. Message codes germane to this message. We'll examine these code fields next.
- *Question Count (QDCOUNT)*. The number of questions contained in the Question section of the DNS message.
- *Answer Record Count (ANCOUNT)*. The number of resource records contained in the Answer section of the DNS message.
- *Authority Record Count (NSCOUNT)*. The number of resource records contained in the Authority section of the DNS message.
- *Additional Record Count (ARCOUNT)*. The number of resource records contained in the Additional section of the DNS message.

The following codes bits have been defined:

- *QR (Query/Response)*. This flag indicates that this message is a query (0) or a response (1).
- *Opcode*. The operation code for this message. Presently, the following values have been defined:
  - 0 = Query
  - 1 = Reserved (formerly inverse query, now retired)
  - 2 = Server status request
  - 3 = Reserved
  - 4 = Notify—enables a master zone server to inform a slave zone server with the same zone (and for a slave to acknowledge) that a change has been made to

the zone data. For Notify messages, the Authority and Additional sections are not used and respective record counts in the DNS header should be set to 0.

- 5 = Update—enables a client or DHCP server to update zone data on a DNS server. For Update messages, the interpretation of DNS message fields and corresponding header fields differs from that described above. The message format for Update messages is described in the next section.
- 6–15 = Unassigned.

- *AA (Authoritative Answer)*. When set, this message contains an authoritative answer to the question. This means the response was derived from a DNS server that was configured with the zone's information. If it is not set, the answer was derived from a nonauthoritative DNS server, likely cached information from a prior query. Where multiple answers are provided, this flag pertains to the first record in the Answer section. When set by the client on the query, this indicates that an authoritative answer (not cached) is required.
- *TC (Truncated Response)*. This code indicates that this message was truncated for transmission. This is generally due to the packet length restriction of UDP packets, the default transport layer protocol used by DNS.
- *RD (Recursion Desired)*. This flag indicates that the querier would like the DNS server to iteratively resolve the query, traversing the domain tree as necessary. Most resolvers set this flag to indicate a query as a recursive query, while a DNS server will generally not set this flag when querying other servers.
- *RA (Recursion Available)*. This flag indicates that recursive query support is available from this DNS server.
- *Reserved or Z bit*. Reserved (0).
- *AD (Authentic Data)*. Used within the context of DNS security extensions (DNSSEC), this bit is set by a name server to indicate that information within the Answer and Authority sections is authentic, meaning it has been authenticated.
- *CD (Checking Disabled)*. Used within the context of DNSSEC, this bit enables a DNSSEC resolver to disable signature validation in a DNSSEC name server's processing of this particular query.
- *Response Code (RCODE)*. Provides result status to the client. The currently defined response codes are summarized in Table 9.1. Note that given the 4-bit RCODE field, decimal values 1–15 are encoded within the DNS header RCODE field.

  The DNS extensions (EDNS0, discussed later in this chapter) OPT resource record adds a capacity for 8 additional RCODE bits, bringing the total to 12 bits (up to decimal value 4095) when used in combination with the header RCODE bits.

You'll notice two interpretations of the decimal value 16. BADVERS is the interpretation when encoded within the OPT resource record while BADSIG is the result when encoded within a TKEY or TSIG resource record.

TABLE 9.1. DNS Message Response Codes[a] (106)

RCODE				
Decimal	Hex	Name	Description	Reference
0	0	NoError	No errors	RFC 1035 (99)
1	1	FormErr	Format error—server unable to interpret the query	RFC 1035 (99)
2	2	ServFail	Server failure—server problem has prevented processing of this query	RFC 1035 (99)
3	3	NXDomain	Nonexistent domain—domain name does not exist	RFC 1035 (99)
4	4	NotImp	Not implemented—query type not supported by this server	RFC 1035 (99)
5	5	Refused	Query refused—server refused the requested query, for example, refusal of a zone transfer request	RFC 1035 (99)
6	6	YXDomain	Name exists when it should not as determined during DNS update prerequisite processing	RFC 2136 (100)
7	7	YXRRSet	RRSet exists when it should not as determined during DNS update prerequisite processing	RFC 2136 (100)
8	8	NXRRSet	RRSet that should exist does not as determined during DNS update prerequisite processing	RFC 2136 (100)
9	9	NotAuth	Server is not authoritative for the zone listed in the zone section of the DNS Update message	RFC 2136 (100)
10	A	NotZone	Name used in the prerequisite or update section of a DNS Update message is not contained in zone denoted by the zone section of the message	RFC 2136 (100)
11–15	B–F	Available for assignment		
16	10	BADVERS	Unsupported (bad) OPT RR version	RFC 2671 (101)
16	10	BADSIG	TSIG Signature Failure	RFC 2845 (102)
17	11	BADKEY	Key not recognized	RFC 2845 (102)
18	12	BADTIME	Signature out of the valid server signature time window	RFC 2845 (102)
19	13	BADMODE	Invalid TKEY Mode—requested mode not supported by this server	RFC 2930 (103)
20	14	BADNAME	Nonexistent or duplicate key name	RFC 2930 (103)
21	15	BADALG	Algorithm not supported	RFC 2930 (103)

TABLE 9.1. (*Continued*)

RCODE				
Decimal	Hex	Name	Description	Reference
22	16	BADTRUNC	Bad truncation—Message Authentication Code (MAC) too short	RFC 4635 (104)
23–3840	14-F00	Available for assignment		
3841–4095	F01-FFF	Reserved for private use		RFC 5395 (105)

[a]If you consult the IANA web site (www.iana.org/assignments/dns-parameters), you'll notice values above 4095. Technically these are not RCODEs but reflect the 16-bit error field within the TSIG and TKEY meta resource record types, providing a capacity up to 65,535 for these two resource record types.

Question Section. The Question section within the DNS message format contains, as you might have guessed, the question that is being asked for this query. This section can contain more than one question, as identified by the number referenced in the QDCOUNT header field. Each of these questions has the following format (Figure 9.12).

The QNAME field contains the domain name, formatted as a series of labels. The QTYPE field indicates the query type, or for what purpose is this question being asked. Any resource record type may be included, which we will cover in detail the next chapter. However, there are some QTYPE values that are unique to requesting zone transfers for example that are presently defined including the following (Table 9.2).

The QCLASS field indicates for which class this query is being made, for example, IN for Internet class, the most common class. Classes essentially enable management of parallel namespaces. Currently defined QCLASSes (and DNS CLASSes) in general are defined in Table 9.3.

Answer Section. The Answer section contains zero or more answers in the form of resource records. The number of answers is specified in the ANCOUNT header field. We'll discuss the different types of resource records in the next chapter, and they all share a common generic format as defined in Figure 9.13.

The Name field, also called the Owner name field, is the lookup name corresponding to this resource record (corresponding to the lookup value or QNAME in the original question).

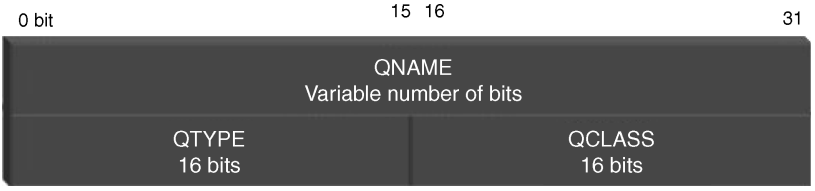

Figure 9.12. Question section format (99).

TABLE 9.2. DNS QTypes[a] (106)

QTypes Only	Query Purpose	QType ID (decimal)	IETF Status	Defining Document
*	All resource records	255	Standard	RFC 1035
MAILA	Mail agent resource records	254	Experimental	RFC 1035
MAILB	Mailbox resource records	253	Obsolete	RFC 1035
AXFR	Absolute zone transfer (entire zone)	252	Standard	RFC 1035
IXFR	Incremental zone transfer (changes only)	251	Proposed Standard	RFC 1995

[a]In addition to RRTypes in Table 12.1 that may be used as QTypes.

The Type field indicates the type of information that is provided for this name. For example, a type of A means that this resource record provides IPv4 address information for the given name. Resource record types are covered in the next chapter and are summarized in Table 10.1.

The Class field represents the namespace class, such as IN for Internet. Valid classes are displayed in Table 9.3.

The TTL or Time-to-Live field provides a time value in seconds with respect to the valid lifetime of the resource record. The receiver of this information may cache this information for `TTL` seconds and may use it reliably. However, upon expiration of the TTL, the cached information should be discarded and a new query issued.

TABLE 9.3. DNS Classes (106)

CLASS				
Decimal	Hexadecimal	Name	Description	Reference
0	0	Reserved	Reserved	RFC 5395
1	1	IN	Internet	RFC 1035
2	2	Unassigned	N/A	IANA
3	3	CH	Chaos	RFC 1035
4	4	HS	Hesiod	RFC 1035
5–253	5-FD	Unassigned	N/A	IANA
254	FE	NONE	None	RFC 2136
255	FF	*(Any)	Any class (valid as QCLASS but not on resource records)	RFC 1035
256–65,270	100-FEFF	Unassigned	N/A	IANA
65,280–65,534	FF00-FFFE	Reserved for private use		RFC 5395
65,535	FFFF	Reserved	Reserved	RFC 5395

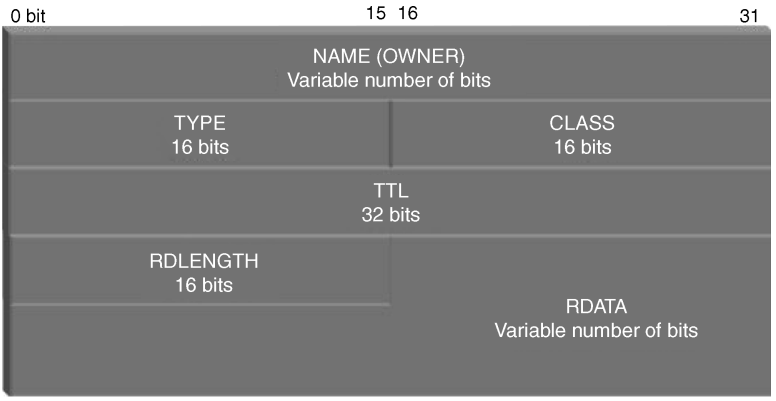

Figure 9.13. Answer section format (99).

The RDLength field indicates the length in bytes of the results (RData) field. The RData field contains the corresponding information of the specified Type in the identified Class, for the given Owner. The RData field has a variable format as we shall see when examining the wide variety of resource record types.

Authority Section. The Authority section contains *NSCOUNT* number of answers in the form of resource records of the same format as discussed in the Answer section. Generally only NS (name server) resource records are valid within the Authority section, though most name servers return an SOA record in this section if the queried name server is authoritative but the answer section is empty. This section also contains information about other name servers that are authoritative for the queried information. This information is used by the querying resolver or more likely, recursive name server, to determine the next name server to query in traversing the domain tree to find the ultimate answer.

Additional Section. The Additional section contains *ASCOUNT* number of answers in the form of resource records, which provide additional or related information to the query, in the same format as discussed in the Answer section.

9.5.5 DNS Update Messages

Update messages enable a client, DHCP server, or other source to perform an update (add, modify, or delete) of one or more resource records within a zone. While Update messages utilize the same basic format as DNS messages just described, the interpretation of some of the fields varies. Update messages, denoted with Op Code = 5 in the DNS message header, are encoded as follows (Figure 9.14).

Contrast this format with that for non-Update DNS messages depicted in Figure 9.10. The message header is of the same format as that of "normal" DNS messages, though the interpretation of the remaining sections differs.

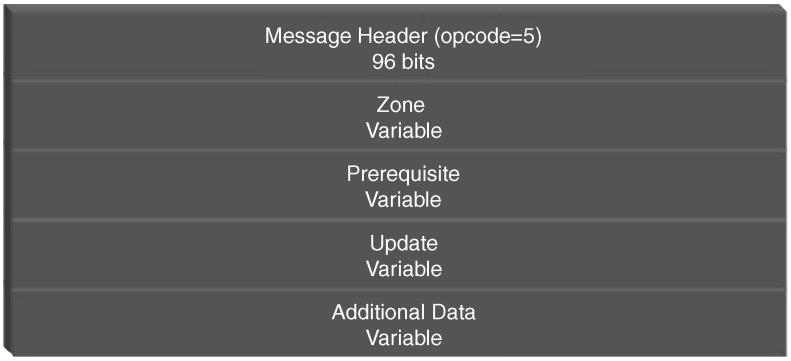

Figure 9.14. DNS Update message format (100).

The Zone section identifies the DNS zone to be updated by this Update message. The Prerequisite section enables the specification of conditions that must be satisfied in order to perform the update successfully. The condition and type of condition are determined by the value of each resource record-encoded parameter within the Prerequisite section. The following table defines how DNS update prerequisites are interpreted based on the values of the Owner, Class, Type and RData fields within the Prerequisites section.

Owner	Class	Type	RData	Prerequisite Interpretation
Match	ANY (255)	ANY	Empty	The matching owner name is in use in this zone
Match	ANY (255)	Match	Empty	An RRSet with matching owner and type exists (value independent, i.e., any RData match)
Match	NONE (254)	ANY	Empty	The matching owner name is not in use in this zone
Match	NONE (254)	Match	Empty	An RRSet with matching owner and type does not exist in this zone
Match	Same as Zone Class	Match	Match	An RRSet with matching owner, type, and RData exists in this zone (value dependent, i.e., RData match)

The Update section contains the resource records to be added to or deleted from the zone using a similar encoding as used in the Prerequisite section as follows.

Owner	Class	Type	RData	Update Interpretation
Owner to add	Same as Zone Class	RR type	RR RData	Add this resource record(s) of the specified owner, type, and RData to the zone's RRSet

(*Continued*)

Owner	Class	Type	RData	Update Interpretation
Owner to delete	ANY (255)	RR type	Empty	Delete the resource records of the specified owner and type
Owner to delete	ANY (255)	ANY	Empty	Delete all resource records of the specified owner name
Owner to delete	NONE (254)	RR type	RR RData	Delete the resource record(s) of the specified owner, type, and RData from the zone

The Additional Data section contains resource records related to this update, for example, out of zone glue records.

Consider an example of an Update message received with the prerequisite and update fields encoded as follows:

Field	Owner	Class	Type	RData
Prerequisite	host.ipamworldwide.com.	IN	DHCID	H8349a+)3jELeA==ES1
Update	host.ipamworldwide.com.	IN	A	10.0.0.200

The Update section contents will only be considered only if the prerequisite condition is met. In this case, the prerequisite condition is that the `host.ipamworldwide.com. IN DHCID H8349a+)3jELeA==ES1 record` exists in the zone, that is, prerequisite type RRSet with matching owner, type, and RData (value dependent). If it does exist, then the `host.ipamworldwide.com. IN A 10.0.0.200` resource record from the Update section will be added to the zone. If not, the update will not be performed.

This particular example illustrates how the ISC DHCP server performs dynamic updates of DNS data upon assigning an IP address, in this case 10.0.0.200 to host.ipamworldwide.com. The DHCID record provides a hash of the host's hardware address receiving the IP address to uniquely identify the host. The prerequisite condition for updating the address record provides a means to assure that only the original holder of this A record can modify it, minimizing naming duplication or hijacking.

9.5.6 DNS Extensions (EDNS0)

Thus far in our discussion of the DNS message header, one may observe that all code bits are assigned but one, and additional response code assignments have been required by necessity. In addition, many hosts can process larger multipart UDP packets than the originally specified size limit of 512 bytes. As a result of these limitations, as well as the desire to add additional domain name label types, DNS extensions were defined in RFC 2671 (101).

RFC 2671 defines version 0 of extension mechanisms for DNS, which is denoted EDNS0. The RFC addresses the above constraints by defining the following extensions:

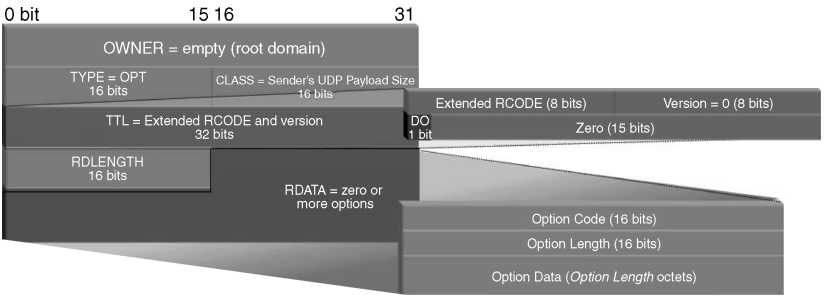

Figure 9.15. EDNS0 format (101).

- A new domain label type is defined to denote DNS extensions. As we discussed, the first two bits of the domain label uniquely identify the label as a length byte (first two bits = $[00]_2$) or as a pointer (first two bits = $[11]_2$). The extended label type has been assigned $[01]_2$ as its first two bits.
- EDNS0 defines a pseudoresource record, the OPT record (i.e., RRType = OPT). The OPT record is placed in the Additional section by the resolver or server to advertise its respective capabilities. The OPT resource record is used to advertise capabilities of the sender (client or server) to the recipient, and only one OPT record should be present.

The OPT pseudoresource record is encoded as follows, enabling specification of the sender's UDP packet size and additional response code bits (Figure 9.15).

The OPT record should never appear in a zone file. Thus, while the OPT pseudoresource record utilizes the same wire format as other resource records, the definition of standard fields has been modified to provide only extension information. The NAME field is zero for the OPT record. The TYPE is OPT, and the CLASS field indicates the maximum size of the sender's UDP payload. The 32-bit TTL field is divided into three fields:

- *Extended Response Code*. Adds 8 bits to the 4-bit RCODE in the DNS message header to provide 12 bits total.
- EDNS version number.
- *Extended Header Flags*. Bit 0 is currrently defined as "DNSSEC Answer OK" meaning that the querying server is capable of processing DNSSEC resource records. The remaining 15 bits of the extended header are currently reserved.

The RDLength field indicates the length of the RData field, which consists of a set of zero or more options, each encoded with an option code, option length, and option value.

One option has been officially defined thus far via RFC 5001 (191): the name server identifier (NSID) option. This option, defined with option code = 3, enables a resolver to request and a server to provide its identity as defined by the server administrator as its name, IP address, pseudorandom number, or other character string (configurable in

BIND using the `server-id` statement). This EDNS0 option is useful for debugging in environments where many servers share a common IP address, such as in deployments of anycast addressing or with load balancers. Two additional options, Long-Lived Query* (LLQ; option code = 1) and Update Lease Life† (UL; option code = 2) are currently on hold as RFCs and have not been officially published regarding these settings.

9.5.7 Resource Records

This chapter has covered the organization of DNS data and traversal of the domain tree, as well as the format of messages to do so. Once we've navigated the tree and located a DNS server authoritative for the information for a domain, how do we actually get the lookup information we're seeking for a particular purpose or application?

Resource records associated with the given domain provide the means to map the question to an answer. The type of resource record defines the desired result type, for example, the A resource record type will provide an IPv4 address as an answer while the AAAA type will provide an IPv6 address. The answer may be "the final answer" or information that can be used to obtain the desired answer via additional queries or other means.

* A Long-Lived Query is a mechanism for a resolver to request receipt of notification of zone information changes; something like a DNSNOTIFY for clients.

† The Update Lease Life mechanism would enable a DHCP server to inform the DNS server within a DNS Update message of the corresponding client's lease length in seconds for new and renewed leases.