11

DNS SERVER DEPLOYMENT STRATEGIES

This chapter* examines deployment strategies and trade-offs for DNS servers. DNS servers generally enable a more role-oriented deployment than DHCP and our discussion will relate configurations to particular roles (external resolution, caching, internal, etc.). Of course the usual trade-offs between budget dollars and proliferation of servers closer to end users applies for DNS as it does for DHCP.

As with DHCP deployments, DNS deployment designs should account for high availability, performance, and security. Using a component building block approach to DNS server deployment in order to segment name space and resolution responsibility can help achieve these general objectives, and we'll discuss such an approach throughout this chapter. Keep in mind that there is no cookie cutter DNS deployment in defining a "one size fits all" architecture. However, by defining role-based server configurations as deployment building blocks, you can select which are applicable to your environment's scale and policies.

* Much of the content of this chapter is based on conversations with and private documentation by Alex Drescher (165).

IP Address Management: Principles and Practice, by Timothy Rooney
Copyright © 2011 the Institute of Electrical and Electronics Engineers, Inc.

11.1 GENERAL DEPLOYMENT GUIDELINES

Some general principles to keep in mind include the following:

- Deploy a master and at least two slaves as authoritative for any given zone or set of zones.
- For Microsoft Windows DNS server deployments, deploy multiple domain controllers for each domain.
- Deploy authoritative servers on different subnets and ideally, different locations for site-diverse high availability.
- Deploy authoritative servers "close" to clients/resolvers for better performance and less network overhead. For external servers, deploy close to Internet connections; for internal servers, deploy nearer to higher density employee areas.
- For master servers, consider deploying a redundant hardware solution.
- Separate DNS servers should be deployed to handle external queries versus internal queries. By external queries, we mean those queries originating from outside the organization, for example, the Internet. Internal queries are those originating from within the organization.
- Consider separating servers responsible for resolving authoritative data from those responsible for resolving recursive queries on behalf of stub resolvers.

11.2 GENERAL DEPLOYMENT BUILDING BLOCKS

This section provides an overview of common DNS deployment scenarios in terms of component building blocks. We will break down these building blocks into four major categories based on from where a query originates (query source) and on the scope of information being queried (query scope). We define the query source as above, with external queries originating from the public Internet, while internal queries originate from within your organization. The query scope also follows this general breakdown, with external scope dealing with Internet-reachable resolution data and internal encompassing intr-aorganization resolution information. The following table summarizes this categorization using very original category names.

	Query Scope	
Query Source	External	Internal
External	External–External	External–Internal
Internal	Internal–External	Internal–Internal

In addition, we'll discuss a few nonrole-specific scenarios that apply across any building block scnenario. These may apply to one or more categories as they provide special resolution or availability features.

A summary of categorized building block scenarios follows:

- *External–External Category*. This category consists of a DNS deployment to resolve queries originating from the Internet for your public (external) resolution information. If you have an Internet connection for a web site, email or for other publically available Internet applications, this category must be addressed in your deployment strategy.
  - *External DNS Server Deployment*. This building block scenario seeks to provide robust name resolution functionality for external clients seeking legitimate name resolutions for the organization's public resources, such as web servers, email servers, and the like, while minimizing exposure to those seeking to attack the DNS infrastructure or infiltrate it for attack purposes. Deployment of external DNS servers features a hidden master server with a number of slave servers. As we'll see, these servers should never be queried by a resolver directly; only by recursive name servers resolving on behalf of resolvers.
- *External–Internal Category*. This category includes queries from outside the organization seeking resolution for internal hosts and resources. Besides providing partners access with a subset of "internal" resolution information, this category should generally be prohibited. DNS server deployment for this category (for partner access) should mimic the external DNS scenario under the External–External Category, though perhaps deployed as a parallel per-partner implementation.
- *Internal–External Category*. This category consists of handling internal queries requesting Internet resource resolution.
  - *Internet Caching DNS Servers*. Internet caching servers are internal DNS servers that cache Internet resolutions for use by internal DNS servers on behalf of internal resolvers. Caching servers can be deployed as a function of or independently of internal resolution DNS servers. In the former case, authoritative servers for internal name space simply escalate queries to Internet root servers down the domain tree to resolve queries, building up a cache of resolved data. The latter case, using independent caching servers, enables other internal-resolving DNS servers to be configured to funnel queries for external data through these caching servers. Doing so enables more control over which servers make external queries while enabling them to build up a substantial cache over time.
- *Internal–Internal Category*. This category deals with internally originating queries for internal resolution information.
  - *Internal Resolution DNS Servers*. DNS servers are required to resolve queries for internal destinations from internal hosts. These DNS servers are configured with authoritative information for the internal name space. Any Internet or irresolvable host queries can be funneled to caching (Internal–External)

servers. As with external master DNS servers, internal master DNS servers should be "hidden" for added security and information integrity.

- *Departmental DNS Servers.* For larger organizations, some business units or entities may desire to run their own name subspace within the organization's name space. This scenario features delegation of name space internally but is otherwise a replica of the internal resolution DNS server's case, though for a subset of the internal name space.
- *Internal Root Servers.* Internal root servers can be configured as the authoritative root of the internal name space for resolution of internal queries.

- *General Cross-Role Deployment Configurations.* This category may apply across multiple deployment scenarios.
  - *Stealth Slave DNS Servers.* While hidden master deployments are discussed in the external and internal deployment scenarios, another "hiding" approach is to hide slave servers. Masters are generally hidden from direct resolution queries from resolvers as well as other name servers; stealth slaves are available for name resolution from resolvers but are hidden from other name servers requesting iterative queries.
  - *Split View DNS Servers.* Deployment of multiple "versions" of a domain can be useful for limiting access to privileged name resolution information. The views feature of BIND 9 enables deployment of multiple views or versions of a domain. The views feature is not currently supported by Microsoft DNS.
  - *Anycast Servers.* Anycast addresses enable the assignment of a single IP address to multiple DNS servers. This enables the use of a common IP address to which to send queries for increased performance and availability. The routing protocol in use in the network performs the routing to the nearest server with the anycast IP address.

Based on the size of your organization (and budget), you may choose to deploy an external–external set of DNS servers and an internal–internal set. This is the minimum deployment configuration, though many organizations also deploy dedicated internal-external servers. Larger or more sophisticated deployments may feature elements of other categories as well. The advantage of this building block approach is simplicity and modularity, enabling selective deployment scenarios based on your environment.

11.3 EXTERNAL–EXTERNAL CATEGORY

This category relates to deployments for responding to queries from external sources for the organization's public resolution information.

11.3.1 External DNS Servers

Once again, "external" refers to servicing DNS queries from outside or external to the organization, that is, from the Internet. Resolution services must be provided for general

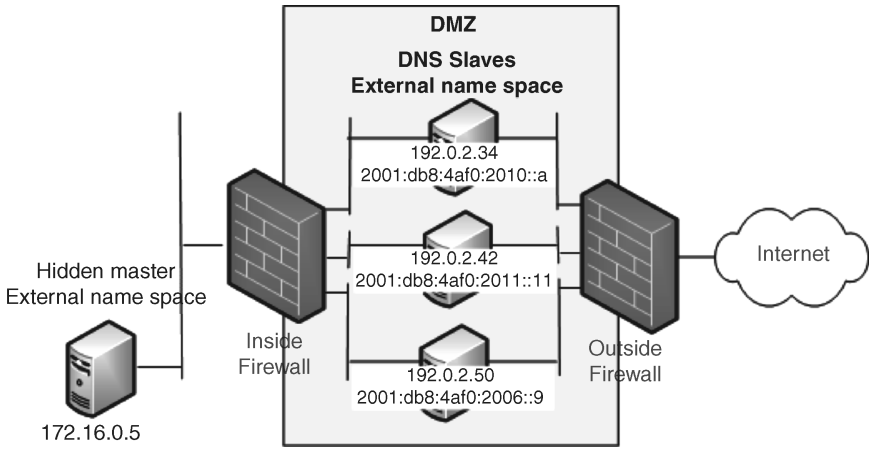

Figure 11.1. External DNS building block example.

access to the organization's web site, email, and other applications, but care must be taken to secure the information integrity of these external servers, given their inherent exposure and potential vulnerability in serving external clients. The recommended approach is to deploy two or more slave DNS servers to service external requests, and to configure these servers with IPv4 and IPv6 addresses. These slave servers may be directly on an external subnet exposed to the Internet, or behind a "first line" firewall within a DMZ, as shown in Figure 11.1.

We've configured each DNS server as dual stack (IPv4 and IPv6) to enable reachability via either protocol. We've placed each external DNS server on its own subnet since we carved up our 192.0.2.0/24 network for external hosts such as these DNS servers. In this case, we're illustrating three /29 subnets on which these servers are deployed: 192.0.2.32/29, 192.0.2.40/29, and 192.0.2.48/29. We've allocated three IP addresses from three /64 subnets derived from our 2001:DB8:4AF0:2000::/56 "external" IPv6 allocation.

Figure 11.1 illustrates a hidden master DNS server deployed behind a DMZ internal firewall and should not be directly query-able from external clients. Since this master server maintains the "master configuration" from which the slave servers transfer, its information integrity must be safeguarded. For this reason, this master DNS server should be configured as hidden, meaning that it cannot be identified by querying other DNS servers. Hiding the master DNS server reduces the risk of an attacker identifying the master server, then attempting to infiltrate its configuration. Imagine the potential impact and embarrassment if an attacker changed your www record to an illicit web site! The mechanics of hiding a master name server entail excluding NS and glue records for the server in this and the parent domain zone file and modifying the master server name ("mname") field of the SOA record in each zone db file. External facing zones are generally static zones, with no dynamic updates, so modifying the mname field can be performed safely in such cases.

Make sure the NS and glue records configured in your parent's domain point to the external slave DNS servers, not the master. This should be arranged through your ISP or domain registrar. For the example in the figure above, you could supply the following NS/glue record information to your parent (e.g., ISP) domain administrator:

```
ipamworldwide.com. 86400 IN NS  extdns1.ipamworldwide.com.
ipamworldwide.com. 86400 IN NS  extdns2.ipamworldwide.com.
ipamworldwide.com. 86400 IN NS  extdns3.ipamworldwide.com.
extdns1.ipamworldwide.com. 86400  IN  A  192.0.2.34
                           86400  IN  AAAA 2001:db8:4af0:2010::a
extdns2.ipamworldwide.com. 86400  IN  A  192.0.2.42
                           86400  IN  AAAA 2001:db8:4af0:2011::11
extdns3.ipamworldwide.com. 86400  IN  A  192.0.2.50
                           86400  IN  AAAA 2001:db8:4af0:2006::9
```

Note that external DNS servers should be deployed on different subnets and on different ISP connections if available, or have your ISP also run a slave server on your behalf. We can restrict DNS queries on our ISP connection firewalls as shown in the examples in Tables 11.1–11.2. We've consolidated the rule for our three /29 subnets into a single /27 for simplicity. In terms of firewall configuration, these are simply guidelines; your policies may be more stringent. Similar allow-* option settings (e.g., allow-query) in BIND or Windows DNS can be defined as access control lists (ACLs) for each server as well as we'll illustrate later in the chapter.

As just mentioned, these servers should be deployed in multiple locations if possible to maximize availability. If you have dual Internet connections, it's recommended that external slaves be deployed in a similar configuration at or near each connection point. Figure 11.2 illustrates a multihomed external DNS configuration. In this configuration, IPAM Worldwide's external DNS servers are accessible via either ISP, over multiple physical links and on differing subnets within the DMZ.

TABLE 11.1. Example Outside Firewall Rules for DNS Messages

Message and Direction	Control	Source Address	Source Port	Destination Address	Destination Port
DNS queries from the Internet	Allow	Any	>1023	192.0.2.32/27, 2001 :db8:4af0:2000/56	53
Responses to DNS queries	Allow	192.0.2.32/27, 2001 :db8:4af0:2000/56	53	Any	>1023
All others	Deny	Any	Any	Any	Any

TABLE 11.2. Example Inside Firewall Rules for DNS Messages

Message and Direction	Control	Source Address	Source Port	Destination Address	Destination Port
Queries from slaves to master (e.g., refresh queries)	Allow	192.0.2.32/27	>1023	172.16.0.5	53
Responses from master to slaves	Allow	172.16.0.5	53, 1053	192.0.2.32/27	>1023
All others	Deny	Any	Any	Any	Any

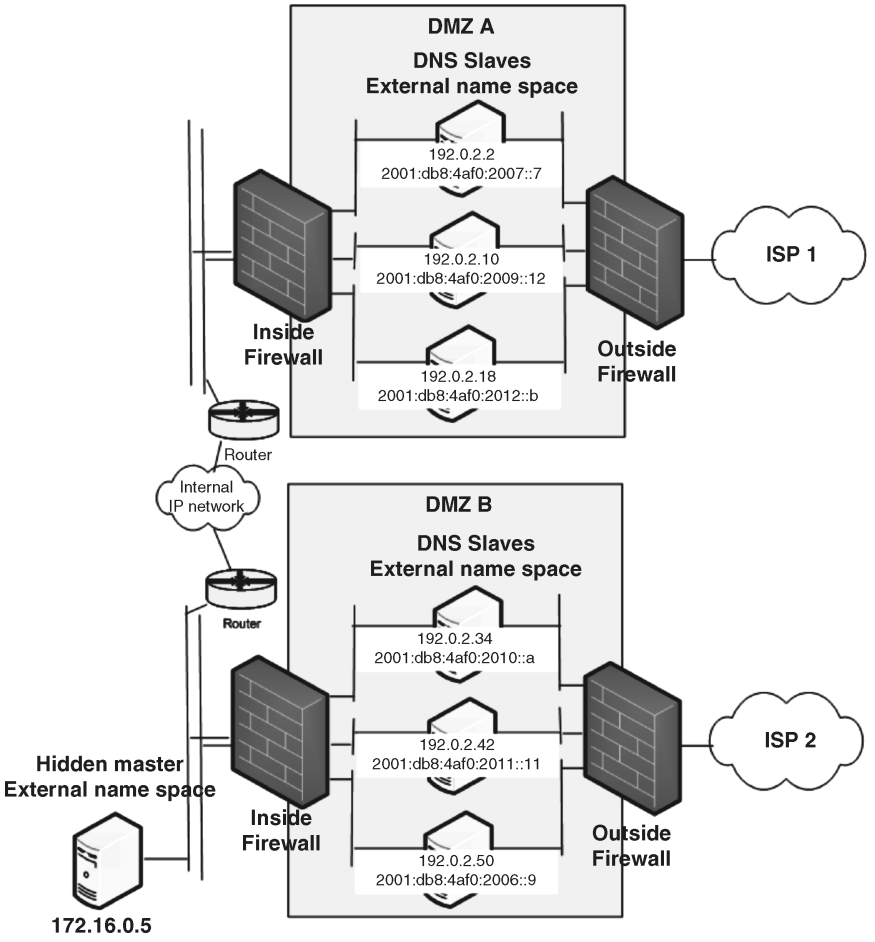

Figure 11.2. External DNS in a multihomed scenario.

Heeding these additional recommendations will also help maintain security of external DNS servers and the information they're resolving.

- Run DNS in a chroot'd jail. This reduces exposure to the server platform and related services that could provide a stepping stone to other attack targets by limiting file system access to a limited set of functionality, not the full root access on the server. This "change root" or chroot configuration runs named in a specified subdirectory, not the root directory. Should an attacker gain access to named, they will have gained access only to the chroot'd directory, not the root directory where access to all file system resources is available. Most appliance products are preconfigured to run DNS in a directory system "jail."
- Keep up with the latest versions of BIND or Microsoft DNS software and subscribe to the bind-announce email list at isc.org or Windows update. Email notifications are sent out upon detection of security vulnerabilities with associated remediation steps and patches. In addition, monitor operating system vulnerabilities that are reported for the platform(s) on which you are running DNS.
- Recursive queries must be disabled. This will enable these servers to process only iterative queries, that is, from other DNS servers, not resolvers. These servers must not perform DNS queries on behalf of anyone. This reduces the processing load of supporting recursive queries and more importantly, reduces the exposure to cache poisoning or denial of service attacks against these servers.
- Configure access control lists on zone transfers as we'll illustrate below.
- Disable dynamic updates and notify's for these external zone(s) if possible. If external name space data changes frequently and requires dynamic updates, restrict update and notify messages between the master and its slaves.
- Configure transaction signature (TSIG) keys to sign zone transfers and updates between the master and slaves as we'll illustrate next. This provides data originator authentication.
- Configure the port number on the master server on which to send notify messages to the slaves if notify is required. This requires specification of the corresponding port on the internal firewall.
- Configure the port number on the slave servers on which to obtain zone transfers from the master. This enables specification of the corresponding port on the internal firewall.
- Secure the rndc control channel by configuring listen-on address/port, allow from, and keys statements. You may even want to disable rndc on these servers.
- Set the version option to a bogus setting. There's no need to communicate what version of BIND you're running and it may provide attackers information on how best to attack the server, especially if you aren't keeping up with newer releases.

11.4 EXTERNAL–INTERNAL CATEGORY

This category comprises external hosts querying information regarding internal (non-public) resolution information. In general divulging information about internal hosts is undesirable and a potential security risk. Even interconnected partners should only have access to guarded information, certainly not the entire internal name space.

11.4.1 Extranet DNS Server Deployment

Inter-partner connections are typically configured as virtual private network (VPN) connections over the Internet or a private network and typically involve a "partner DMZ" or firewall between the partner space and the internal network. The DNS deployment architecture for this category, shown in Figure 11.3, mirrors that of the External–External category though the resolution data configuration is somewhat different. Depending on what resolution data may be divulged to a given partner, the DNS server queried by partner clients must be configured accordingly with such data. Thus, the concept of hidden master and visible slaves supporting no recursion per the External DNS scenario applies.

The partner-specific resolution information may be defined as an "extranet" name space, contained within respective zone files configured on these DNS servers. Additionally implementing views on the DNS servers serving the partner link enables per-partner resolution information if multiple partners access a common DNS server. We'll talk about DNS view configurations a little later but they allow the DNS server to answer queries depending on "who's asking," for example, for Partner A the ftp hostname may resolve differently than the ftp hostname for Partner B.

Each partner's DNS resolution process should be configured to reach these DNS servers to resolve information you wish to divulge about your network. We'll cover this from the complementary perspective of configuring our servers to resolve information for our partner's resolution data in the Internal–External category section.

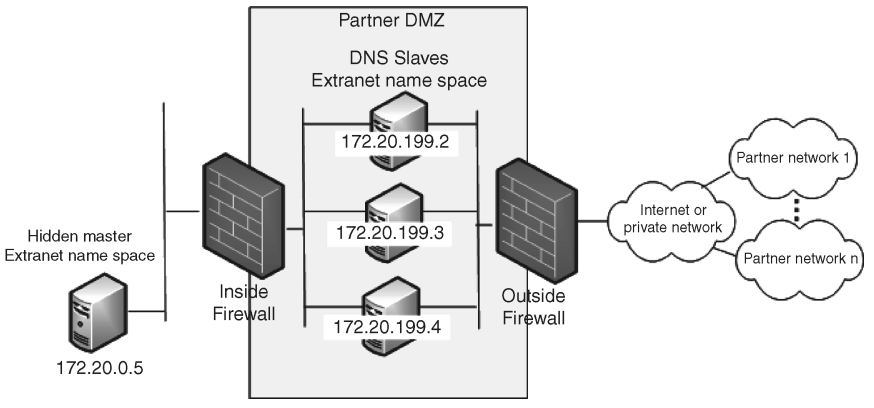

Figure 11.3. Extranet DNS deployment.

11.5 INTERNAL–INTERNAL CATEGORY

11.5.1 Internal Resolution DNS Servers

We'll review a variety of configurations for deploying DNS servers for internal resolution, including a variety of master/slave configurations, caching servers, extranet access servers, and internal root servers.

Internal DNS Servers. Internal DNS servers should be deployed to resolve queries from internal clients for internal host information. We can refer to them as internal TLDs, "top-level domains" since they will be the masters of the top level of the internal name space, for example, ipamworldwide.com., and may delegate subdomains to other internal DNS servers as we'll describe later. The internal master DNS server may be deployed as a hidden master. It's generally a good idea to control the information integrity of master servers by hiding them, as internally initiated attacks account for a large percentage of network security breaches.

Deploying a sufficient number of slave servers, which of course are also authoritative for their respective zone information, enables resolution of client queries, while offloading the master servers to handle only configuration updates. If a master DNS server fails, slaves will continue to resolve queries, but a lengthy outage can compromise the validity and certainly timeliness of the slaves' zone data. The slaves will continue supporting this zone data until the `expire time` is exceeded, after which the server will no longer consider itself authoritative for the zone. Dynamic updates will also not be possible if the master server is down.

An important consideration for Microsoft client environments with client-driven dynamic updates when attempting to hide a master DNS server is that Microsoft clients rely on the MNAME field to identify the master server to update. In this case, using BIND DNS servers, you can still hide the master by changing the MNAME field to point to a legitimate slave server, and configuring the `allow-update-forwarding` option to forward updates to the primary master. In general, we recommend against having clients directly update DNS in favor of having your DHCP servers perform this function. The fewer the entities that can update DNS, the tighter access security can be configured and the fewer the variety of update sources will be able to impact DNS data integrity.

The use of DHCP servers is generally necessary anyway to provide dynamic addressing to laptops, desktops, printers, VoIP phones, and other IP devices. Given that most if not all of these device types will require entries in DNS corresponding to their respective assigned addresses, we need to allow updating of DNS from our DHCP servers. Since we have a hidden master, we could configure the DHCP servers to update a slave DNS server. This server can be deployed with hardware redundancy to minimize any outage time where DNS cannot be updated for DHCP clients.

Figure 11.4 shows an example four-server deployment for our internal ipamworldwide.com name space. As described in the architecture overview, internal client resolvers should be configured with at least two DNS servers. Any number of additional slaves can be deployed in branch offices or remote sites to balance the query load.

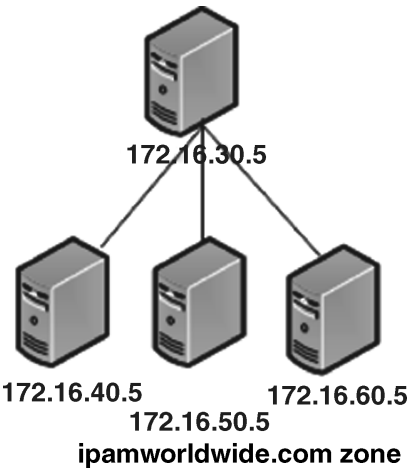

Figure 11.4. Internal DNS servers for internal clients.

11.5.2 Internal Delegation DNS Master/Slave Servers

In larger organizations, subdomains can be delegated to particular departments or divisions. Continuing our example, we've created a finance.ipamworldwide.com domain as nondelegated. This means that the configuration and resource records associated with the finance.ipamworldwide.com domain is included in its parent zone file for ipamworldwide.com.

For other departments, separate DNS administrators may desire to manage their own domain information. Let's consider an example. If the Engineering department desires to run DNS for the eng.ipamworldwide.com domain, the team managing the internal top-level domain, ipamworldwide.com, the parent of eng.ipamworldwide.com may allocate a new delegated domain, that is, zone. The NS and glue records for the DNS servers authoritative for this new zone need to be configured on both the authoritative servers themselves and on those authoritative for the parent zone, ipamworldwide.com. Technically, the eng.ipamworldwide.com zone is authoritative for these NS (and glue) records, not the parent, but the parent must configure them to provide referrals down the domain tree.

Likewise, the associated reverse domain(s) could be maintained within the Engineering organization. Let's assume they've been allocated the 172.20.0.0/15 address space from the organization's private space. Note this space does not fall on octet boundaries. Thus, two reverse domains will be required. Since the 172.20.0.0/15 space is comprised of the 172.20.0.0/16 and the 172.21.0.0/16 space together, we'll create these two octet-boundary-based reverse domains: 20.172.in-addr.arpa and 21.172.in-addr.arpa.

Figure 11.5 depicts the example server deployment. As you can see, it looks exactly like that of the internal TLD DNS servers with a master and several slave servers. This is the common deployment configuration of authoritative BIND-based DNS servers.

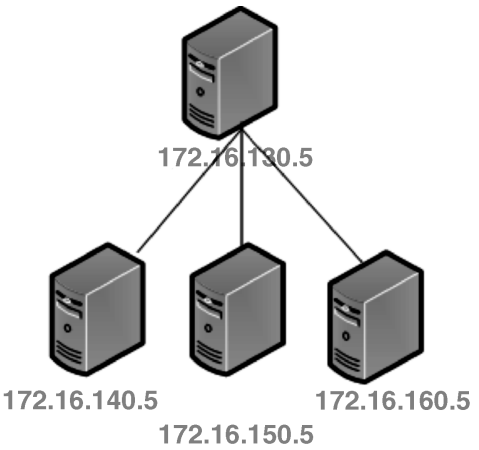

Figure 11.5. Internal name space delegation.

11.5.3 Internal Root Servers

Each of the internal DNS server configurations described so far declare the root (.) zone as type hint, pointing to the root-hints.txt file. So far, we've assumed this file contains the Internet root server NS and A/AAAA records to enable scaling of the domain tree for external information resolution. In environments where Internet access is limited, prohibited, or otherwise generally unavailable, a set of internal root servers can be deployed to authoritatively resolve or more likely, refer queries otherwise not resolved by local recursive servers.

These internal root servers effectively replace the Internet root servers in the organization's other servers' hints files. In other words, these internal root servers are the ultimate authority and last chance for name resolution for internal clients. Thus, this configuration eliminates reliance on Internet name servers, but limits resolution to that information contained within the root servers and their delegated domain servers.

In this configuration, we can use the same configuration files defined above for our internal DNS servers. But we need to define the hints file (root-hints.txt in our case) with a listing of the internal root servers instead of the standard Internet root servers. The hints file contains only NS and glue records for the root servers. If we use a simple example of three root servers, our referenced hints file might look like

```
.                              IN  NS    root1.ipamworldwide.com.
root1.ipamworldwide.com.       IN  A     172.16.1.1
                               IN  AAAA  2001:db8:4af0:f1::1
```

```
.                              IN  NS    root2.ipamworldwide.com.
root2.ipamworldwide.com.       IN  A     172.18.1.34
                               IN  AAAA  2001:db8:4af0:a::1

.                              IN  NS    root3.ipamworldwide.com.
root3.ipamworldwide.com        IN  A     10.251.0.5
                               IN  AAAA  2001:db8:4af0:c001::1
```

The configuration file on each of the root servers might look like the following:

```
acl internal-nets { 10.0.0.0/8; 172.16.0.0/12; 2001:db8:4af0::/48; };
options {
 recursion no;                    // iterative queries only
 allow-query { internal-nets; }; // allow from internal nets
 allow-notify { none; };          // disallow notify processing
 allow-transfer { none; };        // disable zone transfers
 allow-update { none; };          // disable updates
};

zone "." {
 type delegation-only;
 file "db.dot";
};
```

Each root server is a *delegation-only* type server, as denoted within the root zone declaration block at the bottom of the example configuration file above. The delegation-only type is a special form of type master that only responds with referrals, not answers. Such a multimaster configuration in BIND is possible for static zones like this that change infrequently. Any modifications to the root zone implies a new or modified top-level domain assignment and must be made by updating the db.dot file on each root server. There are no dynamic updates, notify's, or zone transfers. All changes must be made by administrator modification of the db.dot file and requires a coordinated loading of the modified zone file on all masters to place it into service synchronously.

The following illustrates a portion of the example db.dot file, which contains resolution data for the internal root zone.

```
$TTL 1d
. IN SOA dns1.ipamworldwide.com. dnsadmin.ipamworldwide.com (
                  1        // serial number
```

```
                2h      // refresh interval of 2 hours
                30m     // retry after 30 minutes
                1w      // expire after 1 week
                1d );   // negative caching TTL of 1 day

ipamworldwide.com.      IN NS  dns1.ipamworldwide.com.
                        IN NS  dns2.ipamworldwide.com.
                        IN NS  dns3.ipamworldwide.com.
                        IN NS  dns4.ipamworldwide.com.
partner.net             IN NS  dns-par1.ipamworldwide.com.
                        IN NS dns-par2.ipamworldwide.com.
. . .
16.172.in-addr.arpa     IN NS    dns1.ipamworldwide.com.
                        IN NS    dns2.ipamworldwide.com.
. . .
0.f.a.4.8.b.d.0.1.0.0.2.ip6.arpa IN NS dns1.ipamworldwide.com.
                             IN NS dns2.ipamworldwide.com.
. . .
dns1.ipamworldwide.com.       IN   A    172.16.40.5
dns2.ipamworldwide.com.       IN   A    172.16.50.5
. . .
dns-par1.ipamworldwide.com.   IN   A    172.20.199.2
dns-par2.ipamworldwide.com.   IN   A    172.20.199.3
```

Referrals to other DNS servers we've configured previously enable authoritative resolution of queries. Here, any queries falling within ipamworldwide.com, including eng.ipamworldwide.com, will be sent to our internal authoritative servers. Note that we do not include the 172.16.30.5 server in this list as this is a hidden server.

Any resolutions requiring external, for example, partner extranet DNS servers would require corresponding entry in the hints file as well to refer to internal partner-facing servers. In our example above, access to the partner.net domain and subdomains would be referred to authoritative DNS servers dns-par1.ipamworldwide.com or dns-par2.ipamworldwide.com. As we'll discuss under the next category, these servers may be configured as *stub* servers for the partner.net zone; alternatively, direct referral to the partner.net DNS servers may be used on these entries within the root zone file. The bottom line is that these root servers can delegate top-level domains to other DNS servers, which must in turn be configured to resolve authoritatively for the corresponding domains and subdomains.

11.5.4 Stealth Slave DNS Servers

Stealth slave DNS servers as so called due to the lack of NS and glue records for the server in the parent zone as we just observed for the 172.16.30.5 server in the previous section; hence, this hidden name server is not identified via NS queries. We've used this configuration to hide a master server, but this can equally apply to slave servers. Thus, when traversing the domain tree, other DNS servers will not query this hidden server for resolution, as it will not be "advertised" in the parent zone's referrals.

This type of configuration may be deployed for a slave server in order to reduce inter-server traffic or to control such traffic to a fixed combination of resolvers and other servers. Local resolvers can be configured to query a stealth slave server. Other than removing the stealth slave server's NS and glue records, the configuration is equivalent to that of a normal slave server.

11.5.5 Multi-tiered Server Configurations

Deploying hidden masters can help reduce attack exposure to the server that is master for a set of zones. This enables resolvers and other servers to query slave servers, which are also authoritative for configured zones. But in some cases, it is desirable to add a third layer to supplement the two-tier master-slave model. This upper level tier would feature a master DNS server, perhaps master for all internal name space that can in effect provide the true master database of DNS information for an organization. This scenario is illustrated in Figure 11.6.

Let's call this top-level DNS server a Tier 1 server. It is configured with all zones as type master. Our former master servers, 172.16.30.5 and 172.16.130.5, which we'll refer to as Tier 2 servers, are now configured as slaves, pulling zone transfers from our Tier 1 master. The set of original slave servers, at Tier 3, remain as such and continue to pull zone transfers from their respective Tier 2 servers. These Tier 2 servers, though slaves, are configured within the `masters` statement of each Tier 3 server's zone statements. Thus, no changes are required in the configuration of our Tier 3 servers. However, our Tier 2 servers must be modified as slaves for each configured zone with the Tier 1 server identified as each zone's master. The Tier 1 server is referred to as the *primary master* in this configuration, as this is the server on which zone updates may be made directly, with zone transfers to Tier 2 and Tier 3 successively to update all authoritative servers accordingly.

11.6 INTERNAL–EXTERNAL CATEGORY

This category of deployment scenarios addresses DNS resolutions for information external to an organization by resolvers within the organization.

11.6.1 Hybrid Authoritative/Caching DNS Servers

Most authoritative recursive DNS servers cache resolution information they receive during the query resolution process, on behalf of resolvers, so most authoritative

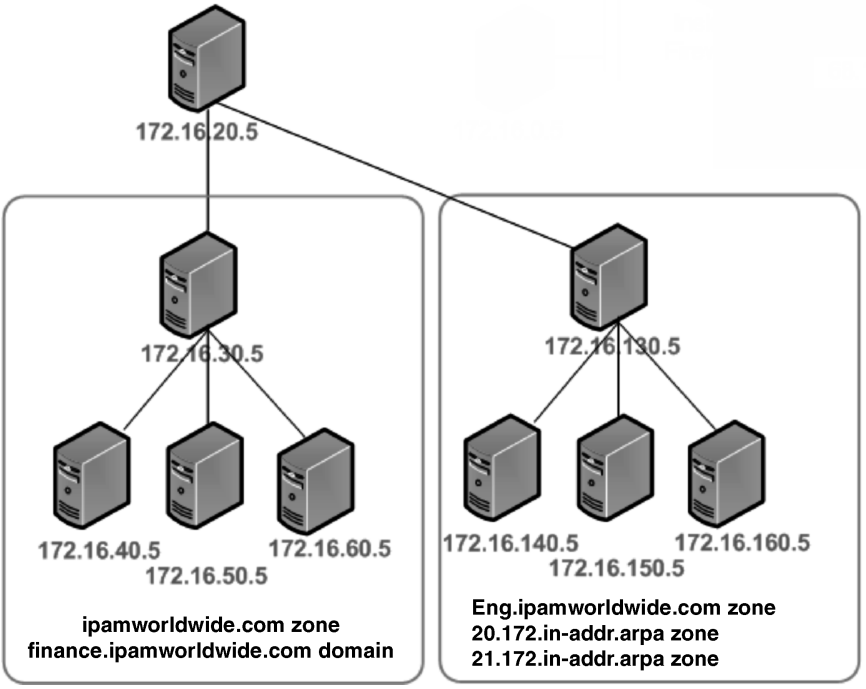

Figure 11.6. Three-tiered internal server structure.

servers are technically "hybrid" servers. The internal server configurations we've discussed so far fall into this scenario: they attempt to resolve from authoritative information and failing that, escalate to Internet (or internal) root servers. For small to modest sized organizations with a handful of internal DNS servers, this configuration works well.

A concern arises however if several (defined by personal tolerance but about 10 or more) such servers perform these Internet queries. DNS resolutions require IP traffic outbound from the organization to the Internet, which may increase exposure from a security policy perspective. And many servers may issue redundant queries for the same resolution information, reducing efficiencies. Therefore, our next section highlights the use of a set of dedicated caching servers through which all outbound queries can be issued.

11.6.2 Dedicated Caching Servers

Dedicated caching servers can serve as funnel points to resolve queries from internal hosts for information outside of the internal name space. Our internal servers will resolve ipamworldwide.com. queries for internal clients, but Internet web sites and email addresses will require resolution using DNS servers supporting respective name

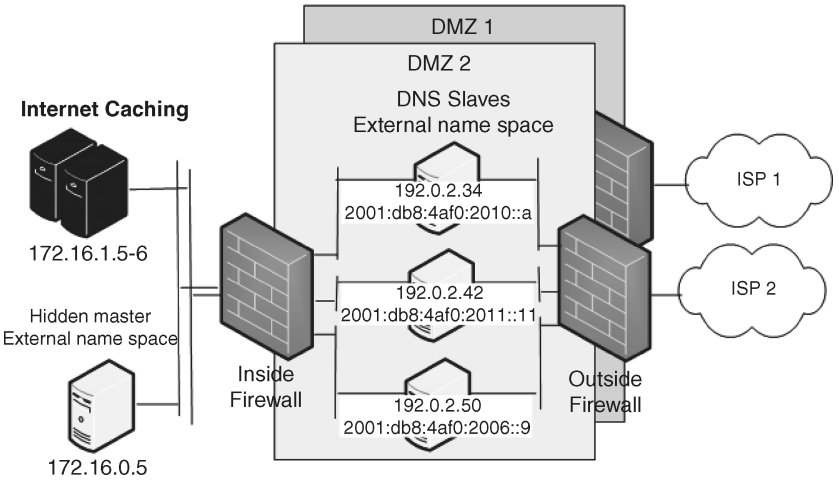

Figure 11.7. Addition of caching servers for external resolution.

spaces. The deployment of dedicated caching servers helps reduce outbound queries and simplifies configuration of firewalls for Internet DNS queries from internal sources. Other name servers within the organization will *forward* queries to these caching name servers when they are unable to resolve directly from authoritative configuration or their own cache.

Dedicated caching servers should be deployed in a high availability configuration, due to the reliance on this server for resolving Internet queries on behalf of internal hosts. Since these caching servers will frequently send and receive Internet traffic, they should be deployed close to Internet connections. Adding this to our previous external DNS figure, Figure 11.7 illustrates deployment of a high availability pair within the internal network but relatively close to the Internet connection. If you have two diverse Internet connections, it's a good idea to deploy a server or pair near each connection, though only one pair is shown in Figure 11.7.

While the external servers resolve queries for your public information for external queriers, the caching servers resolve external information on behalf of your internal clients. The Internet Caching name servers' IP addresses need to be added to the firewall permit lists to enable resolution of Internet hostnames for internal clients. The use of one or a small number of such name servers enables specification of only these few addresses instead of every DNS server address within the organization that would otherwise execute iterative queries.

Since the server is caching responses for all internal clients, the resolution efficiency tends to be higher as the server will be more likely to have query response information in its cache over time. However, vulnerability to cache poisoning and the use of inaccurate or malicious resolution information can create misdirected application connections. Make sure your firewall does not derandomize UDP port numbers used in outbound DNS queries. Also consider configuring DNSSEC validation options (trusted keys) on these

TABLE 11.3. Example Updated Outside Firewall Configuration for DNS Messages

Message and Direction	Control	Source Address	Source Port	Destination Address	Destination Port
DNS queries from the Internet	Allow	Any	>1023	192.0.2.32/27	53
Responses to DNS queries	Allow	192.0.2.32/27	53	Any	>1023
Internet caching server queries	Allow	NAT {172.16.1.5}	>1023	Any	53
Responses to Internet caching server queries	Allow	Any	53	NAT {172.16.1.5}	>1023
All others	Deny	Any	Any	Any	Any

TABLE 11.4. Example Updated Inside Firewall Configuration for DNS Messages

Message and Direction	Control	Source Address	Source Port	Destination Address	Destination Port
Queries from slaves to master (e.g., refresh queries)	Allow	192.0.2.32/27	>1023	172.16.0.5	53
Responses from master to slaves	Allow	172.16.0.5	53, 1053	192.0.2.32/27	>1023
Internet caching server queries	Allow	172.16.1.5	>1023	Any	53
Responses to Internet caching server queries	Allow	Any	53	172.16.1.5	>1023
All others	Deny	Any	Any	Any	Any

servers as we'll discuss in Chapter 13. The firewall example configuration we discussed earlier could be updated as shown in Tables 11.3 and 11.4. We use "NAT{172.16.1.5}" to denote the IP address resulting from NAT'ing the caching server's IP address. One such entry should be made for each server.

Important aspects of the caching DNS server configuration include the following:

- As a caching server, the server is not authoritative for any zones. The only zone file to configure (or simply include) on this server is the root-hints.file.
- Allow queries from internal sources only.
- Disallow dynamic updates and zone transfers.
- Cache management options can be used based on server resources and caching policies.

- Keep caching servers updated with security patches and upgrades as they are exposed to cache poisoning and other "man-in-the-middle" attacks. This is discussed in the next chapter.
- Configure trusted or managed keys if DNSSEC validation is desired (details in Chapter 13)
- Other internal DNS servers will forward queries they cannot resolve through configuration or cache to these Internet caching servers.

Caching Server Configuration Example. An example named.conf configuration for this type of server follows:

```
acl internal-nets { 10.0.0.0/8; 172.16.0.0/12;
    2001:db8:4af0::/48; } ;
options {
  directory "/opt/named/dns/etc";
  recursion yes;
  version "hidden";
  allow-query { internal-nets; };
  allow-transfer { none; };
  allow-update { none; };
};
zone "." {
      type hint;
      file "Internet-root-hints.file";
};
```

The "Internet-root-hints.file" file should be the standard NIC root hints file pointing to the Internet root DNS servers. As this server builds up its cache, it will respond to internal queries using the cache first, then querying down the domain tree as appropriate starting with the root servers.

Resulting Internal Server Configuration Changes. Given the change in strategy from escalating nonauthoritative resolutions to Internet (or internal) roots to the use of caching servers, the following configuration changes are required for all Internal–Internal Category server configurations we've discussed thus far.

- Remove the following root zone statement block referring to the hints file

```
zone "." {
    type hint
    file root-hints.txt
}
```

- Enable forwarding to the caching servers instead of using the hints file. This is accomplished by inserting the following statements into the options{} block of each server's named.conf file

```
options {
. . .
    forwarders { 172.16.1.5; 172.16.1.6;};
    forward only;
. . .
}
```

This configuration instructs the DNS server to forward all queries to our caching servers. Forwarding can be configured at the global level as illustrated above, at a per-zone level and at a global level with zone-level exceptions by entering a blank forwarders statement (`forwarders {};`) within the respective zone statement block.

Considering our 172.16.40.5 slave server configuration, we can update its named.conf file by removing the root zone block and adding our two forwarding statements as described above. Then we can exempt the ipamworldwide.com zone from forwarding by inserting our blank forwarders statement within the zone block as follows:

```
zone "ipamworldwide.com" {
    type slave;
    forwarders { };
    masters { 172.16.30.5 ; };
    file "bak.ipamworldwide.com";
};
```

With this configuration, queries for resource records within the ipamworldwide.com domain will be resolved by the server itself; other queries will be forwarded to our Internet caching servers.

Another twist on this configuration involves retaining the root zone and hints file configuration on each server while changing the `forward only;` statement to `forward first;` within the named.conf file. This configures the server to attempt to resolve the query initially using forwarding, but failing resolution, to use alternative means such as resolving via the root servers. The `forward only;` statement instructs the server to forward as the first and last resort for resolution.

11.6.3 Extranet Resolution Servers

This scenario includes configuring internal DNS servers to resolve queries from internal clients for extranet partner information. This configuration is the complement of the extranet scenario we discussed in the External–Internal Category. In this particular scenario, there are three configuration possibilities.

Extranet Forward Zone. Building on our discussion of forwarding from the previous section, we can define queries bound for the partner's domain as a zone of type forward. All queries received by the internal DNS servers for resolutions within the partner's domain would be forwarded to the specified partner DNS servers. To implement this scenario for our partner's domain, let's say partner.net, we declare this zone within our named.conf file as follows:

```
zone "partner.net" {
    type forward;
    forwarders { 192.168.100.5; 192.168.200.5; };
    forward only;
};
```

Extranet Stub Zone. A stub zone is a special form of a slave zone whereby only the NS and glue records for the zone are transferred and maintained on the server, not the entire zone resource record contents. Like a stub resolver, a stub zone is configured with "who to ask" for queries regarding the given zone and not to provide the answer directly as a slave zone would. In the particular case of an extranet link, a stub zone could be created for our partner domain, partner.net.

```
zone "partner.net" {
    type stub;
    masters { 192.168.249.11;};
};
```

Extranet Delegation Via Internal Root. If you're using internal root servers, you can define the partner.net zone as delegation only and refer to the partner's DNS server(s) by configuring corresponding NS and glue records. In our prior internal root zone file example, we referred to an internal set of DNS servers to resolve queries to the partner.net domain. These referenced servers would need to configure this zone as forward or stub; alternatively, the root zone file could point directly to the partner's servers, though this may become a maintenance issue if your partner changes server IP addresses or hostnames.

11.7 CROSS-ROLE CATEGORY

11.7.1 Split View DNS Servers

The views feature of BIND 9 enables deployment of multiple versions of a zone to a DNS server. The server can filter the query based on an address match list on the query source and destination and whether the query is recursive or not. This filtering determines which version or view of the zone will be searched for resolution of the query. Each view is

configured with its respective version of the zone file. A common example of using views is for "split DNS," or providing internal versus external versions of an organization's name space. While deployment of a single set of DNS servers to handle both internal and external queries is not recommended, it is a more practical approach for smaller organizations. But views can also be used on internal name space to restrict access to certain host resolutions to particular resolver clients.

In the example that follows, we'll define a new subdomain, `hr.ipamworldwide.com`, defining two versions of the domain, and then associate these versions with corresponding views on the DNS server. The concept is to provide general access to the hr portal server (hostname `portal`), but restrict access to other hosts within the `hr.ipamworldwide.com` subdomain to only HR users on the network.

The first thing we'll consider is the zone files themselves. Let's create our general or default domain version in a zone file called `db.hr.ipamworldwide.com.default` and another file with broader visibility for HR clients called `db.hr.ipamworldwide.com.hr`. Note that if the `hr` subdomain is not delegated, two versions of the parent zone file would be needed, one for each view of the subdomain.

Our `db.hr.ipamworldwide.com.default` file would contain a limited number or even just an A record (beyond the SOA, NS and glue records for the zone):

```
portal      IN    A      172.16.4.24
```

Whereas the `db.hr.ipamworldwide.com.hr` file would contain more resource records such as

```
payroll     IN    A      172.16.4.10
benefits    IN    A      172.16.4.14
empdb       IN    A      172.16.4.22
portal      IN    A      172.16.4.24
promo       IN    A      172.16.4.30
```

Note that two versions of the `4.16.172.in-addr.arpa.` domain are required as well, each corresponding to the IP addresses and hostnames exposed in each view. We can use the same `default` and `hr` suffix on the db file names.

```
db.172.16.4.default file excerpt:
. . .
24          IN    PTR    portal.hr.ipamworldwide.com.
. . .
db.172.16.4.hr file excerpt:
. . .
10          IN    PTR    payroll.hr.ipamworldwide.com.
14          IN    PTR    benefits.hr.ipamworldwide.com.
22          IN    PTR    empdb.hr.ipamworldwide.com.
```

```
24          IN    PTR   portal.hr.ipamworldwide.com.
30          IN    PTR   promo.hr.ipamworldwide.com.
. . .
```

Now that we've created our zone files for each version, we can associate them with the corresponding views on the DNS server. Let's use `hrdns.hr.ipamworldwide.com` as our master DNS server. Here's a partial example `named.conf` file for this server:

```
acl human-res {172.16.4.0/23; } ;

view "hr" {
       match-clients { "human-res"; };
       match-destinations { localnets; };

      zone "hr.ipamworldwide.com." {
            type master;
            file "db.eng.ipamworldwide.com.hr";
      };

      zone "4.16.172.in-addr.arpa." {
             type master;
             file "db.172.16.4.hr";
      };
};

view "default" {
       match-clients { any; };

       zone "hr.ipamworldwide.com." {
          type master;
          file "db.hr.ipamworldwide.com.default";
       };

       zone "4.16.172.in-addr.arpa." {
      type master;
      file "db.172.16.4.default";
      };
};
```

Note that the order of view statements within named.conf is important! The first view matching the query will be used to determine which zone file's information will be accessed. Hence we define our default view, which matches all client queries, after the more discriminating `hr` view statement. We stated that this was a partial definition. Special handling is required to properly perform zone transfers to slave servers likewise configured with these views. After all, an update to a resource record related to the portal.hr.ipamworldwide.com host could be interpreted as falling into either view; which one should be updated?

Assuring notifies, updates, and transfers among the correct views requires manipulation of the match-clients ACL as well as the use of query-source, notify-source, and transfer-source statements. By defining the source IP address for each of these functions and applying a corresponding ACL, interserver communications can be funneled to the correct view. In expanding on our example above, we need to modify our "human-res" ACL to also negate the IP address(es) used in the default view's source statements. That is, notifies from the master for the default view need to be blocked from the human-res view, given the importance of order. Likewise transfer requests from the slave for the default view need to be blocked from the human-res view. In addition, the query-source on both sets of servers should likewise be configured.

This is illustrated in Figure 11.8. Using letters to symbolize the *-source* options on each server for brevity, we can visualize that a notify regarding a zone update for View 2

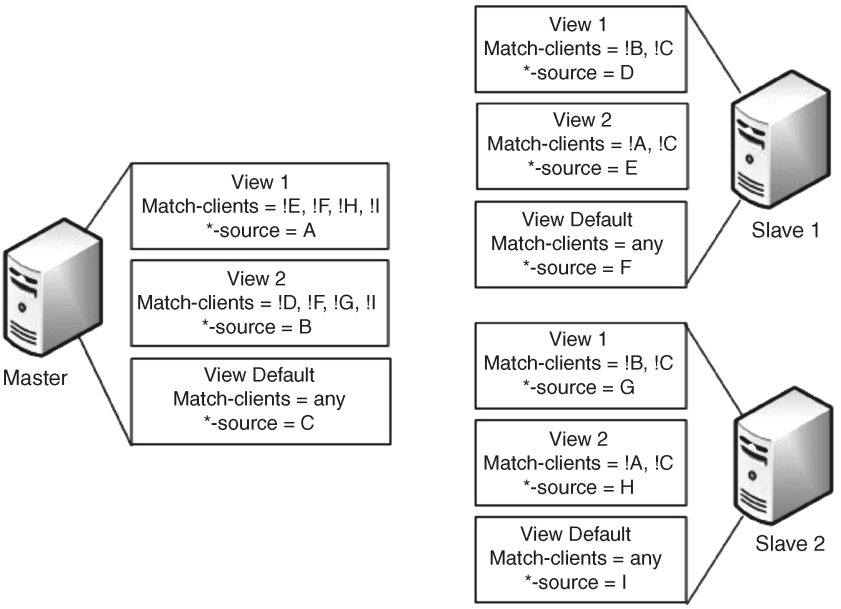

Figure 11.8. Views configuration for interserver communications.

from the Master to Slave 1, would first be matched against View 1 on the slave. The notify from View 2 will use source IP address B. On Slave 1, B is negated within the match-clients portion of the view definition, but falls within the match criteria of View 2 and is thence applied. Likewise, a transfer request from Slave 2 sourced on IP address I for the Default View to the Master would be negated within View 1 and View 2, but applied to the Default View correctly.

Applying this model to our simpler configuration example above with a master name server using IP source addresses 172.16.4.101 (e.g., address A in Figure 11.8) and 172.16.4.102 (address C), and a single slave using source addresses 172.16.5.201 (address D) and 172.16.5.202 (address F) we arrive at the following for the master:

```
acl human-res {!{!172.16.4.0/23; any;}; !172.16.5.202; };

view "hr" {
       match-clients { "human-res"; };
       match-destinations { localnets; };
       query-source address 172.16.4.101;
       zone "hr.ipamworldwide.com." {
             type master;
             notify-source 172.16.4.101;
             file "db.eng.ipamworldwide.com.hr";
      };

       zone "4.16.172.in-addr.arpa." {
           type master;
           notify-source 172.16.4.101;
           file "db.172.16.4.hr";
      };
};

view "default" {
       match-clients { any; };
       query-source address 172.16.4.102;
```

* That is, query-source and notify-source options on the master and query-source and transfer-source on each slave.

```
        zone "hr.ipamworldwide.com." {
              type master;
              notify-source 172.16.4.102;
              file "db.hr.ipamworldwide.com.default";
       };

       zone "4.16.172.in-addr.arpa." {
              type master;
              notify-source 172.16.4.102;
              file "db.172.16.4.default";
       };
};
```

The following reflects the corresponding slave server's view configuration:

```
acl human-res-slave {!{!172.16.4.0/23;any;}; !172.16.4.102; };

options {
       directory "/opt/dns/etc";
};

view "hr" {
       match-clients { "human-res-slave"; };
       match-destinations { localnets; };
          query-source address 172.16.5.201;
          zone "hr.ipamworldwide.com." {
                type master;
                notify-source 172.16.5.201;
                masters { 172.16.4.101; };
          };

          zone "4.16.172.in-addr.arpa." {
                type master;
                notify-source 172.16.5.201;
                   masters { 172.16.4.101; };
          };
};
```

```
view "default" {
       match-clients { any; };
       query-source address 172.16.5.202;
       zone "hr.ipamworldwide.com." {
             type master;
             notify-source 172.16.5.202;
             masters { 172.16.4.102; };
      };

      zone "4.16.172.in-addr.arpa." {
             type master;
             notify-source 172.16.5.202;
             masters { 172.16.4.102; };
      };
};
```

11.7.2 Deploying DNS Servers with Anycast Addresses

Configuring DNS servers with anycast addresses enables multiple DNS servers to utilize a common IP address. Recall that an anycast address is an address assigned to multiple interfaces, typically on different nodes. Anycast is used when attempting to reach any one of the anycast addressable hosts without caring which host is reached. The routing infrastructure handles routing metric updates to track reachability and routing to the nearest host configured with the destination anycast address. Figure 11.9 illustrates an example with three DNS servers configured with anycast address 10.4.23.1.

As depicted in Figure 11.9, Router 1 has three routes to anycast address 10.4.23.1/32, corresponding to our three servers. The closest server is that homed on Router 2 and is two hops from Router 1. The next closest server is home on Router 5 and is reachable in three hops via Router 4. Lastly, the server connected to Router 6 is reachable in four hops via either Router 2 or 4. The logical view from Router 1's perspective is illustrated in Figure 11.10, where the anycast IP address is considered a single destination, reachable via multiple paths.

Anycast Benefits. Deploying anycast provides a number of benefits:

- Simplified resolver configuration
- Improved resolution performance

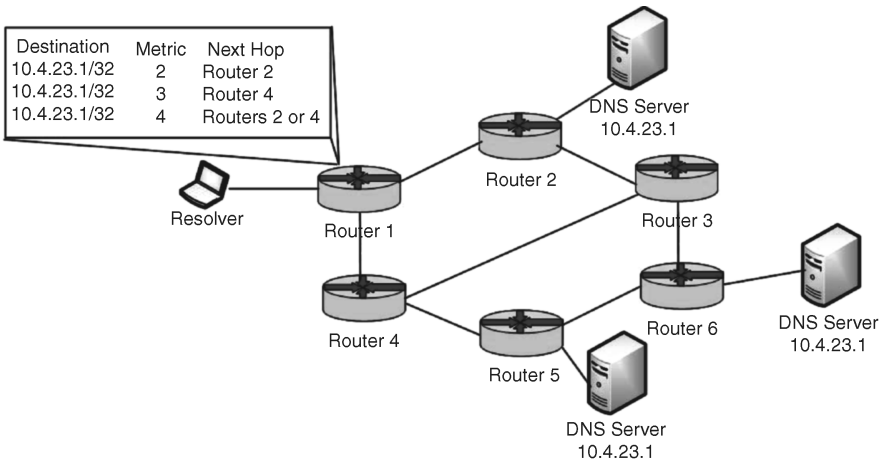

Figure 11.9. Anycast routing table example.

- High availability DNS services
- Resilience from DNS denial of service attacks

Resolvers configured with the DNS servers' anycast address would have their queries routed to the nearest DNS server configured with that anycast address. Thus, regardless of where the resolver host connects to the network, the same anycast IP address may be used by the resolver to locate a DNS server. This localized query process can also improve performance of the resolution process. A query to a DNS anycast address should be routed to the closest DNS server, thereby reducing the round trip delay portion of the overall query process.

The outage of a DNS server can be communicated (by absence of communication) to the routing infrastructure in order to update routing tables accordingly. This requires the

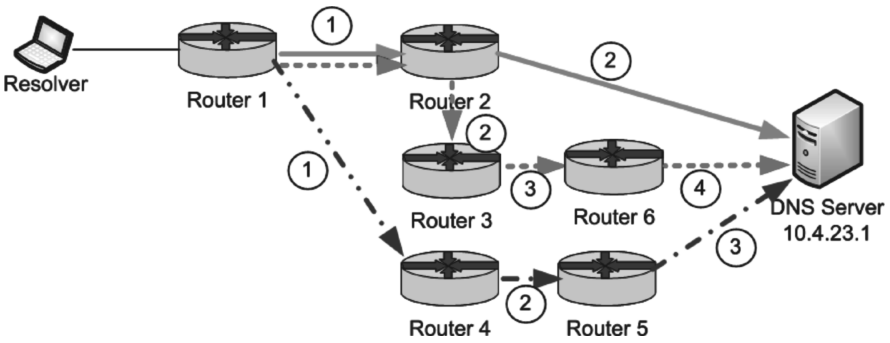

Figure 11.10. Logical routing perspective from Router 1 showing hop counts.

DNS server run a routing daemon using the routing protocol of choice to communicate reachability to the local router. Participation in routing protocol updates enables the local router to update its routing table with an appropriate metric and to pass this on to other routers via the routing protocol. Depending on the deployment of the DNS server, internal or external, a corresponding interior or exterior routing protocol would need to be running on the DNS server. The server simply needs to communicate that its anycast address is reachable. This is typically performed by assigning one of the server loopback addresses* as the anycast address and running a routing daemon on one or more ports advertising reachability to the anycast address. It would be especially useful if this routing update was linked to the status of the DNS daemon or service on the server, though application status is not generally considered when communicating IP address reachability.

Deploying anycast affords mitigation against denial of service attacks as evidenced by the distributed denial of service (DDoS) attack on multiple root servers on February 6, 2007 (136). Of the six root servers targeted, the two most severely affected were those that had not yet implemented anycast. The other four root servers, having deployed anycast, enabled the spreading of the attack across more physical servers. Thus, a DDoS attack on the I-root server, which did not have anycast in place, severely impacted the ability of the server to respond to legitimate queries, while the attack on the F-root server, which had over 40 servers sharing the F-root anycast address, distributed the impact of the attack across these servers. This form of load sharing enabled the F-root server(s) to continue processing legitimate queries while suffering a barrage of artificial requests.

Anycast Caveats. While anycast provides many benefits, consider the constraints and caveats of deploying anycast. Because resolvers may query any DNS server configured with the anycast address at a given time, it's important that the resolution information configured on the server be consistent. For example, the implementation on Internet root servers consists of a set of master servers with static information. These root servers do not accept dynamic updates. If anycast is desired for dynamic zones, then each server must have a unicast address in addition to its anycast address†. This enables updates to be directed to the master's unicast address, which may in turn notify its slaves via their respective unicast addresses. A hidden master configuration can be used with the slaves configured with anycast addresses.

Another consideration is the requirement to run a route daemon on your DNS servers configured with anycast addresses. While routing of packets to anycast addresses is primarily a routing function, the unreachability of a DNS server host may result in lost query attempts. Such would be the case if static routes are used to configure routers with fixed metrics for the DNS servers configured with a common anycast address. Should a server become unavailable, the serving router has no way to detect this and would not reroute packets destined for the anycast address. Therefore, incorporating a routing

* The term "loopback address" here refers to the software loopback address commonly implemented in routers and servers as the "box address" reachable on any of its interfaces.

† Every anycast server will require a unicast address for administration, but to support dynamic zones, an additional unicast address is required to provide an interface for updates, notify's and zone transfers.

daemon on the DNS server improves overall robustness. Should a server fail, the local router will determine that it is no longer reachable and will update its routing table and those of other routers via routing protocol updates. Internet root servers support BGP, given their deployment on the global Internet, though deployment within organizations will likely require support for OSPF, IGRP, or the interior routing protocol of choice.

Lastly, troubleshooting is a little more challenging when using anycast. Debugging a bogus response from a server's anycast address is difficult given the server ambiguity. To identify which anycast-addressed server is troublesome, it's a good idea to configure the server identification with the BIND server-id option. You can define a string identifier or just use the hostname argument to use the server's hostname. This value is retrievable by issuing a query with qname="ID.SERVER", qtype=TXT with qclass=CHAOS. Using the dig utility, this looks like

dig id.server chaos txt @<anycast-address>

Perhaps a better way is to use the dig +nsid argument with a query so you can correlate a bad response with the server identification in one transaction.

dig +nsid <query> @<anycast-address>

For more details on anycast configurations, see Refs (137) and (138).

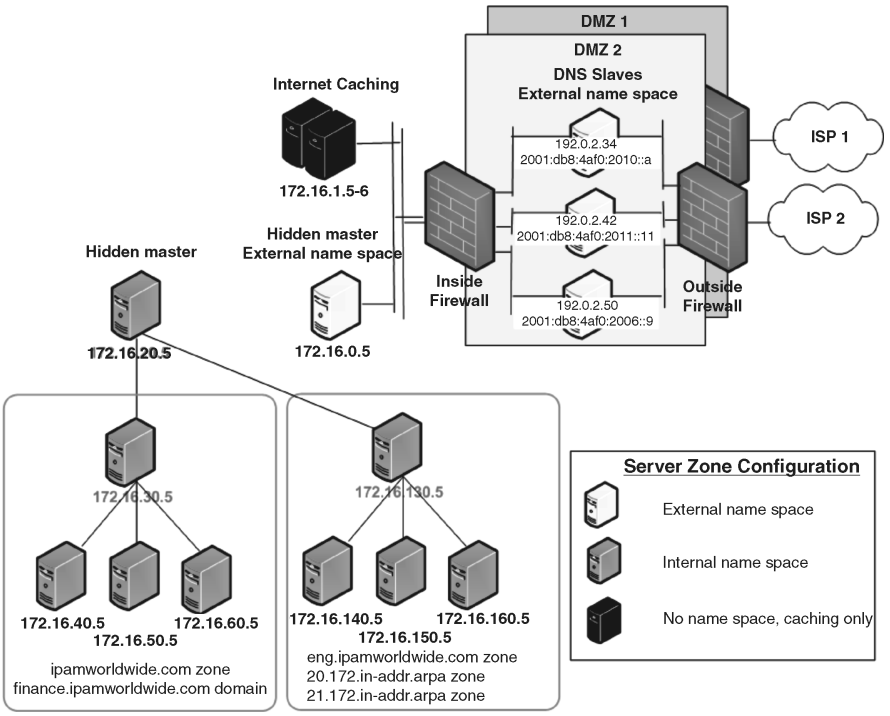

Figure 11.11. IPAM Worldwide DNS server deployments.

11.8 PUTTING IT ALL TOGETHER

We've presented numerous building block scenarios in this chapter addressing a wide variety of configurations, each targeted at addressing a specific DNS resolution purpose. Depending on the size and scale of your IP network, you may choose to implement several building block scenarios for different applications on your network. Just remember that there are no true cookie-cutter answers; each of these scenarios should be evaluated based on your individual needs. The intention here is to provide some guidelines in a modular fashion to help simplify the overall deployment design process.

We've intentionally defined each of these scenario building blocks with its own set of discrete DNS servers. This role-based approach facilitates modularity but also helps with troubleshooting and in managing security policies. Instead of having a common server handle both internal and external resolutions for example, segregating these functions across multiple servers provides physical and functional separation.

Most organizations will deploy at minimum the external DNS scenario (External–External Category) and the internal DNS (Internal–Internal) building blocks. Those with partners may also add the extranet scenarios covering both inbound and outbound query resolutions. Internet caching servers may be deployed to enable funneling of outbound Internet resolutions and to build up rich cache information over time. Views and/or anycast may also be configured based on your needs. The potential number of scenario combinations is endless. Figure 11.11 illustrates one possible incarnation of a holistic DNS infrastructure for IPAM Worldwide.