DNS Basic Name Resolution Techniques: Iterative and Recursive Resolution

Conventional name resolution transforms a DNS name into an IP address. At the highest level, this process can be considered to have two phases. In the first phase, we locate a DNS name server that has the information we need: the address that goes with a particular name. In the second phase, we send that server a request containing the name we want to resolve, and it sends back the address required.

The Difficult Part of Name Resolution: Finding The Correct Server

Somewhat ironically, the second phase (the actual mapping of the name into an address) is fairly simple. It is the first phase—finding the right server—that is potentially difficult, and comprises most of the work in DNS name resolution. While perhaps surprising, this is a predictable result of how DNS is structured. Name information in DNS is not centralized, but rather distributed throughout a hierarchy of servers, each of which is responsible for one zone in the DNS name space. This means we have to follow a special sequence of steps to let us find the server that has the information we need.

The formal process of name resolution parallels the tree-like hierarchy of the DNS name space, authorities and servers. Resolution of a particular DNS name starts with the most general part of the name, and proceeds from it to the most specific part. Naturally, the most general part of every name is the *root* of the DNS tree, represented in a name as a trailing "dot", sometimes omitted. The next most-specific part is the top-level domain, then the second-level domain and so forth. The DNS name servers are "linked" in that the DNS server at one level knows the name of the servers that are responsible for subdomains in zones below it at the next level.

Suppose we start with the fully-qualified domain name (FQDN) "C.B.A.". Formally, every name resolution begins with the root of the tree—this is why the root name servers are so important. It's possible that the root name servers are authoritative for this name, but probably not; that's not what the root name servers are usually used for. What the root name server ***does*** know is the name of the server responsible for the top-level domain, "A.".

The name server for "A." in turn may have the information to resolve "C.B.A." It's still fairly high-level, though, so "C.B.A" is probably not directly within its zone. In that case, it will not know the address we seek, but it will know the name of the server responsible for "B.A.". In turn, that name server may be authoritative for "C.B.A.", or it may just know the address of the server for "C.B.A.", which will have the information we need. As you can see, it is very possible that several different servers may be needed in a name resolution.

Key Concept: Since DNS name information is stored as a distributed database spread across many servers, name resolution cannot usually be performed using a single request/response communication. It is first necessary to find the correct server that has the information that the resolver requires. This usually requires a sequence of message exchanges, starting from a root name server and proceeding down to the specific server containing the resource records that the client requires.

© 2001-2004 Charles M. Kozierok. All Rights Reserved.

DNS Name Resolution Techniques

The DNS standards actually define two distinct ways of following this hierarchy of servers to discover the correct one. They both eventually lead to the right device, but they differ in how they assign responsibility for resolution when it requires multiple steps.

Iterative Resolution

When a client sends an iterative request to a name server, the server responds back with either the answer to the request (for a regular resolution, the IP address we want) ***or*** the name of another server that has the information or is closer to it. The original client must then *iterate* by sending a new request to this referred server, which again may either answer it or provide another server name. The process continues until the right server is found; the method is illustrated in Figure 243.

Recursive Resolution

When a client sends a recursive request to a name server, the server responds back with the answer if it has the information sought. If it doesn't, the server takes responsibility for finding the answer by becoming a client on behalf of the original client and sending new requests to other servers. The original client only sends one request, and eventually gets the information it wants (or an error message if it is not available). This technique is shown in Figure 244.

Contrasting Iterative and Recursive Resolution

To help explain the difference between these methods, let's take a side-trip to a real-world case. Suppose you are trying to find the phone number of your old friend Carol, with whom you haven't spoken in years. You call your friend Joe; he doesn't have Carol's number, but he gives you John's number, suggesting you call him. So you dial up John; he doesn't have the information but he knows the number of Carol's best friend, Debbie, and gives that to you. You call Debbie and she gives you Carol's information. This is an example of an iterative process. In contrast, suppose you called Joe and Joe said “I don't know, but I think I know how to find out “. He called John and then Debbie and called you back with the phone number. That would be like recursive resolution.

So in essence, iteration is doing the job yourself, while recursion is “passing the buck”. You might think that everyone would always want to use recursion since it makes “the other guy” do the work. This is true, but “passing the buck” is not considered good form if it is not done with permission. Not all name servers support recursion, especially servers near the top of the hierarchy. Obviously, we don't want to bog down the root name servers and the ones that handle “.COM” and other critical TLDs with doing recursion. It is for this reason that clients must *request* that name servers perform recursion for them.

One place where recursion ***is*** often used is with the local name server on a network. Rather than making client machine resolvers perform iterative resolution, it is common for the resolver to generate a recursive request to the local DNS server, which then generates iterative requests to other servers as needed. As you can see, recursive and iterative

 © 2001-2004 Charles M. Kozierok. All Rights Reserved.

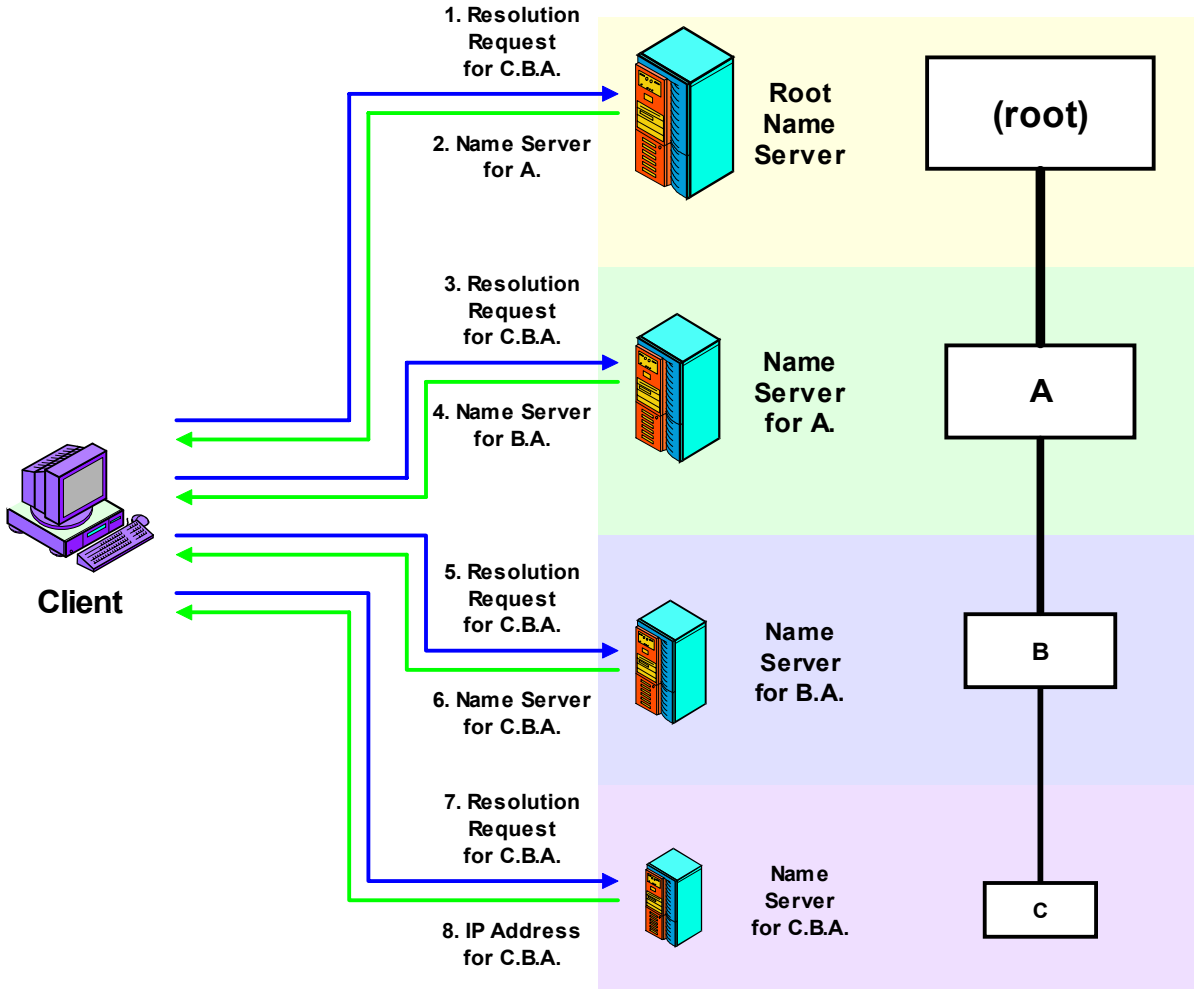

Figure 243: Iterative DNS Name Resolution

In this example, the client is performing a name resolution for "C.B.A." using strictly iterative resolution. It is thus responsible for forming all DNS requests and processing all replies. It starts by sending a request to the root name server for this mythical hierarchy. That server doesn't have the address of "C.B.A.", so it instead returns the address of the name server for "A.". The client then sends its query to that name server, which points the client to the server for "B.A.". That name server refers the client to the name server that actually has the address for "C.B.A.", which returns it to the client. Contrast to Figure 244.

requests can be combined in a single resolution, providing significant flexibility to the process as a whole. This is demonstrated in a more realistic example in the topic detailing the DNS name resolution process.

Again, remember that for the purpose of understanding resolution, a DNS server can in fact act as a client. As soon as a DNS server accepts a recursive request for resolution on a name it cannot resolve itself, it becomes a client in the process. I should also point out that it is common for resolvers to know the names of not one but two local DNS servers, so if a problem occurs reaching the first they can try the second.

© 2001-2004 Charles M. Kozierok. All Rights Reserved.

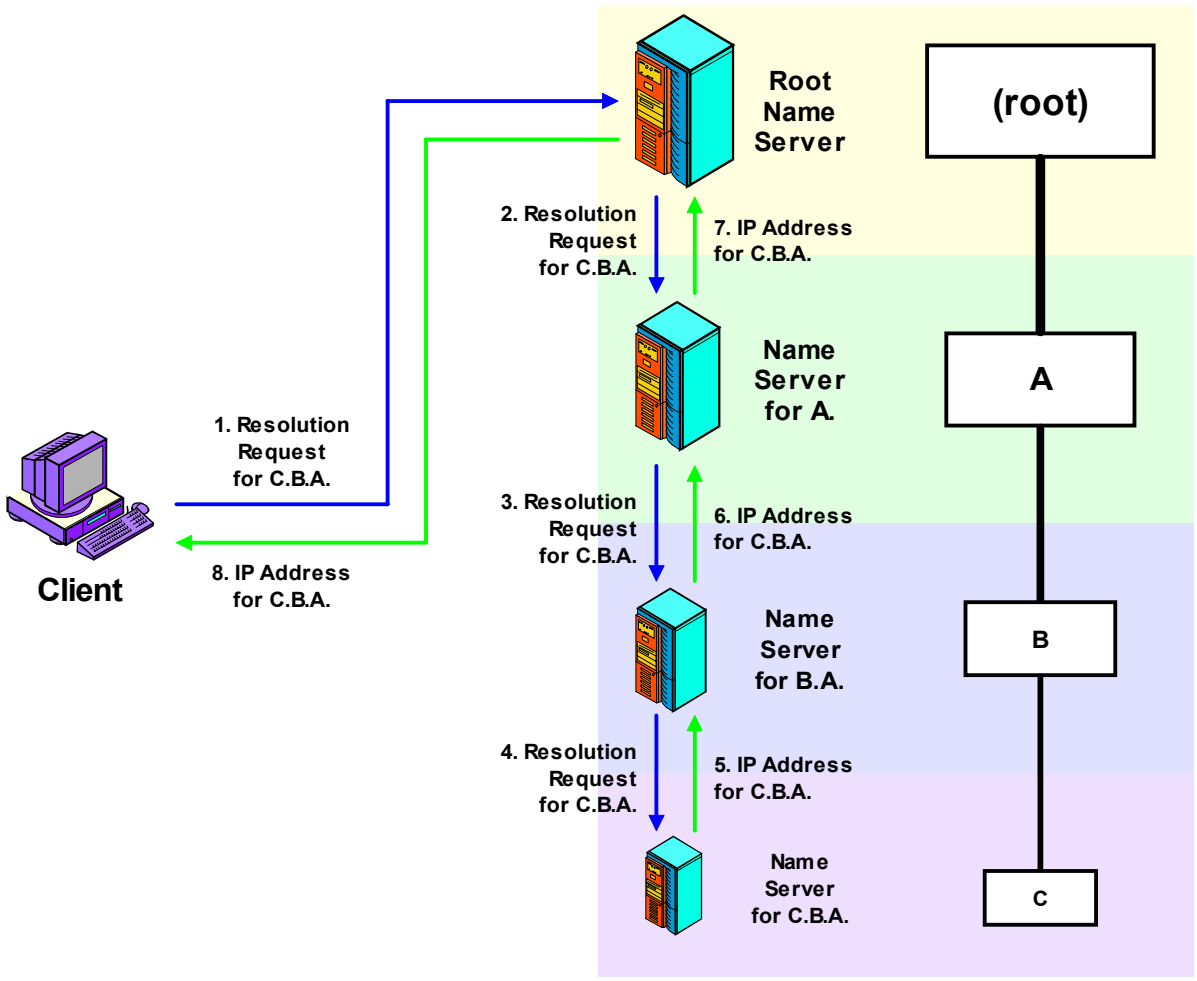

Figure 244: Recursive DNS Name Resolution

This is the same theoretical DNS resolution that I showed in Figure 243, but this time, the client asks for the name servers to perform recursive resolution and they agree to do so. As in the iterative case, the client sends its initial request to the root name server. That server doesn't have the address of "C.B.A.", but instead of merely returning to the client the address of the name server for "A.", it sends a request to that server itself. That name server sends a request to the server for "B.A.", which in turn sends a request to the server for "C.B.A.". The address of "C.B.A." is then carried back up the chain of requests, from the server of "C.B.A." to that of "B.A.", then "A.", then the root, and then finally, back to the client.

Key Concept: The two methods of name resolution in DNS are *iterative resolution* and *recursive resolution*. In iterative resolution, if a client sends a request to a name server that does not have the information the client needs, the server returns a pointer to a different name server and the client sends a new request to that server. In recursive resolution, if a client sends a request to a server that doesn't have the requested information, that server takes on the responsibility for sending requests to other servers to find the necessary records, then returns them to the client. A server doing this takes on the role of client for its requests to other servers.

© 2001-2004 Charles M. Kozierok. All Rights Reserved.