CCNA Exam Prep: Back to Networking Basics with Hank Preston and Patrick Gargano

DNS Deepdive: Exploring a Backbone of Network Communication

The Internet would never have taken off like it did if users were expected to access their favorite websites with urls that looked like `https://192.168.34.56/socialmedia/homepage.html`. Or in the IPv6 days `https://2001:db8:abcd:8f42:78::aef/socialmedia/homepage.html`. Instead users can use easy to remember domain names like `https://www.example.com`. The network protocol that makes this all work is DNS, or the Domain Name System. As a CCNA candidate you are expected to understand the important role DNS plays within a network.

> 4.3 Explain the role of DHCP and **DNS** within the network

In this lab you will explore DNS with some hands on exercises focused on:

- Why DNS is part of nearly every network communication
- How DNS translates names -> IPs
- How DNS settings are assigned to hosts
- How an IOS router can be a basic DNS server

Setup and Scenario

This set of lab based demonstrations includes a small network made up of a "Branch" and "Data Center" location. The Data Center provides a `dnsserver` for the network, as well as hosts two web applications on the `webserver`. The branch router `br1-rtr` provides IP addresses and DNS server details to hosts at Branch 1 through DHCP.

The DNS server is *authoritative* for the domain `example.com`. This means that it is configured with the DNS records for all hosts and services such as `www.example.com`. It will also *recursively* perform lookups for all domain queries that it is NOT authoritative for. These lookups will be done through other DNS servers known as *forwarders*. In this case, Cisco's Umbrella (OpenDNS) servers will be used for lookups to domains such as `cisco.com`.

The Web Server is running two web applications. `www.example.com` and `app.example.com`. These example applications are simple webpages that indicate which application you have accessed.

We will use `br1-host` to perform our exploration of DNS through the following tools:

- `ping` - A command line utility for testing reachability and connectivity
- `dig` - A command line utility for performing DNS lookups
- `curl` - A command line web client

> **IMPORTANT NOTE:** `dig` and `curl` are NOT installed by default on our host. Install them by:
>
> 1. Open the console for `br1-host1`
> 2. Install them with the command `sudo apk add bind-tools curl`

In our final lab step, we will enable the branch router `br1-rtr` to be a local DNS server for the branch.

> Note: The credentials for all devices are `cisco / cisco`.

*Be sure to **START** the lab before continuing to the demo labs.*

Seeing DNS in action with basic lookups and resolution

Verify basic DNS with Ping

1. Open the console for `br1-host` and attempt to ping the DNS server using the FQDN (fully qualified domain name) `dnsserver.example.com`.

   ```
   ping -c 3 dnsserver.example.com
   ```

   ```
   # Output
   PING dnsserver.example.com (192.168.0.11): 56 data bytes
   64 bytes from 192.168.0.11: seq=0 ttl=42 time=1.679 ms
   64 bytes from 192.168.0.11: seq=1 ttl=42 time=2.356 ms
   64 bytes from 192.168.0.11: seq=2 ttl=42 time=2.322 ms
   ```

   - Notice how ping indicates the replies from the IP address and not the name

2. Now try pinging the webserver using its FQDN `webserver.example.com`.

   ```
   ping -c 3 webserver.example.com
   ```

   - You should get a similar result.

3. Excellent job, you've now verified that DNS services are functioning on our network.

4. The Linux DNS resolver determines the DNS servers by reading the contents of the file `/etc/resolv.conf`. Display the file.

   ```
   cat /etc/resolv.conf
   ```

   ```
   # Output
   search example.com
   nameserver 192.168.0.11
   ```

5. We see that the `nameserver` is set to be `192.168.0.11`. This was configured by the DHCP client and populated by the options returned by `br1-rtr`.

    > You can look at these settings on the router by running `show run | sec dhcp`

6. But what is that `search example.com` setting about? It was configured by the DHCP server as well, and provides the default "domain name" to use for lookups when one isn't provided. This is useful to allow users to NOT have to specify the FQDN for common applications.

7. Test this by pinging the web server again, but just with the "hostname".

    ```
    ping -c 3 webserver
    ```

    ```
    # Output
    PING webserver (192.168.0.12): 56 data bytes
    64 bytes from 192.168.0.12: seq=0 ttl=62 time=1.785 ms
    64 bytes from 192.168.0.12: seq=1 ttl=62 time=2.463 ms
    64 bytes from 192.168.0.12: seq=2 ttl=62 time=2.428 ms
    ```

    - Nice and convenient!

Using DNS to access web applications

We're going to move from basic connectivity testing to how DNS is a key part of web applications, one of the most common types of applications used today.

1. First, start a Packet Capture on the link between `br1-rtr` and `dc-rtr` to capture and inspect DNS packets.
    1. Right click the link and choose Packet Capture.

        > Hint: Click the **+** in the Panes view to have the packet capture and the terminal visible at the same time

    2. Click the **SETTINGS** button.
    3. Filter for just DNS packets by adding the BPF filter `udp port 53` and clicking **APPLY**.
    4. Click **START** to begin capturing packets.

2. Return to the terminal for `br1-host1` and use `curl` to access `www.example.com`.

    ```
    curl www.example.com
    ```

    ```
    # Output
    <html>
        <head>
            <title>www.example.com</title>
        </head>
        <body>
            Welcome to www.example.com!
        </body>
    </html>
    ```

    - Excellent, we were able to reach our web application.

3. Look at the output from the packet capture. You should see 4 packets.

    1. A **Standard query** from the host to the server for `A www.example.com`
    2. A **Standard query** from the host to the server for `AAAA www.example.com`
    3. A **Standard query response** from the server to the host for `A www.example.com`

    4. A **Standard query response** from the server to the host for `AAAA www.example.com`

        > The `A` queries are for IPv4. The `AAAA` (called "quad-A") are for IPv6.

4. Click the response for the `A www.example.com` and look at the packet details.

    1. Expand `Domain Name System (response)`
    2. Expand `Answers`
    3. You should see two answers.
        1. `www.example.com: type CNAME`
        2. `webserver.example.com: type A`
    4. `CNAME` types are DNS records that are "alias's" or pointers to another DNS record. There are many uses for CNAME records, but a common one is to support multiple uniquely named applications on a single network host (ie IP address). Here the cname points to `webserver.example.com`.
    5. `A` types are DNS records that point to a single IPv4 address. Quad-A (`AAAA`) records point to IPv6 addresses.
    6. See how the `addr 192.168.0.12` is listed for the A record - the address for the web server.
5. Understanding the DNS resolution for a "name" is an important step in troubleshooting network application issues. If a host can't resolve the "name" to the correct IP address, the issue is a DNS problem, and not a network connectivity problem.

6. The `dig` utility is useful for undestanding the DNS process for a client. Use it to lookup `app.example.com`.

    ```
    dig app.example.com
    ```

    ```
    ; <<>> DiG 9.18.33 <<>> app.example.com
    ;; global options: +cmd
    ;; Got answer:
    ;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 21339
    ```

```
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
; COOKIE: 902840cbf2a8c22f01000000679fc8acbc5ea1ae651c2b8b (good)
;; QUESTION SECTION:
;app.example.com.               IN      A

;; ANSWER SECTION:
app.example.com.        86400   IN      CNAME   webserver.example.com.
webserver.example.com.  86400   IN      A       192.168.0.12

;; Query time: 10 msec
;; SERVER: 192.168.0.11#53(192.168.0.11) (UDP)
;; WHEN: Sun Feb 02 19:34:04 UTC 2025
;; MSG SIZE  rcvd: 112
```

- Focus on the `;; ANSWER SECTION:`. You can see the same CNAME -> A record results from the packet capture. In fact, you can see those packets captured as well.

7. Now attempt to access the web application with curl.

```
curl app.example.com
```

```
# Output
<html>
    <head>
        <title>app.example.com</title>
    </head>
    <body>
        Welcome to app.example.com!
    </body>
</html>
```

- A similar response here, but now we've accessed the "app" and not "www" application. But both are from the same webserver.

8. So you might be thining, this is "convenient", but we are network engineers. I'll just use the IP address for the application. Let's try that.

```
curl 192.168.0.12
```

```
# Output
<html>
<head><title>404 Not Found</title></head>
<body>
<center><h1>404 Not Found</h1></center>
<hr><center>nginx</center>
</body>
</html>
```

- Uh-oh... It looks like the application isn't reachable with just the IP address. Let's see why.

9. Open up the console for the `webserver` node.

10. Look at the configuration file for `app.example.com`.

```
cat /etc/nginx/http.d/app.example.com.conf
```

- What is listed as the `server_name` in the configuration?

11. Now repeat for `www.example.com`.

```
cat /etc/nginx/http.d/www.example.com.conf
```

- What is listed as the `server_name` in the configuration?

12. Many different applications are often hosted on a single server. Web servers use the name provided as the `Host:` in the web request to identify which application a request belongs to.

13. Return to `br1-host1` and add the "verbose" flag to the curl request.

```
curl -v app.example.com
```

- What is listed for the `Host:` in the request object? (Note: The "request" object is identified with `<` symbols.)

14. So we can see, DNS is ***REQUIRED*** for web applications to function correctly. It isn't just a "nice to have.

Reverse DNS lookups to find the NAME for an IP ADDRESS

Most of the time DNS is used to determine the IP address for a hostname. However, there are times when you might have an IP address and want to know what hostname uses that IP address. This type of lookup is called a "reverse DNS lookup". It may also be called a "pointer" lookup after the record type `PTR` that is involved in this type of query.

1. Open the console for `br1-host1` and perform a reverse lookup for `192.168.0.11`.

```
dig -x 192.168.0.11
```

```
# Output (ANSWER SECTION)
;; ANSWER SECTION:
11.0.168.192.in-addr.arpa. 86400 IN     PTR     dnsserver.example.com.
```

- Notice that the IP address is written in a "strange" format. `11.0.168.192.in-addr.arpa.`.

2. The `-x` argument to `dig` is used to avoid having to manually format the lookup query correctly. But we can do the lookup manually.

   ```
   dig 11.0.168.192.in-addr.arpa. PTR
   ```

   - The results should be the same as the "shortcut" method.

Reverse lookups are particularly useful for network engineers who might first see the IP address of a host, and need to determine the hostname it relates to. However, not all DNS administrators at enterprises setup their DNS infrastructure to support reverse lookups. If your organization does NOT support PTR lookups, talk with your DNS team if it might be possible to setup.

Enabling DNS lookups on a network devices

It is fairly common for standard configurations of routers and switches to DISABLE DNS lookups. But rather than adding `no ip domain lookup`, properly configuring DNS can be a very useful approach.

1. Open the console for `br1-rtr`.
2. Attempt to ping the DC router.

   ```
   ping dc-rtr.example.com
   ```

   ```
   ! Output
   % Unrecognized host or address, or protocol not running.
   ```

3. Let's see if we can fix this error. Add this configuration to the router.

   ```
   ip name-server 192.168.0.11
   ```

4. Try the ping again. Pretty nice right?
5. Try to ping using just the hostname.

   ```
   ping dc-rtr
   ```

6. It didn't work because the router doesn't know what domain name to use for "unqualifed hostname". Configure the router to use `example.com`.

   ```
   ip domain list example.com
   ```

7. Try the hostname based ping again. Even better right?
8. Now you can use DNS names as part of configurations or operational work. For example, try to SSH to the webserver using just the server name.

   ```
   ssh -l cisco webserver
   ```

Watching DNS recursive lookups in action

So far we've been exploring DNS for a single domain, `example.com`, who's "authoritative" server is the server in our data center. But how does DNS work when a user wants to access a service from the public internet. Our DNS server isn't configured with every domain and host on the entire internet. This is where the true power of ***"The Domain Name System"*** comes into play. It provides as a structured network of name servers spanning the globe that allows a DNS server running in out simple CCNA lab to provide answers for every address and name on the Internet.

If a DNS server can't locally resolve a query, it can "forward" the request to other servers in a process known as "recursive lookups". In fact, many DNS servers operate solely as "recursive DNS resolvers", taking requests from clients and then forwarding the request to other DNS servers for resolution. The server then returns the reply to the original client.

We'll see this process in action in this next exercise.

1. First, if the Packet Capture on the link between `br1-rtr` and `dc-rtr` is still running, stop it and clear the results.
2. We are going to capture traffic on the link between `dnsserver` and `dc-sw` in this lab. It will allow us to see the traffic from our host and the server as well as the recursive traffic from our server to public DNS servers for `cisco.com`.
   1. Right click the link and choose Packet Capture.

      > Hint: Place the capture in a split pane from the terminal for `br1-host1`

   2. Click the **SETTINGS** button.
   3. Filter for just DNS packets by adding the BPF filter `udp port 53` and clicking **APPLY**.
   4. Click **START** to begin capturing packets.
3. Open the console for `br1-host1` and perform a DNS lookup for `www.cisco.com`.

   ```
   dig u.cisco.com
   ```

   ```
   # Output
   ; <<>> DiG 9.18.33 <<>> u.cisco.com
   ;; global options: +cmd
   ```

```
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 64133
;; flags: qr rd ra; QUERY: 1, ANSWER: 5, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
; COOKIE: c8779f4f40fa767901000000679fd57db40ede4c6fdd42b1 (good)
;; QUESTION SECTION:
;u.cisco.com.                   IN      A

;; ANSWER SECTION:
u.cisco.com.            300     IN      CNAME   dgnr73mf6r0pg.cloudfront.net.
dgnr73mf6r0pg.cloudfront.net. 60 IN     A       108.138.167.22
dgnr73mf6r0pg.cloudfront.net. 60 IN     A       108.138.167.52
dgnr73mf6r0pg.cloudfront.net. 60 IN     A       108.138.167.53
dgnr73mf6r0pg.cloudfront.net. 60 IN     A       108.138.167.99

;; Query time: 100 msec
;; SERVER: 192.168.0.11#53(192.168.0.11) (UDP)
;; WHEN: Sun Feb 02 20:28:46 UTC 2025
;; MSG SIZE  rcvd: 174
```

Note: Your exact output may differ for the host name and IP addresses used. These can change overtime as public web applications are upgraded and maintained.

- Look at the `;; ANSWER SECTION:`. This should be familiar now, `u.cisco.com` is a `CNAME` to a `cloudfront.net` address. We see 4 different `A` records returned. Why do you think that multiple records are returned?

  ▶ Answer

4. Now look at the captured packets. The first packet should be the query from `br1-host1` to `dnsserver`. From there, `dnsserver` begins recursivly performing queries to provide an answer to the client.
   1. Look through the packets and see if you can find the "response" from the Umbrella DNS Server (`208.67.220.220` or `208.67.222.222`) that includes the CNAME and A record details that you saw on the host output.
   2. Find the return packet from `dnsserver` (`192.168.0.11`) to `br1-host1` (`192.168.11.11`).
   3. The other packets include traffic related to security and verification of the DNS protocol. The details of how this works is beyond CCNA level knowledge and this lab.
5. You can do additional lookups for other domains from the host. Just change the address in the `dig` command. You may need to **CLEAR** and **START** the capture if the buffer of 50 packets is reached.
6. When you are done exploring, stop the capture and close the window.

Exploring DNS's role in network applications (Email, Voice/Video)

So far we've seen `CNAME` and `A`/`AAAA` record types in our DNS queries. These are the most common records types and are critical for how network applications work, but there are other DNS record types that allow network based applications to work. Let us take a brief exploration deeper into DNS services.

DNS and Email

We'll begin with looking at how DNS factors into email services.

As a future CCNA, you are probably familar with *Simple Mail Transfer Protocol* (SMTP). SMTP servers receive and process email messages that are sent and read by users around the world. But how does a mail client know how to reach the SMTP server for an email message? Well, NAME -> ADDRESS is the role of DNS, and it indeed serves this purpose as well. The DNS record type `MX` stands for *mail exchanger* and is used here. Let's see it in action!

1. Open the console for `br1-host1`.
2. Lookup the `MX` record for `cisco.com`.

   ```
   dig cisco.com MX

   # Output (Just the ANSWER SECTIOn)
   ;; ANSWER SECTION:
   cisco.com.              863     IN      MX      20 rcdn-mx-01.cisco.com.
   cisco.com.              863     IN      MX      10 alln-mx-01.cisco.com.
   cisco.com.              863     IN      MX      30 aer-mx-01.cisco.com.
   ```

   - There are three different MX recordds for Cisco's email. The returned data is in the format `PRIORITY MAIL_SERVER_HOSTNAME`. Lower priority is preferred, so in this case the host `alln-mx-01.cisco.com` would be used.

     Note: The exact output you receive might differ from the output above if Cisco has made changes to their email systems.

3. But to what IP address will the email be sent? DNS provides the answer here as well.

   ```
   dig alln-mx-01.cisco.com

   # Output (Just the ANSWER SECTION)
   ;; ANSWER SECTION:
   alln-mx-01.cisco.com.   314     IN      A       173.37.147.230
   ```

4. Excellent! Now you can see the important role DNS plays in email systems. But the MX record isn't the only role DNS has.

Email spam prevention, security and DNS

Are you familiar with email "spam"? I'm sure you are... SPAM, and more importantly preventing SPAM is something that IT administrators have been working on since the start of email. There are many ways SPAM is prevented today, and some of them leverage DNS as a way to verify the "trusted" nature of email messages.

Two such technologies that rely on DNS are Sender Policy Framework (SPF) and Domain Keys Identified Mail (DKIM).

Let's lookup these settings for email from Cisco.com.

> Note: The examples in this section assume current settings for email services for Cisco. If settings change, the query results may no longer look exactly like this example.

1. Many "verifications" for applications that use DNS rely on the `TXT` record type. `TXT` records are just "text". Generally any text is allowed in a `TXT` record. This makes it easy to add new verifications to DNS without requiring changes in the DNS protocol.
2. Lookup all the `TXT` records for `cisco.com`

   ```
   dig cisco.com TXT
   ```

   - There will be MANY records returned. Scroll through them and see if you can figure out what some might be used for.
3. Rather than try to hunt through the list for the SPF record, let's filter down the results.

   ```
   dig cisco.com TXT | grep v=spf1

   # Output
   cisco.com.              815     IN      TXT     "v=spf1 redirect=spfa._spf.cisco.com"
   ```

   - There it is, the SPF record for cisco.com. The details and specifics of the record are beyond what we need to go into for CCNA. But now you know it's there :-)
4. To find the DKIM record you need to know a littlebit about how Cisco sends email. That is because the format of the DKIM record is `<selector>._domainkey.<domain.com>`. The `domain.com` part is just `cisco.com`. And `._domainkey` is just that string. But `selector` refers to a value from the `DKIM-Signature` header in the email message. Here is an example from a valid email from a Cisco.com address.

   ```
   DKIM-Signature: v=1; a=rsa-sha256; c=relaxed/relaxed;
   d=cisco.com; i=@cisco.com; l=3436; q=dns/txt; s=iport;
   t=1738537160; x=1739746760;
   ```

   - The important part is `s=iport`. The `selector` is `iport`.
5. Lookup the DKIM TXT record.

   ```
   dig iport._domainkey.cisco.com TXT

   # Output (The ANSWER SECTION)
   ;; ANSWER SECTION:
   iport._domainkey.cisco.com. 959 IN      TXT     "v=DKIM1; p=MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCctxGhJnvNpd
   ```

   - And there we have it, a DKIM record for cisco.com email. Like the SPF record, understanding the content of this record is beyond the CCNA knowledge expectations.

DNS and making Voice and Video Calls

Voice and video calls are pretty common today. If you've ever called someone using an address that looked a lot like an email address, DNS likely played a part in making that call work.

The standard communication protocol for both voice and video calls is the Session Initiation Protocol (SIP), and just like email servers require DNS to find the SMTP servers to send email to, SIP clients use DNS to find the appropriate communication gateways for addresses. Let's see what DNS can tell us about Cisco.com's SIP servers.

1. SIP uses another DNS record type, the `SRV` or "service record". This record type is used for more than just SIP, it can be used for any type of "service" or client based applications.
2. SIP SRV records can be found at `_sip._tcp.<domain>`.
3. Lookup the record for `cisco.com`.

   ```
   dig _sip._tcp.cisco.com SRV

   # Output (ANSWER SECTION)
   ;; ANSWER SECTION:
   _sip._tcp.cisco.com.    3525    IN      SRV     1 50 5060 vcsgw104.cisco.com.
   _sip._tcp.cisco.com.    3525    IN      SRV     1 50 5060 vcsgw103.cisco.com.
   _sip._tcp.cisco.com.    3525    IN      SRV     1 50 5060 vcsgw106.cisco.com.
   _sip._tcp.cisco.com.    3525    IN      SRV     1 50 5060 vcsgw101.cisco.com.
   _sip._tcp.cisco.com.    3525    IN      SRV     1 50 5060 vcsgw102.cisco.com.
   _sip._tcp.cisco.com.    3525    IN      SRV     1 50 5060 vcsgw105.cisco.com.
   ```

   - The data in the record provides the Priority (`1`), Weight (`50`), Port (`5060`), and Target (`vcsgw##.cisco.com`)
4. With this information, a client making a call to someone with an `@cisco.com` address knows it can contact one of the listed gateways on port `TCP/5060` to initiate the call.

DNS and Applications Summary

We've only scratched the surface of the important role that DNS plays in network based communications. But hopefully it will help you "explain the role" it plays when you see questions on your CCNA exam.

Configuring a Cisco IOS router as a DNS server

In this final part of our lab, we will look at how a Cisco IOS router can be configured to provide DNS services to a network. In production networks, a dedicated DNS server (either Linux or Windows based) is typically the right solution. However, a small branch location that lacks server resources can still benefit from local DNS resolution. Cisco routers have been called "integrated service routers", and DNS is one of the integrated services that can be provided.

1. Open the console for `br1-rtr` and enable the DNS server with this configuration command.

   ```
   ip dns server
   ```

2. Change the DHCP pool configuration on `br1-rtr` to tell hosts to use itself as the DNS server.

   ```
   ip dhcp pool branch1
   no dns-server 192.168.0.11
   dns-server 192.168.11.1
   ```

3. Open the console for `br1-host1` and request a refresh to the DHCP lease that has been assigned.

   ```
   sudo udhcpc
   ```

   ```
   #Output
   udhcpc: started, v1.36.1
   udhcpc: broadcasting discover
   udhcpc: broadcasting select for 192.168.11.11, server 192.168.11.1
   udhcpc: lease of 192.168.11.11 obtained from 192.168.11.1, lease time 86400
   ```

4. Verify the DNS server has been updated.

   ```
   cat /etc/resolv.conf
   ```

   ```
   # Output
   search example.com
   nameserver 192.168.11.1
   ```

5. Do a DNS lookup.

   ```
   dig app.example.com
   ```

   ```
   # Output (The SERVER)
   ;; SERVER: 192.168.11.1#53(192.168.11.1) (UDP)
   ```

   - Notice that the server that was queried and replied is now the router

6. Setup some packet captures and follow the packets for query/response flows for different domains.

Creating DNS host records on Cisco IOS XE

The DNS server in IOS XE isn't as full featured a dedicated DNS servers like Bind or Windows DNS Server, but some basic records can be created including `A`, `AAAA`, `MX`, and `SRV`.

1. Let's see this in action by creating an `A` record for `localrecord.example.com` on the router.

2. Open the console for `br1-rtr` and add this configuration.

   ```
   ip host localrecord.example.com 10.0.0.11
   ```

   > Note: `10.0.0.11` is the Loopback0 address on the router

3. Now try to lookup this record from `br1-host1`.

   ```
   dig localrecord.example.com
   ```

   - Did you get the expected reply from the router?

Great Job!

Excellent work on this lab! You've seen DNS in action, explored its role as a foundation in everyday application and Internet use, and even saw how you can configure a basic DNS server using Cisco IOS.