Troubleshooting Application Connectivity Issues

5.6.1

Troubleshooting Common Network Connectivity Issues

There can be many reasons why network connectivity is not functioning the way you might expect. Whenever an application or a website or any target destination that is being accessed over a network is not behaving as expected, the network takes the blame. While the network might be the culprit in some instances, there are many other reasons why applications stop responding as expected.

Network troubleshooting usually follows the OSI layers. You can start either top to bottom beginning at the application layer and making your way down to the physical layer. Or you can go from the bottom to the top. In this example, we will cover a typical troubleshooting session starting from physical layer and making our way up the stack of layers towards the application layer.

First and foremost, from a client perspective, it is very important to determine how the client connects to the network. Is it a wired or wireless connection?

If the client connects via an Ethernet cable, make sure the NIC comes online and there are electrical signals being exchanged with the switch port to which the cable is connected. Depending on the operating system that the client is running, the status of the network connection will show as a solid green, or it will display "enabled" or "connected" text next to the network interface card in the operating system settings. If the NIC shows as connected, you know the physical layer is working as expected and can move the troubleshooting up the stack. If the NIC on the client does not show up as connected or enabled, then check the configuration on the switch. The port to which the client is connecting might be shut down, or maybe the cable connecting the client to the network port in the wall is defective, or the cable connecting the network port from the wall all the way to the switch might be defective. Troubleshooting at the physical layer basically boils down to making sure there are four uninterrupted pairs of twisted copper cables between the network client and the switch port.

If the client wirelessly connects to the network, make sure that the wireless network interface is turned on and it can send and receive wireless signals to and from the nearest wireless access point. Also, make sure you stay in the range of the wireless access point as long as you need to be connected to the network.

Moving up to the data link layer, or Layer 2, make sure the client is able to learn destination MAC addresses (using ARP) and also that the switch to which the client is connecting is able to learn the MAC

addresses received in its ports. On most operating systems you can view the ARP table with the a form of the **arp** command (such as `arp -a` on a Windows 10 PC) or on Cisco switches you can verify the MAC address table with the command `show mac address-table`. If you can verify that the both these tables are accurate, then you can move to the next layer. If the client cannot see any MAC addresses in its local ARP table, check for any Layer 2 access control lists on the switch port that might block this traffic. Also make sure that the switch port is configured for the correct client VLAN.

At the network layer, or Layer 3, make sure the client obtains the correct IP address from the DHCP server, or is manually configured with the correct IP address and the correct default gateway. If the destination of your traffic is in a different subnet than the subnet you are connected to, that means the traffic will have to be sent to the local router (default gateway). Check Layer 3 connectivity one hop at a time. First check connectivity to the first Layer 3 hop in the path of the traffic, which is the default gateway, to make sure you can reach it. If Layer 3 connectivity can be established all the way from the client to the destination, move on with troubleshooting to the transport layer, or Layer 4. If Layer 3 connectivity cannot be established, check IP access lists on the router interfaces, check the routing table on both the client and the default gateway router and make sure the traffic is routed correctly. Routing protocol issues and access control lists blocking IP traffic are some of the usual problems encountered at this layer.

You have verified end to end communication between the source and destination of the traffic in the previous step. This is a major milestone, so give yourself a pat on the back. You almost cleared the network team from the responsibility of that application outage. Before blaming the application support team there is one more thing to verify. Make sure the client can access the port on which the application is running. If the destination of the traffic is a webpage served by a web server, make sure TCP ports 80 (HTTP) and 443 (HTTPS) are accessible and reachable from the client. It could be that the web server is running on an esoteric port like 8080, so make sure you know the correct port the application you are trying to connect to is running. Networking tools like curl or a custom telnet command specifying the application port can be used to ensure the transport layer works end-to-end between the source and destination of the traffic. If a transport connection cannot be established, verify firewalls and security appliances that are placed on the path of traffic for rules that are blocking the traffic based on TCP and UDP ports. Verify if any load balancing is enabled and if the load balancer is working as expected, or if any proxy servers intercepting the traffic are filtering and denying the connection.

So you got this far, checked end-to-end connectivity, you can connect to the port on which the application is running, so the network team is in the clear, right? Almost. One additional thing to verify is traffic load and network delay. Networking tools like **iperf** can generate traffic and load stress the network to ensure that large amounts of data can be transported between the source and destination. These issues are the most difficult to troubleshoot because they can be difficult to reproduce. They can be temporarily caused by a spike in network traffic, or could be outside your control all together. Implementing QoS throughout the network can help with these issues. With QoS, traffic is categorized into different buckets and each bucket gets separate treatment from the network. For example, real time traffic, like voice and video can be classified as such by changing QoS fields in the Layer 2 and Layer 3 packet headers so that when switches and routers process this type of traffic, it gets a higher priority and guaranteed bandwidth if necessary.

Or, maybe you are lucky to begin with and are verifying a web server access or a REST API endpoint and the server returns a 500 status code. In that case you can start troubleshooting the web server and skip all of the network troubleshooting steps.

If you got this far in your network troubleshooting, there is a good chance that the problem is not with the network and a closer look at the application server is in order. Slow or no responses from the application could also mean an overloaded backend database, or just faulty code introduced through new features. In this case, solutions like Cisco AppDynamics can offer a deeper view into application performance and root cause analysis of application issues.

5.6.2

Networking Tools - Using ifconfig

`ifconfig` is a software utility for UNIX-based operating systems. There is also a similar utility for Microsoft Windows-based operating systems called `ipconfig`. The main purpose of this utility is to manage, configure, and monitor network interfaces and their parameters. `ifconfig` runs as a command-line interface tool and comes by default installed with most operating systems.

Common uses for `ifconfig` are the following:

- Configure IP address and subnet mask for network interfaces.
- Query the status of network interfaces.
- Enable/disable network interfaces.
- Change the MAC address on an Ethernet network interface.

Issuing the `ifconfig --help` command in the command line interface will display all the options that are available with this version of `ifconfig`. The output should look similar to the following.

```
devasc@labvm:~$ ifconfig --help
Usage:
  ifconfig [-a] [-v] [-s] <interface> [[<AF>] <address>]
  [add <address>[/<prefixlen>]]
  [del <address>[/<prefixlen>]]
  [[-]broadcast [<address>]]  [[-]pointopoint [<address>]]
  [netmask <address>]  [dstaddr <address>]  [tunnel <address>]
  [outfill <NN>] [keepalive <NN>]
  [hw <HW> <address>]  [mtu <NN>]
  [[-]trailers]  [[-]arp]  [[-]allmulti]
  [multicast]  [[-]promisc]
  [mem_start <NN>]  [io_addr <NN>]  [irq <NN>]  [media <type>]
  [txqueuelen <NN>]
  [[-]dynamic]
  [up|down] ...
<output omitted>
```

From this output we can see that `ifconfig` gives us the option to `add` (add) or `del` (delete) IP addresses and their subnet mask (prefix length) to a specific network interface. The `hw ether` gives us the option to change the Ethernet MAC address. Care should be taken especially when shutting down interfaces, because, if you are remotely connected to that host via the interface you are shutting down, you have just disconnected your session and possibly have to physically walk up to the device to bring it back online. That is not such a big problem when the device is in the room next door, but it can be quite daunting when it is physically in a data center hundreds of miles away and you have to drive for hours or even take a flight to bring it back online.

If `ifconfig` is issued without any parameters, it just returns the status of all the network interfaces on that host. For your DEVASC VM, the output should look similar to the following:

```
devasc@labvm:~$ ifconfig
dummy0: flags=195<UP,BROADCAST,RUNNING,NOARP>  mtu 1500
        inet 192.0.2.1  netmask 255.255.255.255  broadcast 0.0.0.0
        inet6 fe80::48db:6aff:fe27:4849  prefixlen 64  scopeid 0x20<link>
        ether 4a:db:6a:27:48:49  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 12293  bytes 2544763 (2.5 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::a00:27ff:fee9:3de6  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:e9:3d:e6  txqueuelen 1000  (Ethernet)
        RX packets 280055  bytes 281957761 (281.9 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 112889  bytes 10175993 (10.1 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 46014  bytes 14094803 (14.0 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 46014  bytes 14094803 (14.0 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

devasc@labvm:~$
```

`MTU` is the Maximum Transmission Unit and specifies the maximum number of bytes that the frame can be transmitted on this medium before being fragmented.

The `RX packets` and `RX bytes` contain the values of the received packets and bytes respectively on that interface. In the example above there were 280055 packets received on the `enp0s3` interface that

contained 281.9 MB of data. The `TX packets` and `TX bytes` contain the values of the transmit packets and bytes on that specific interface. In the example, there were 112889 packets transmitted on `enp0s3` which accounted for 10.1 MB of data.

`ifconfig` is still used extensively for both configuration and monitoring purposes. There are also GUI clients for most operating systems that take this functionality into a graphical interface and make it more visual for end users to configure network interfaces.

Note: The `ifconfig` command has been used within Linux for many years. However, some Linux distributions have deprecated the `ifconfig` command. The `ip address` command is becoming the new alternative. You will see the `ip address` command used in some of the labs in this course.

5.6.3

Using ping

Similar to `ifconfig`, `ping` is a software utility used to test IP network reachability for hosts and devices connected to a specific network. It is also available on virtually all operating systems and is extremely useful for troubleshooting connectivity issues. The `ping` utility uses Internet Control Message Protocol (ICMP) to send packets to the target host and then waits for ICMP echo replies. Based on this exchange of ICMP packets, `ping` reports errors, packet loss, roundtrip time, time to live (TTL) for received packets, and more.

On Windows 10, enter the `ping` command to view its usage information. The output should look similar to the following:

```
C:\> ping

Usage: ping [-t] [-a] [-n count] [-l size] [-f] [-i TTL] [-v TOS]
            [-r count] [-s count] [[-j host-list] | [-k host-list]]
            [-w timeout] [-R] [-S srcaddr] [-c compartment] [-p]
            [-4] [-6] target_name

Options:
    -t             Ping the specified host until stopped.
                   To see statistics and continue - type Control-Break;
                   To stop - type Control-C.
    -a             Resolve addresses to hostnames.
    -n count       Number of echo requests to send.
    -l size        Send buffer size.
    -f             Set Don't Fragment flag in packet (IPv4-only).
    -i TTL         Time To Live.
    -v TOS         Type Of Service (IPv4-only. This setting has been deprecated
```

```
                    and has no effect on the type of service field in the IP
                    Header).
    -r count        Record route for count hops (IPv4-only).
    -s count        Timestamp for count hops (IPv4-only).
    -j host-list    Loose source route along host-list (IPv4-only).
    -k host-list    Strict source route along host-list (IPv4-only).
    -w timeout      Timeout in milliseconds to wait for each reply.
    -R              Use routing header to test reverse route also (IPv6-only).
                    Per RFC 5095 the use of this routing header has been
                    deprecated. Some systems may drop echo requests if
                    this header is used.
    -S srcaddr      Source address to use.
    -c compartment  Routing compartment identifier.
    -p              Ping a Hyper-V Network Virtualization provider address.
    -4              Force using IPv4.
    -6              Force using IPv6.


C:\>
```

On MacOS Catalina, enter the `ping` command to view its usage information. The output should look similar to the following:

```
$ ping
usage: ping [-AaDdfnoQqRrv] [-c count] [-G sweepmaxsize]
            [-g sweepminsize] [-h sweepincrsize] [-i wait]
            [-l preload] [-M mask | time] [-m ttl] [-p pattern]
            [-S src_addr] [-s packetsize] [-t timeout][-W waittime]
            [-z tos] host
       ping [-AaDdfLnoQqRrv] [-c count] [-I iface] [-i wait]
            [-l preload] [-M mask | time] [-m ttl] [-p pattern] [-S src_addr]
            [-s packetsize] [-T ttl] [-t timeout] [-W waittime]
            [-z tos] mcast-group
Apple specific options (to be specified before mcast-group or host like all options)
            -b boundif           # bind the socket to the interface
            -k traffic_class     # set traffic class socket option
            -K net_service_type  # set traffic class socket options
            -apple-connect       # call connect(2) in the socket
            -apple-time          # display current time
```

On your DEVASC VM, add the `-help` option to view its usage information. The output should look similar to the following:

```
devasc@labvm:~$ ping -help

Usage
  ping [options] <destination>
```

```
Options:
  <destination>      dns name or ip address
  -a                 use audible ping
  -A                 use adaptive ping
  -B                 sticky source address
  -c <count>         stop after <count> replies
  -D                 print timestamps
  -d                 use SO_DEBUG socket option
  -f                 flood ping
  -h                 print help and exit
<output omitted>

IPv4 options:
  -4                 use IPv4
  -b                 allow pinging broadcast
  -R                 record route
  -T <timestamp>     define timestamp, can be one of <tsonly|tsandaddr|tsprespec>

IPv6 options:
  -6                 use IPv6
  -F <flowlabel>     define flow label, default is random
  -N <nodeinfo opt>  use icmp6 node info query, try <help> as argument

For more details see ping(8).
devasc@labvm:~$
```

By default, `ping` (or `ping -help` in Linux) will display all the options it has available. Some of the options you can specify include:

- Count of how many ICMP echo requests you want to send
- Source IP address in case there are multiple network interfaces on the host
- Timeout to wait for an echo reply packet
- Packet size, if you want to send larger packet sizes than the default 64 bytes. This option is very important when determining what is the MTU on an interface.

For example, enter the command `ping -c 5 ww&#8203;w.cisco.com` in your DEVASC VM terminal window to see if the web server is reachable from your PC and responding to ICMP echo-requests. The `-c` option allows you to specify that only 5 ping packets should be sent.

```
devasc@labvm:~$ ping -c 5 www.cisco.com
PING e2867.dsca.akamaiedge.net (23.204.11.200) 56(84) bytes of data.
64 bytes from a23-204-11-200.deploy.static.akamaitechnologies.com (23.204.11.200):
icmp_seq=1 ttl=57 time=81.4 ms
64 bytes from a23-204-11-200.deploy.static.akamaitechnologies.com (23.204.11.200):
icmp_seq=2 ttl=57 time=28.5 ms
64 bytes from a23-204-11-200.deploy.static.akamaitechnologies.com (23.204.11.200):
```

```
icmp_seq=3 ttl=57 time=31.5 ms
64 bytes from a23-204-11-200.deploy.static.akamaitechnologies.com (23.204.11.200):
icmp_seq=4 ttl=57 time=28.8 ms
64 bytes from a23-204-11-200.deploy.static.akamaitechnologies.com (23.204.11.200):
icmp_seq=5 ttl=57 time=26.5 ms

--- e2867.dsca.akamaiedge.net ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4024ms
rtt min/avg/max/mdev = 26.481/39.335/81.372/21.078 ms
devasc@labvm:~$
```

The output of the command in your environment will most probably look a bit different but the major components should be the same. We specified a count of 5 ICMP echo request packets to be sent to ww w.cisco.com. The ping utility automatically does the DNS resolution and in this case it resolved the ww w.cisco.com name to the 23.204.11.200 IPv4 address. The packets are sent, and responses are received from 23.204.11.200. TTL for the received echo replies and round trip times are calculated and displayed. The final statistics, confirm that 5 ICMP echo-request packets have been transmitted and 5 ICMP echo-reply packets have been received, hence there is a 0.0% packet loss. Statistics about the minimum, average, maximum and standard deviation of the time it took for the packets to get to the destination and back are also displayed.

Keep in mind that if you do not receive any replies from the destination you are trying to reach with `ping` it does not mean that the host is offline or not reachable. It could simply mean that ICMP echo-request packets are filtered by a firewall and are not allowed to reach the destination host. It is actually a best practice to expose only the services needed to be available on the hosts in the network. For example, a web server would only expose TCP port 443 for secure HTTP traffic and deny any other types of traffic either through a local firewall on the web server itself or a network firewall.

For IPv6 there exists a similar utility on Linux and MacOS that is called `ping6` and is also available on most operating systems. Windows and Cisco IOS uses the same `ping` command for both IPv4 and IPv6.

5.6.4

Using traceroute

You have seen how `ping` can display host reachability on the network. `traceroute` builds on top of that functionality and displays the route that the packets take on their way to the destination. The Microsoft Windows alternative is also a command-line utility and is called `tracert`. Observing the path the network traffic takes from its source to the destination is extremely important from a troubleshooting perspective, as routing loops and non-optimal paths can be detected and then remedied.

`traceroute` uses ICMP packets to determine the path to the destination. The Time to Live (TTL) field in the IP packet header is used primarily to avoid infinite loops in the network. For each hop or router that an IP packet goes through, the TTL field is decremented by one. When the TTL field value reaches 0, the

packet is discarded, avoiding the dreaded infinite loops. Usually, the TTL field is set to its maximum value, 255, on the host that is the source of the traffic, as the host is trying to maximize the chances of that packet getting to its destination. `traceroute` reverses this logic, and gradually increments the TTL value of the packet it is sending, from 1 and keeps adding 1 to the TTL field on the next packet and so on. Setting a TTL value of 1 for the first packet, means the packet will be discarded on the first router. By default, most routers send back to the source of the traffic an ICMP Time Exceeded packet informing it that the packet has reached a TTL value of 0 and had to be discarded. `traceroute` uses the information received from the router to figure out its IP address and hostname and also round trip times.

Note: Instead of ICMP, by default, Linux uses UDP and a high port range (33434 – 33534). Destinations along the path respond with ICMP port unreachable messages instead of the echo replies sent in ICMP-based traceroutes.

On Windows 10, use `tracert` to see the available options as shown in the following output:

```
C:\> tracert

Usage: tracert [-d] [-h maximum_hops] [-j host-list] [-w timeout]
               [-R] [-S srcaddr] [-4] [-6] target_name

Options:
    -d                 Do not resolve addresses to hostnames.
    -h maximum_hops    Maximum number of hops to search for target.
    -j host-list       Loose source route along host-list (IPv4-only).
    -w timeout         Wait timeout milliseconds for each reply.
    -R                 Trace round-trip path (IPv6-only).
    -S srcaddr         Source address to use (IPv6-only).
    -4                 Force using IPv4.
    -6                 Force using IPv6.

C:\>
```

On MacOS, use `traceroute` to see the available options as shown in the following output:

```
$ traceroute
Version 1.4a12+Darwin
Usage: traceroute [-adDeFInrSvx] [-A as_server] [-f first_ttl] [-g gateway] [-i iface]
    [-M first_ttl] [-m max_ttl] [-p port] [-P proto] [-q nqueries] [-s src_addr]
    [-t tos] [-w waittime] [-z pausemsecs] host [packetlen]
```

On your DEVASC VM, use `traceroute --help` to see the available options as shown in the following output:

```
devasc@labvm:~$ traceroute --help
Usage: traceroute [OPTION...] HOST
Print the route packets trace to network host.
```

```
  -f, --first-hop=NUM        set initial hop distance, i.e., time-to-live
  -g, --gateways=GATES       list of gateways for loose source routing
  -I, --icmp                 use ICMP ECHO as probe
  -m, --max-hop=NUM          set maximal hop count (default: 64)
  -M, --type=METHOD          use METHOD (`icmp' or `udp') for traceroute
                             operations, defaulting to `udp'
  -p, --port=PORT            use destination PORT port (default: 33434)
  -q, --tries=NUM            send NUM probe packets per hop (default: 3)
      --resolve-hostnames    resolve hostnames
  -t, --tos=NUM              set type of service (TOS) to NUM
  -w, --wait=NUM             wait NUM seconds for response (default: 3)
  -?, --help                 give this help list
      --usage                give a short usage message
  -V, --version              print program version

Mandatory or optional arguments to long options are also mandatory or optional
for any corresponding short options.

Report bugs to <bug-inetutils@gnu.org>.
devasc@labvm:~$]
```

Several options are available with `traceroute` including:

- Specifying the TTL value of the first packet sent. By default this is 1.
- Specifying the maximum TTL value. By default, it will increase the TTL value up to 64 or until the destination is reached.
- Specifying the source address in case there are multiple interfaces on the host.
- Specifying QoS value in the IP header.
- Specifying the packet length.

Because of the way Virtual Box implements a NAT network, you cannot trace outside of your DEVASC VM. You would need to change your VM to Bridged mode. But then, you would not be able to communicate with the CSR1000v in other labs. Therefore, we recommend leaving your VM in NAT mode.

However, you can `tracert` from your Windows device or `traceroute` from your MacOS device. The following output is from a MacOS device inside the corporate Cisco network tracing the route to one of Yahoo's web servers.

```
$ traceroute www.yahoo.com
traceroute: Warning: www.yahoo.com has multiple addresses; using 98.138.219.232
traceroute to atsv2-fp-shed.wg1.b.yahoo.com (98.138.219.232), 64 hops max, 52 byte
packets
 1  sjc2x-dtbb.cisco.com (10.1x.y.z)  2.422 ms  1.916 ms  1.773 ms
 2  sjc2x-dt5.cisco.com (12x.1y.1z.1ww)  2.045 ms
    sjc2x-dt5-01.cisco.com (12x.1y.1z.15w)  2.099 ms  1.968 ms
 3  sjc2x-sbb5.cisco.com (1xx.1x.1xx.4y)  1.713 ms  1.984 ms
```

```
    sjc2x-sbb5-10.cisco.com (1xx.1x.1y.4w)  1.665 ms
 4  sjc2x-rbb.cisco.com (1xx.1y.zz.yyy)  1.836 ms  1.804 ms  1.696 ms
 5  sjc1x-rbb-7.cisco.com (1xx.zz.y.ww)  68.448 ms  1.880 ms  1.939 ms
 6  sjc1x-corp-0.cisco.com (1xx.yy.z.w)  1.890 ms  2.660 ms  2.793 ms
 7  * * *
 8  * * *
 9  * * *
...
61  * * *
62  * * *
63  * * *
64  * * *
```

Note: The output above has been altered for security reasons, but your output should actually have both valid hostnames and IP addresses.

From this output, we can see the first 6 hops or routers that are on the path towards www.yahoo.com. The entries for 2 and 3 have two values, suggesting there is load balancing implemented on this specific path. Round trip times are also included in the output. In this output you can also see that the `traceroute` traffic is not allowed outside of the corporate Cisco network, so the complete path to the destination is not available. By filtering ICMP Time Exceeded messages with firewalls or just disabling it at the host and router level, visibility of the path with traceroute is greatly limited. Still, even with these limitations, `traceroute` is an extremely useful utility to have in your tool belt for troubleshooting network-related issues.

For IPv6 there is an alternative called `traceroute6` for UNIX-based operating systems. `tracert` is used for both IPv4 and IPv6 on Windows, however you can use the -4 or -6 parameter to select the Layer 3 protocol.

5.6.5

Using nslookup

`nslookup` is another command-line utility used for querying DNS to obtain domain name to IP address mapping. Like other tools mentioned in this section, `nslookup` is widely available on most all operating systems. This tool is useful to determine if the DNS server configured on a specific host is working as expected and actually resolving hostnames to IP addresses. It could be that maybe a DNS server is not configured at all on the host, so make sure you check `/etc/resolv.conf` on UNIX-like operating systems and that you have at least a `nameserver` defined.

The DEVASC VM Linux OS does not implement a help option for the `nslookup` command. However, you can enter `man nslookup` to learn more about the available options.

In the terminal, execute the command `nslookup www.cisco.com 8.8.8.8` to resolve the IP address or addresses for Cisco's web server and specify that you want to use Google's DNS server at 8.8.8.8 to do the resolution.

Note: Dig is often the preferred tool to use to query DNS.

```
devasc@labvm:~$ nslookup www.cisco.com 8.8.8.8
Server:         8.8.8.8
Address:        8.8.8.8#53

Non-authoritative answer:
www.cisco.com   canonical name = www.cisco.com.akadns.net.
www.cisco.com.akadns.net        canonical name = wwwds.cisco.com.edgekey.net.
wwwds.cisco.com.edgekey.net     canonical name =
wwwds.cisco.com.edgekey.net.globalredir.akadns.net.
wwwds.cisco.com.edgekey.net.globalredir.akadns.net      canonical name =
e2867.dsca.akamaiedge.net.
Name:   e2867.dsca.akamaiedge.net
Address: 23.204.11.200
Name:   e2867.dsca.akamaiedge.net
Address: 2600:1404:5800:392::b33
Name:   e2867.dsca.akamaiedge.net
Address: 2600:1404:5800:39a::b33

devasc@labvm:~$
```

The DNS service running on server 8.8.8.8 resolved the www.cisco.com domain to 3 IP addresses as you can see above. This resolution from names to IP addresses is critically important to the functioning of any network. It is much easier to remember www.cisco.com than an IPv4 or IPv6 address every time you are trying to access the Cisco website.

5.6.6

Packet Tracer – Troubleshoot Common Network Problems

Networks have a lot of components working together to ensure connectivity and data delivery. Often, these components may not work properly. This may be due to a simple device misconfiguration, or many, seemingly unrelated problems that must be systematically resolved. As a developer, you may need to troubleshoot network issues to regain connectivity. To troubleshoot network issues, it is necessary to take a step-by-step methodical approach, using clues to determine the problem and implement a solution. You may often find more than one problem preventing a connection from working.

You will complete these objectives:

- Part 1: Test connectivity
- Part 2: Troubleshoot R3
- Part 3: Troubleshoot R1
- Part 4: Troubleshoot DNS

Troubleshoot Common Network Problems

Troubleshoot Common Network Problems

5.6.7

Lab - Network Troubleshooting Tools

In the effort to fix network connection issues, it is important for a developer to understand how to use basic network troubleshooting tools. These tools are used to determine what the connection problem might be.

You will complete these objectives:

- Part 1: Launch the DEVASC VM
- Part 2: Explore the `ifconfig` Troubleshooting Tool
- Part 3: Explore the `ping` Troubleshooting Tool
- Part 4: Explore the `traceroute` Troubleshooting Tool
- Part 5: Explore the `nslookup` Troubleshooting Tool

Network Troubleshooting Tools

5.5
Networking Protocols

5.7
Networking Fundamentals Summary