CSCI 476: Computer Security

Network Security: DNS Cache Poisoning

Reese Pearsall

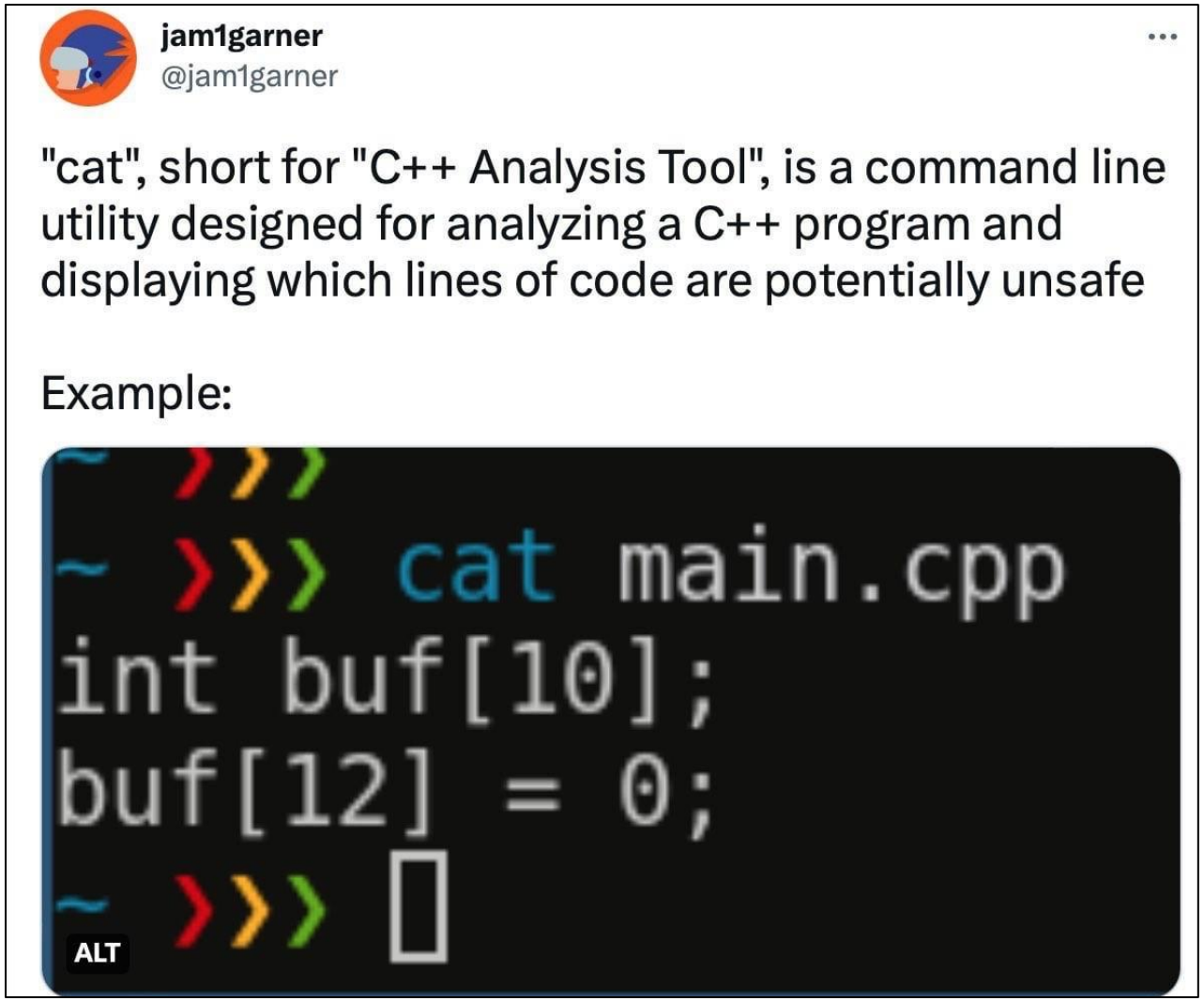
Spring 2023

Announcement

Guest Speaker Hill AFB

Lab 6 (TCP/IP Attacks) Due Sunday **4/2** @ 11:59 PM

Friday's lecture will be a help session for Lab 6



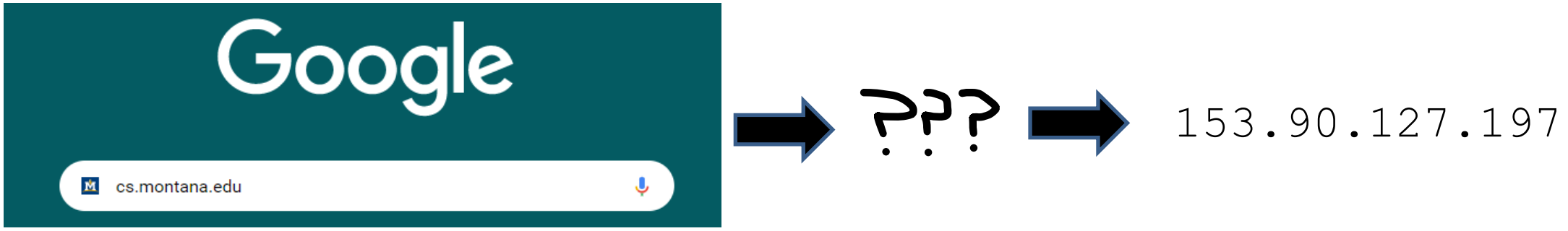
When browsing the web, computers need the **IP address** of the host we are communicating with

Humans do not use IP addresses when using the internet, they use hostnames (English)

We need a way to go from **hostnames** to **IP addresses**

Humans browse the web using hostnames
- (They need English)

Computers understand numbers
- (They need IP addresses)



When browsing the web, computers need the **IP address** of the host we are communicating with

Humans do not use IP addresses when using the internet, they use hostnames (English)

We need a way to go from **hostnames** to **IP addresses**

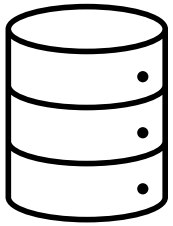
Humans browse the web using hostnames • (They need English)

Computers understand numbers • (They need IP addresses)

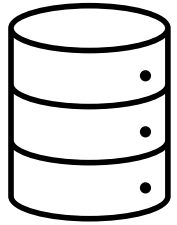
DNS

`153.90.127.197`

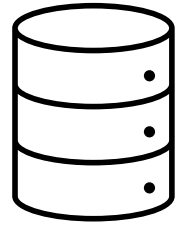
DNS Architecture



montana.edu



youtube.com



irs.gov

Clients will first need to resolve hostnames with DNS

3.

1.

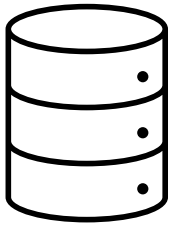
montana.edu ?

153.90.3.95

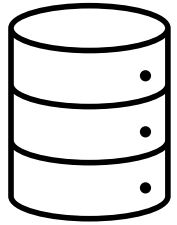
2.

Clients

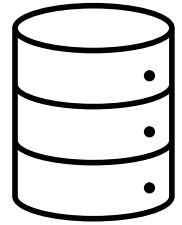
DNS Architecture



montana.edu



youtube.com



irs.gov

.com

.gov

.edu

Clients

DNS Architecture

(how big would that map be?)

- DNS is a **distributed**, **hierarchical** database (no DNS server has all the records!)

Hierarchy consists of different types of DNS servers:

DNS Architecture

(how big would that map be?)

- DNS is a **distributed**, **hierarchical** database (no DNS server has all the records!)

Hierarchy consists of different types of DNS servers:

Authoritative DNS servers- Organization's own DNS with up-to-date records

facebook.com DNS

amazon.com DNS

montana.edu DNS

harvard.edu DNS

DNS Architecture

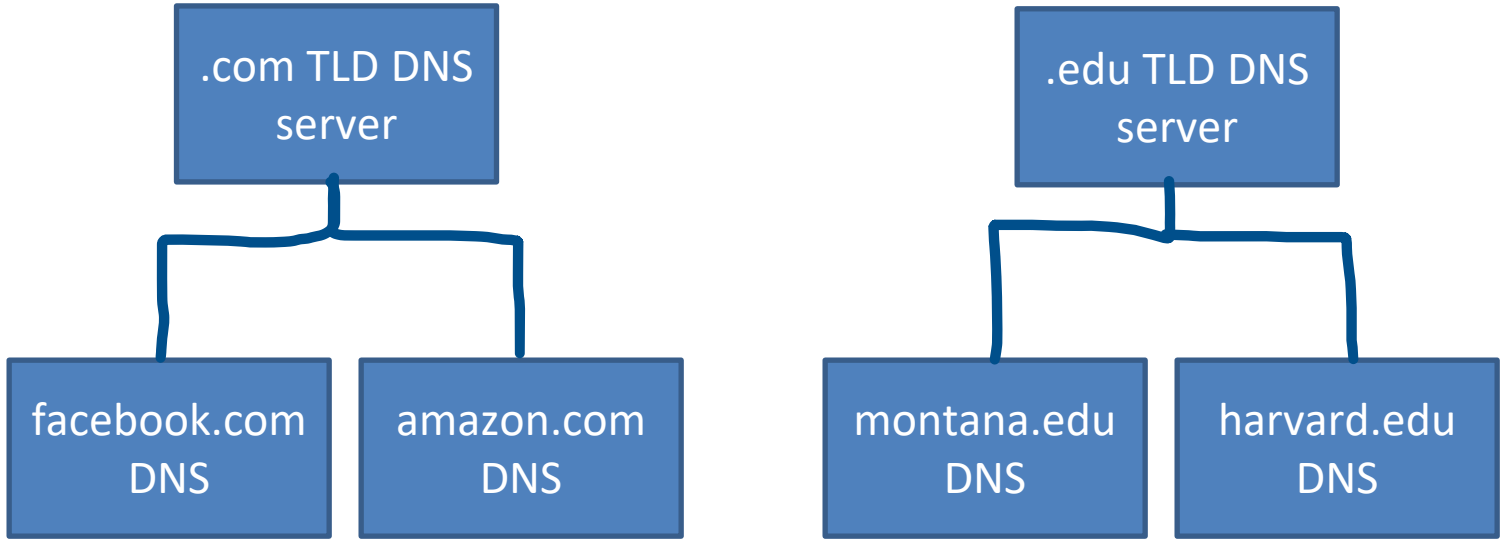
(how big would that map be?)

- DNS is a **distributed**, **hierarchical** database (no DNS server has all the records!)

Hierarchy consists of different types of DNS servers:

Authoritative DNS servers- Organization's own DNS with up-to-date records

Top-level domain (TLD) servers- responsible for keeping IP addresses for authoritative DNS servers for each top-level domain (.com, .edu, .jp, etc)



DNS Architecture

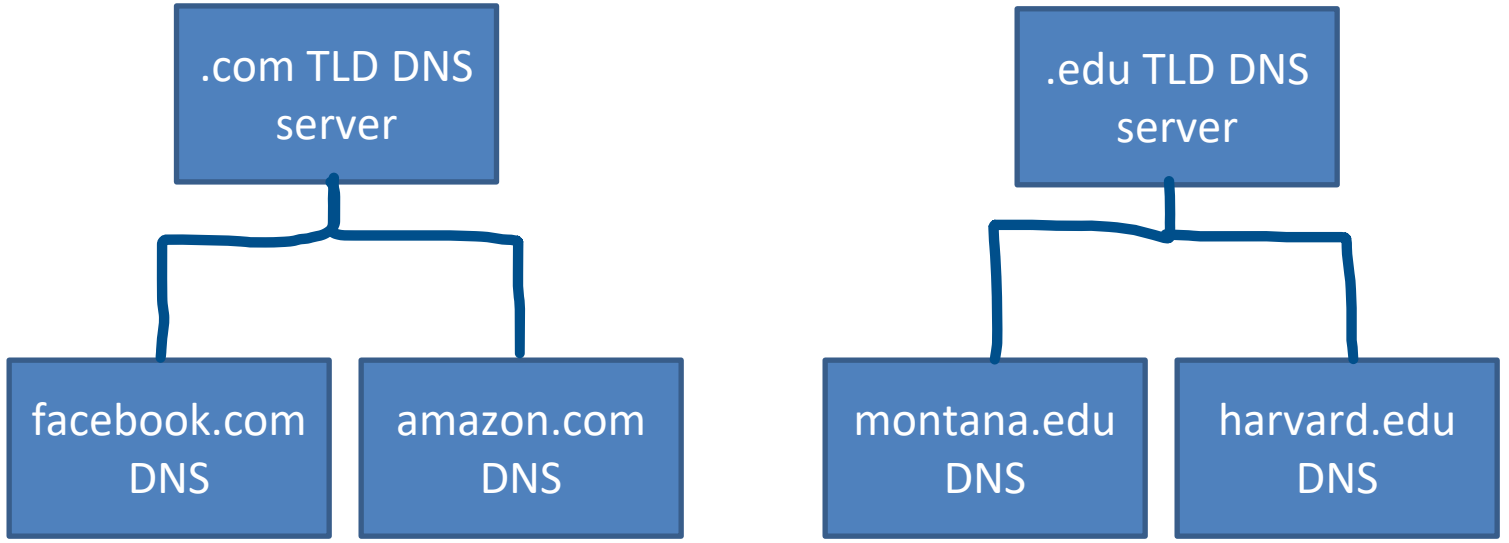
(how big would that map be?)

- DNS is a **distributed**, **hierarchical** database (no DNS server has all the records!)

Hierarchy consists of different types of DNS servers:

Authoritative DNS servers- Organization's own DNS with up-to-date records

Top-level domain (TLD) servers- responsible for keeping IP addresses for authoritative DNS servers for each top-level domain (.com, .edu, .jp, etc)



DNS Architecture

(how big would that map be?)

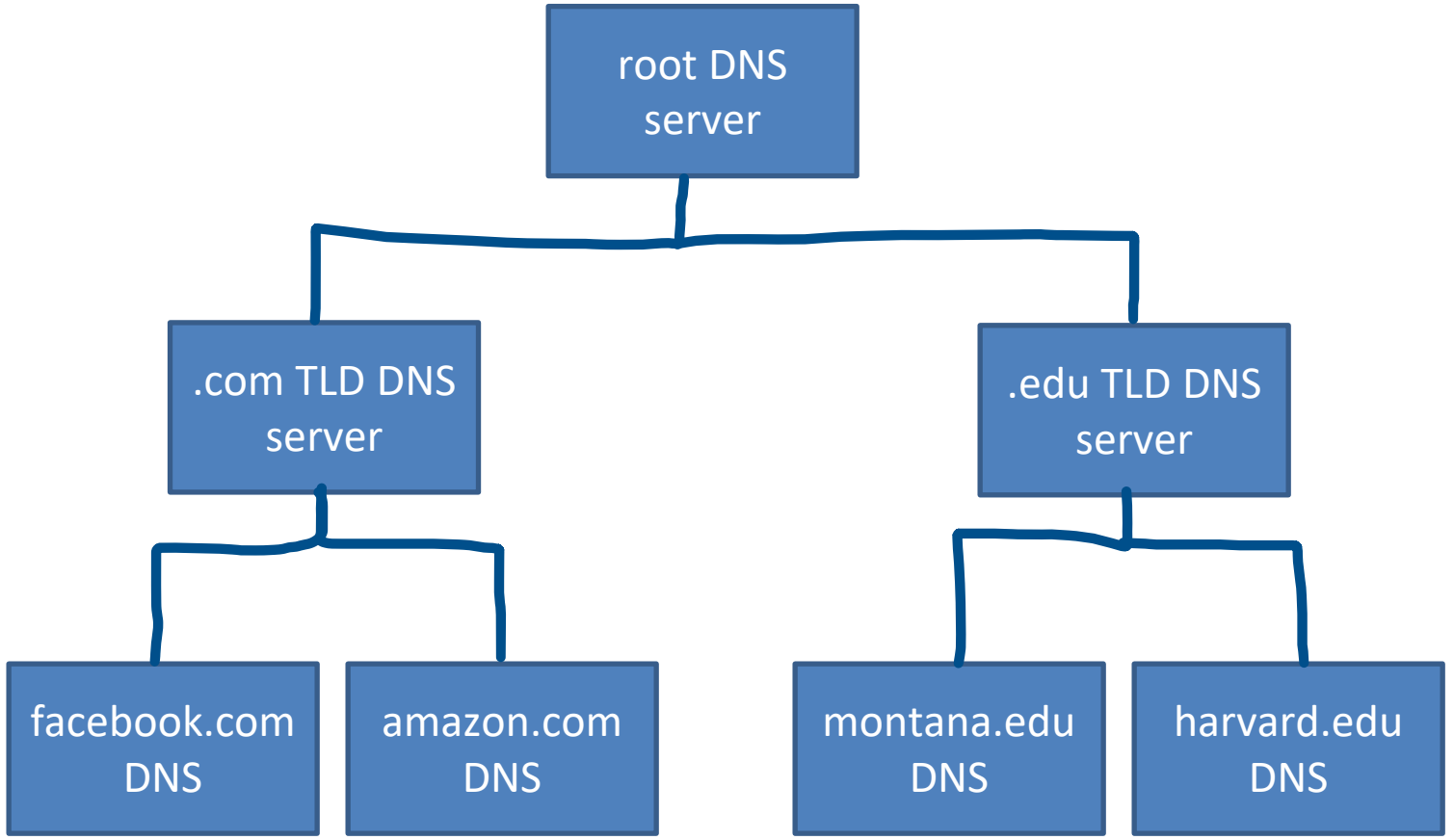
- DNS is a **distributed**, **hierarchical** database (no DNS server has all the records!)

Hierarchy consists of different types of DNS servers:

Authoritative DNS servers- Organization's own DNS with up-to-date records

Top-level domain (TLD) servers- responsible for keeping IP addresses for authoritative DNS servers for each top-level domain (.com, .edu, .jp, etc)

Root DNS servers- responsible for maintaining IP addresses for TLD servers



DNS Root server locations

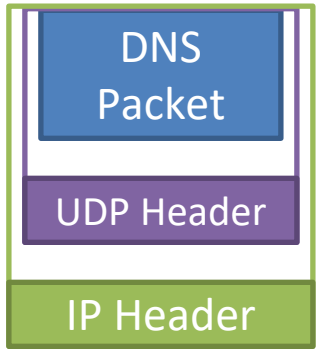
https://root-servers.org/

Domain Name System (DNS)

Application-level protocol used to map Domain Names to IP Addresses

DNS uses UDP as the transport layer protocol

- No handshake
- No guarantee that packet will arrive



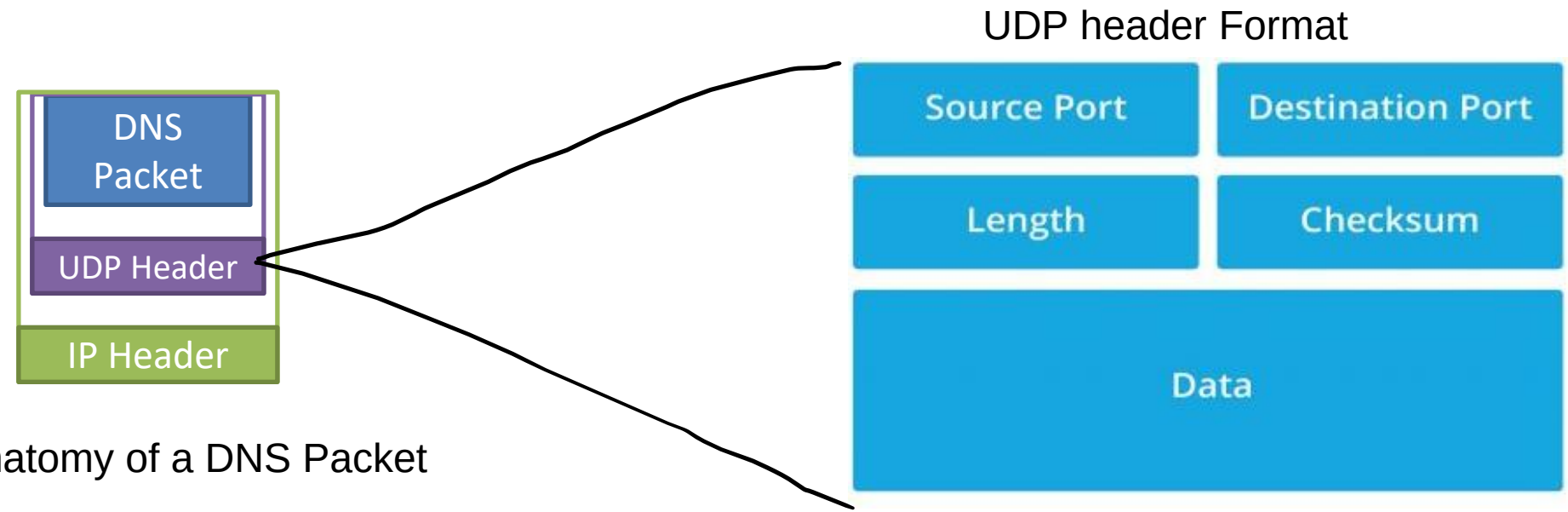
Anatomy of a DNS Packet

Domain Name System (DNS)

Application-level protocol used to map Domain Names to IP Addresses

DNS uses UDP as the transport layer protocol

- No handshake
- No guarantee that packet will arrive



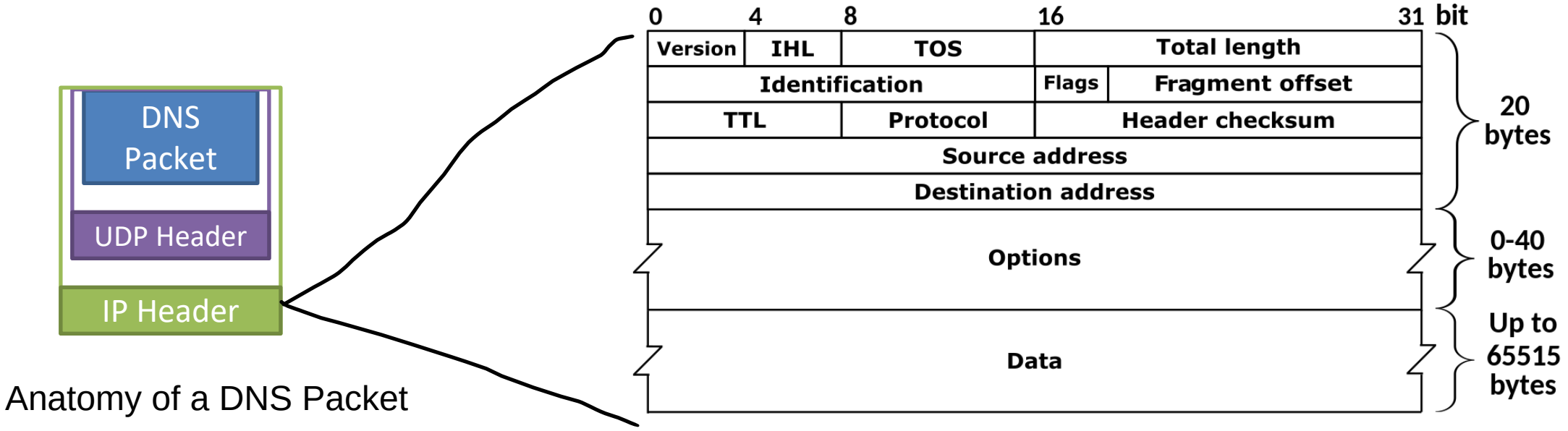
Anatomy of a DNS Packet

Domain Name System (DNS)

Application-level protocol used to map Domain Names to IP Addresses

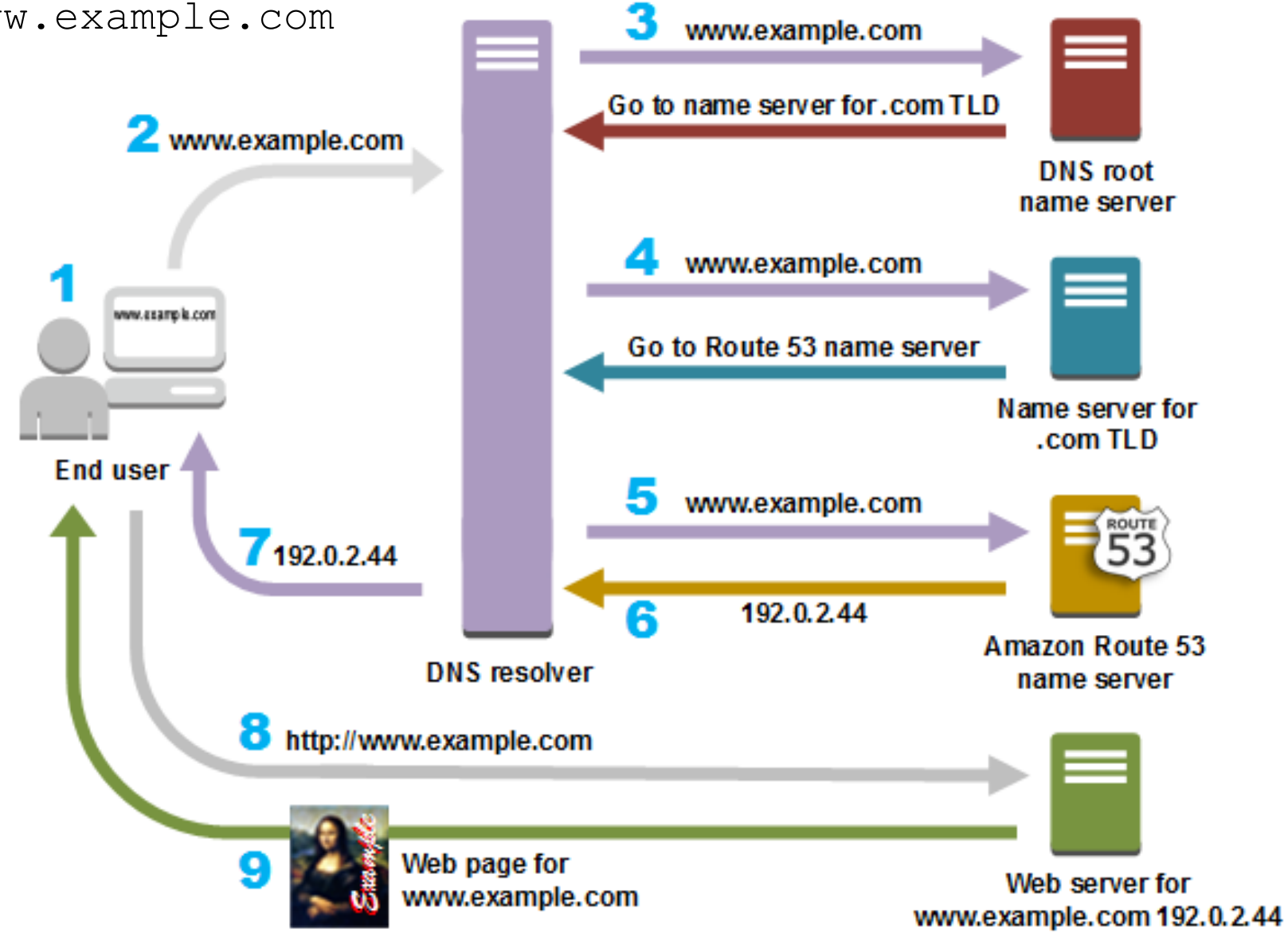
DNS uses UDP as the transport layer protocol

- No handshake
- No guarantee that packet will arrive



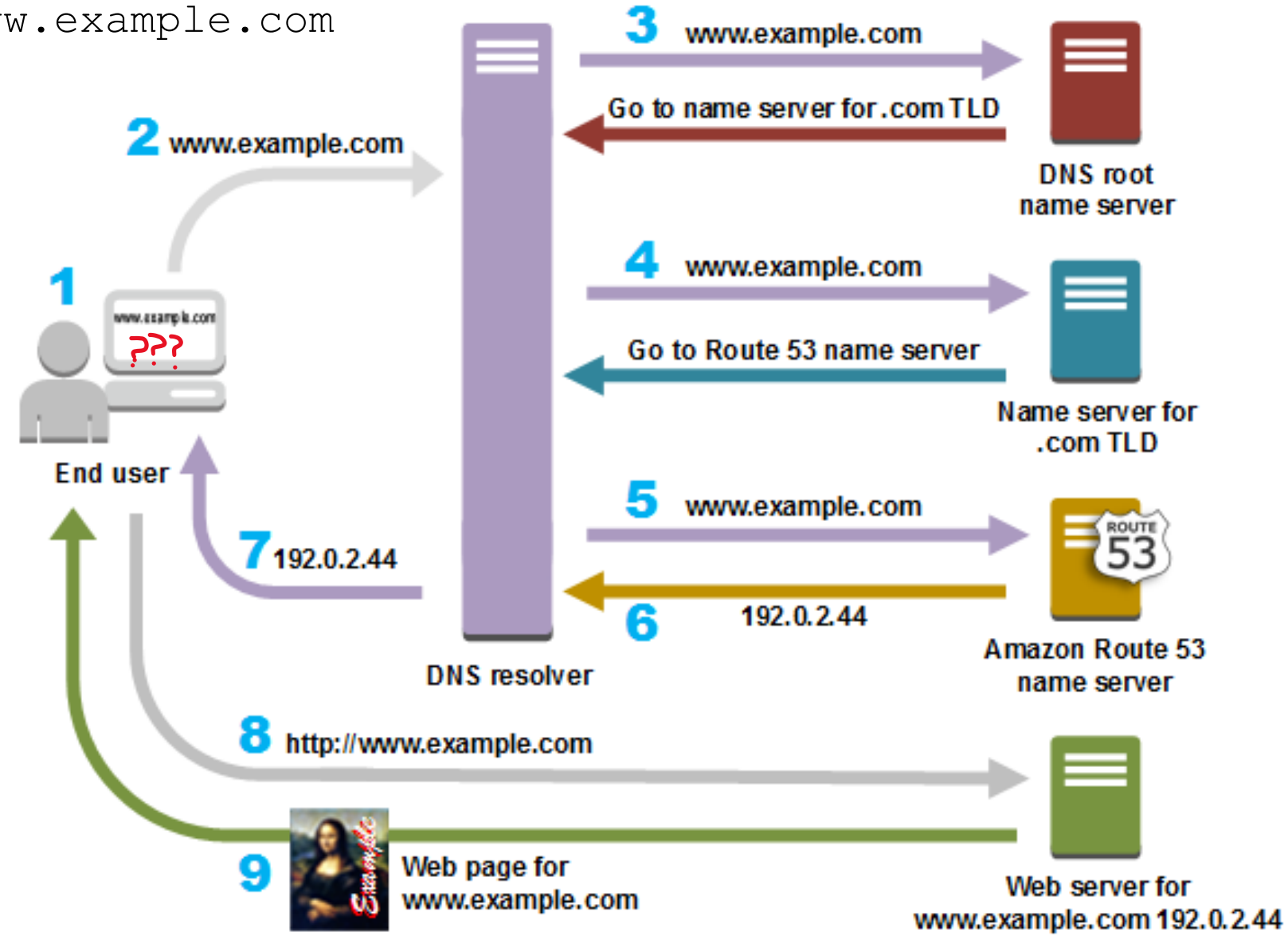
Anatomy of a DNS Packet

Suppose the user wants to go to `www.example.com`



Suppose the user wants to go to `www.example.com`

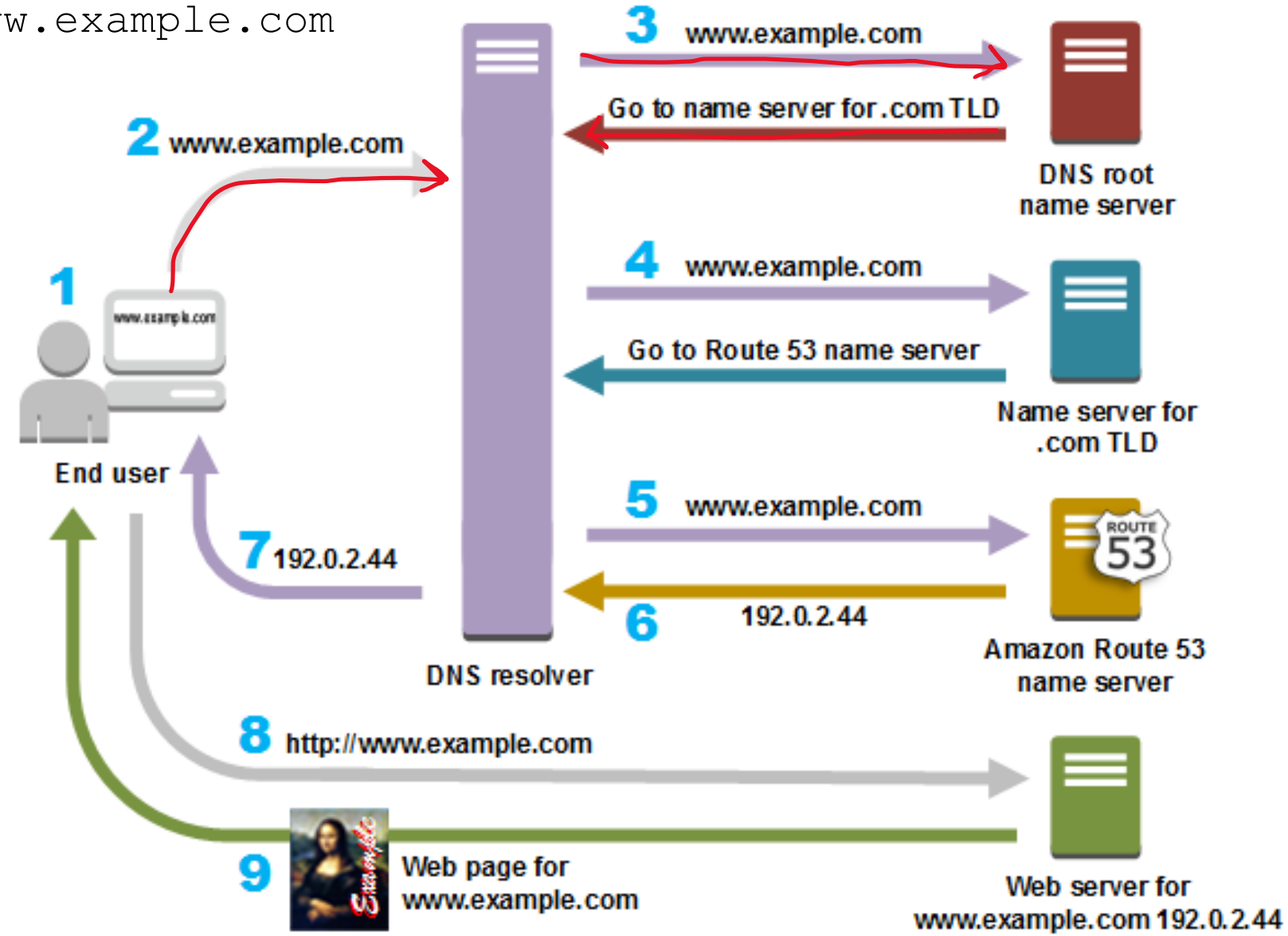
Step 0: The computer first checks its **local cache** to see if an entry exists



Suppose the user wants to go to `www.example.com`

Step 0: The computer first checks its **local cache** to see if an entry exists

Step 1: The user contacts a DNS resolver, which contacts a DNS root name server for the .com TLD

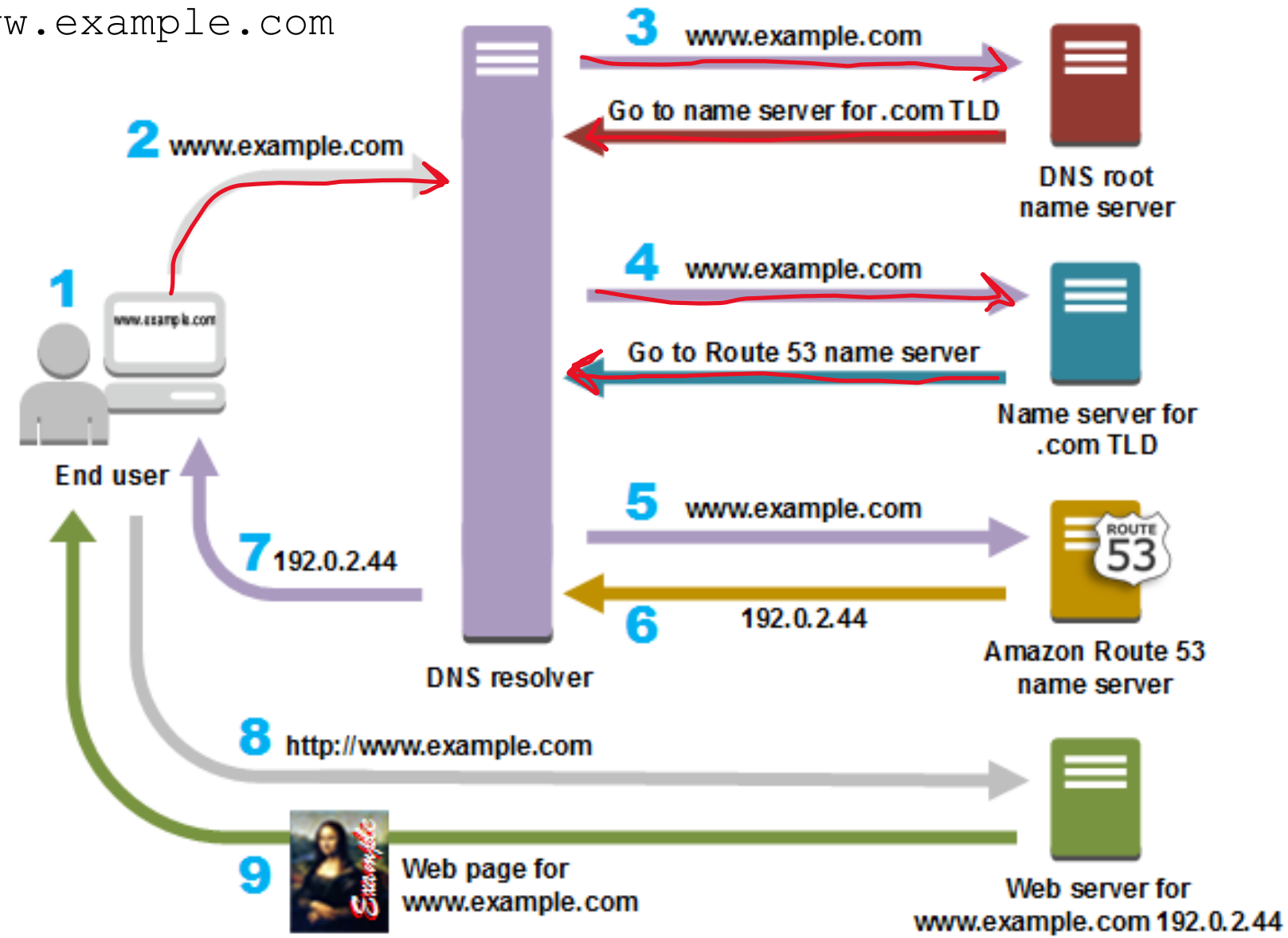


Suppose the user wants to go to `www.example.com`

Step 0: The computer first checks its **local cache** to see if an entry exists

Step 1: The user contacts a DNS resolver, which contacts a DNS root name server for the .com TLD

Step 2: The DNS resolver now contacts the .com TLD server, which returns the IP address of the example.com's Authorative server



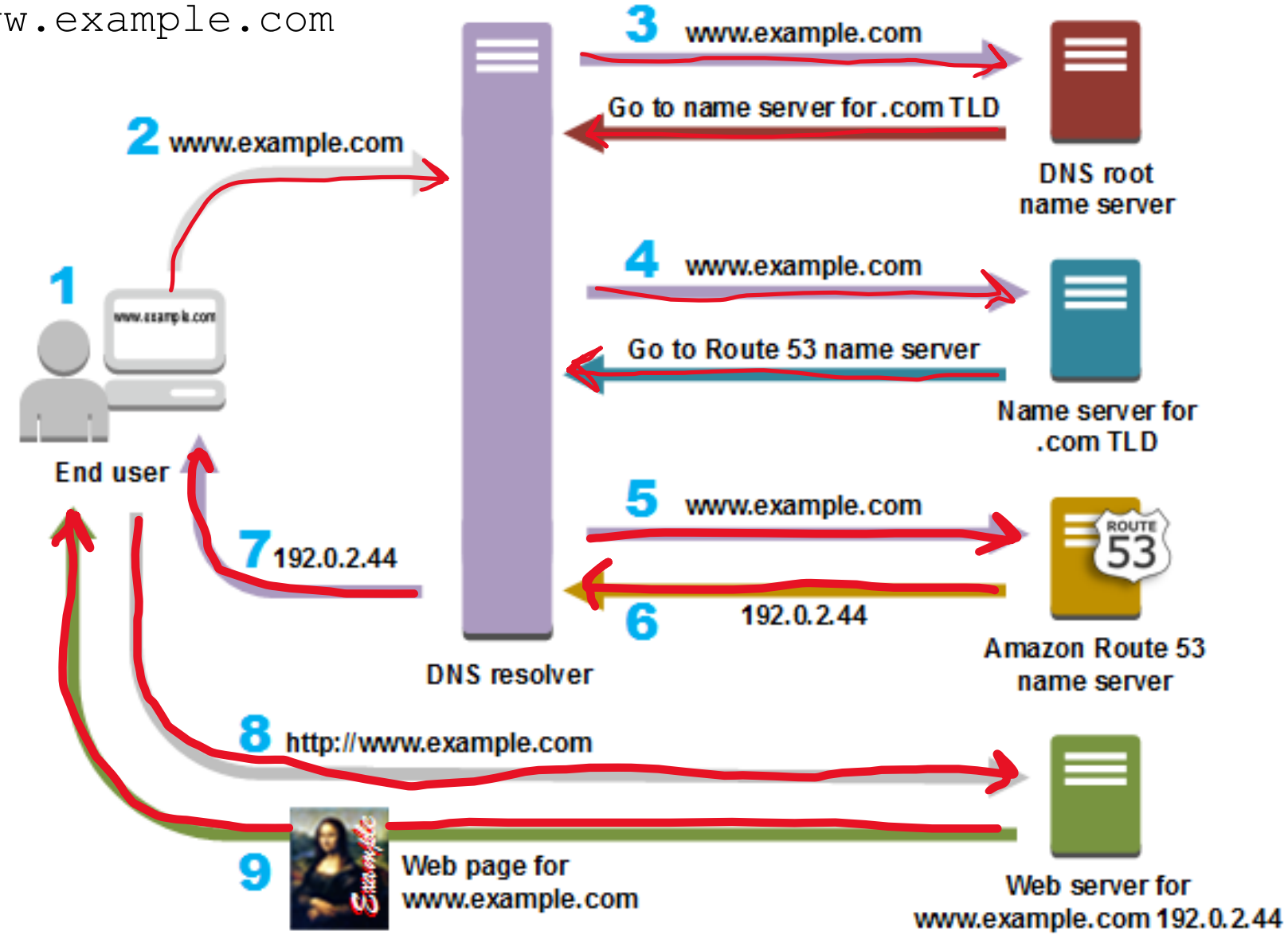
Suppose the user wants to go to `www.example.com`

Step 0: The computer first checks its **local cache** to see if an entry exists

Step 1: The user contacts a DNS resolver, which contacts a DNS root name server for the .com TLD

Step 2: The DNS resolver now contacts the .com TLD server, which returns the IP address of the example.com's Authorative server

Step 3: The Authorative server gives us the IP address for www.example.com, and we can now send an HTTP request to that IP address!



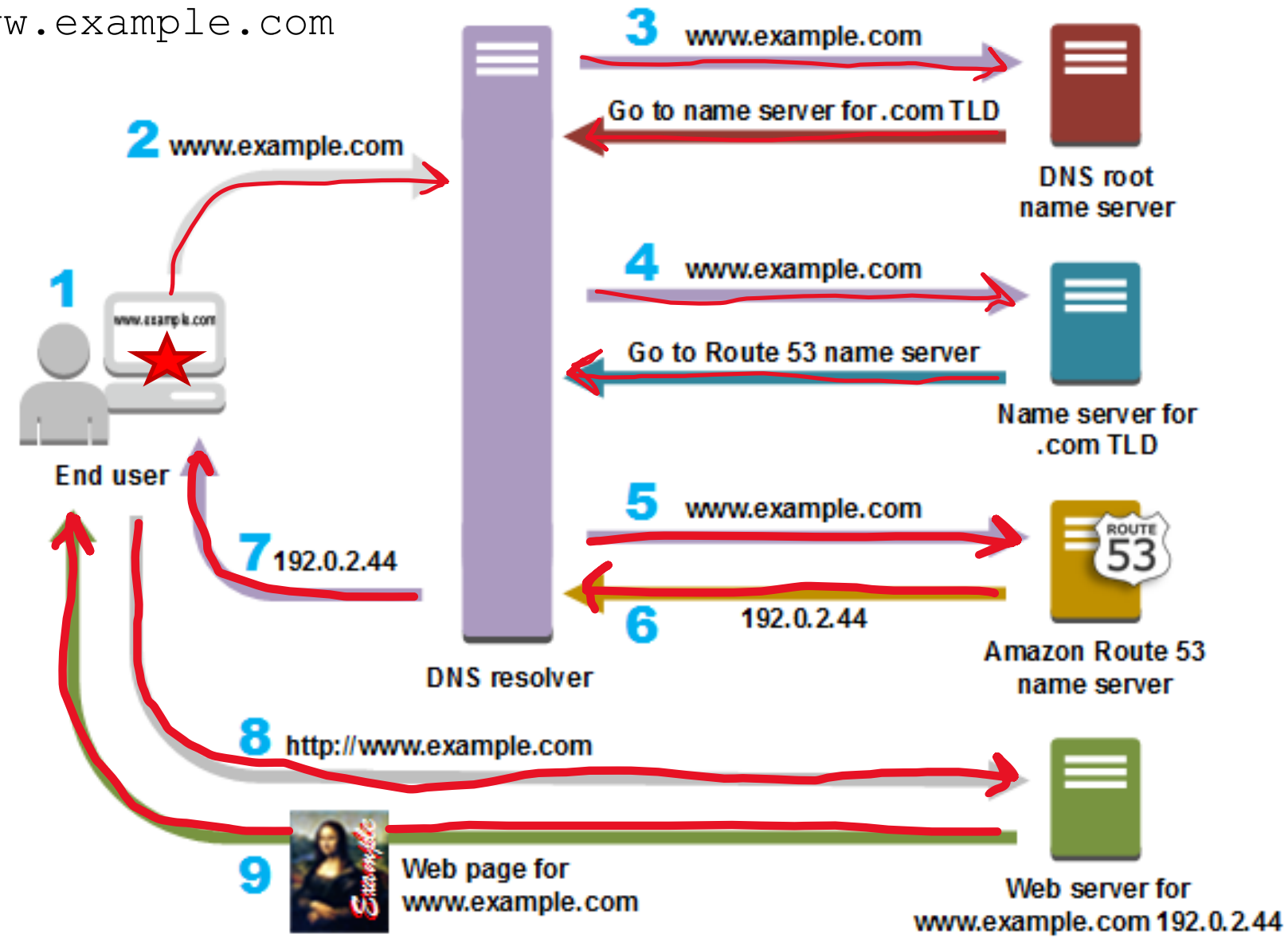
Suppose the user wants to go to `www.example.com`

Step 0: The computer first checks its **local cache** to see if an entry exists

Step 1: The user contacts a DNS resolver, which contacts a DNS root name server for the .com TLD

Step 2: The DNS resolver now contacts the .com TLD server, which returns the IP address of the example.com's Authorative server

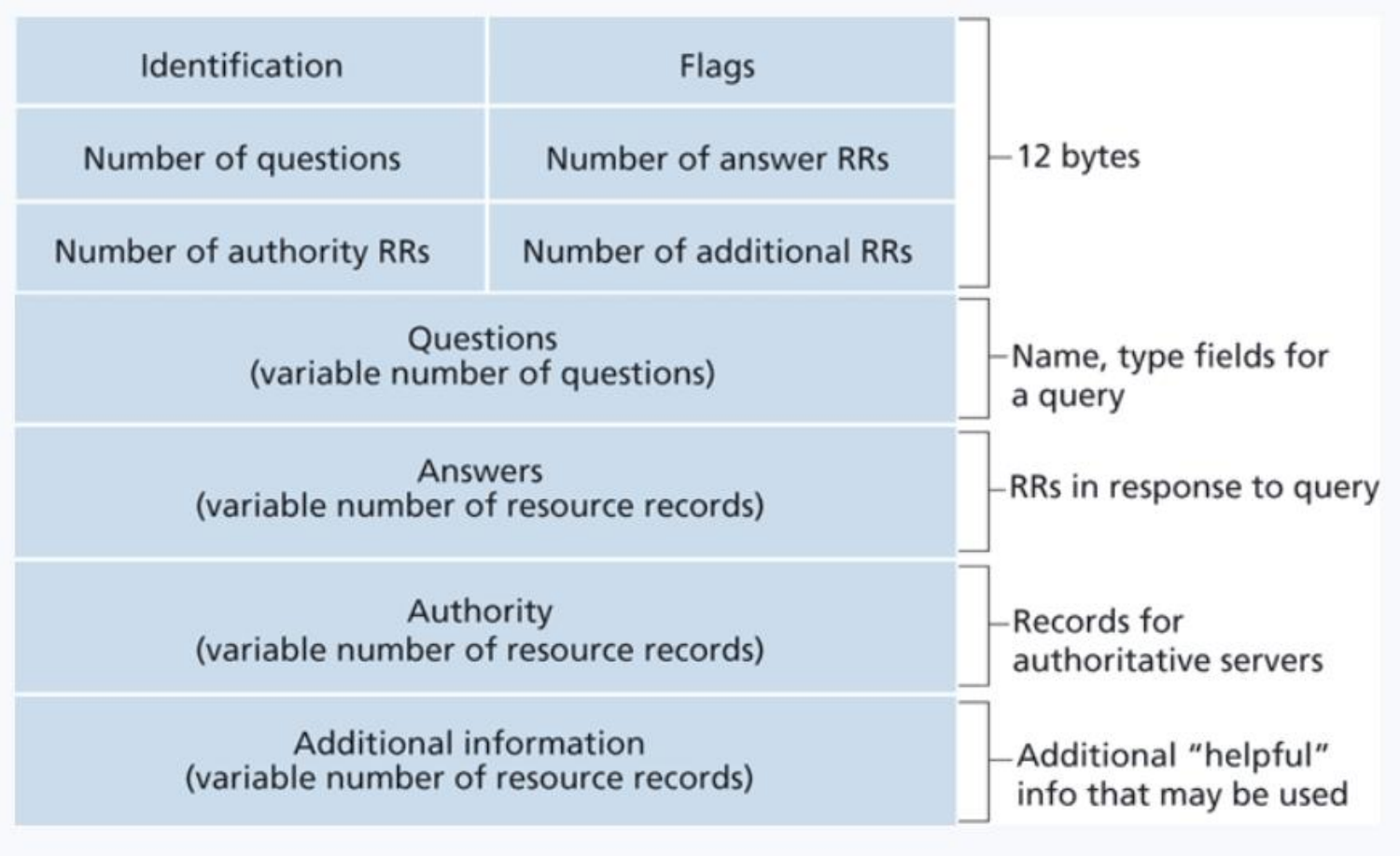
Step 3: The Authorative server gives us the IP address for www.example.com, and we can now send an HTTP request to that IP address!



IMPORTANT

The user's machine will now save the IP address for www.example.com in its **cache**

DNS Header



The domain name of the request
Ie. Google.com

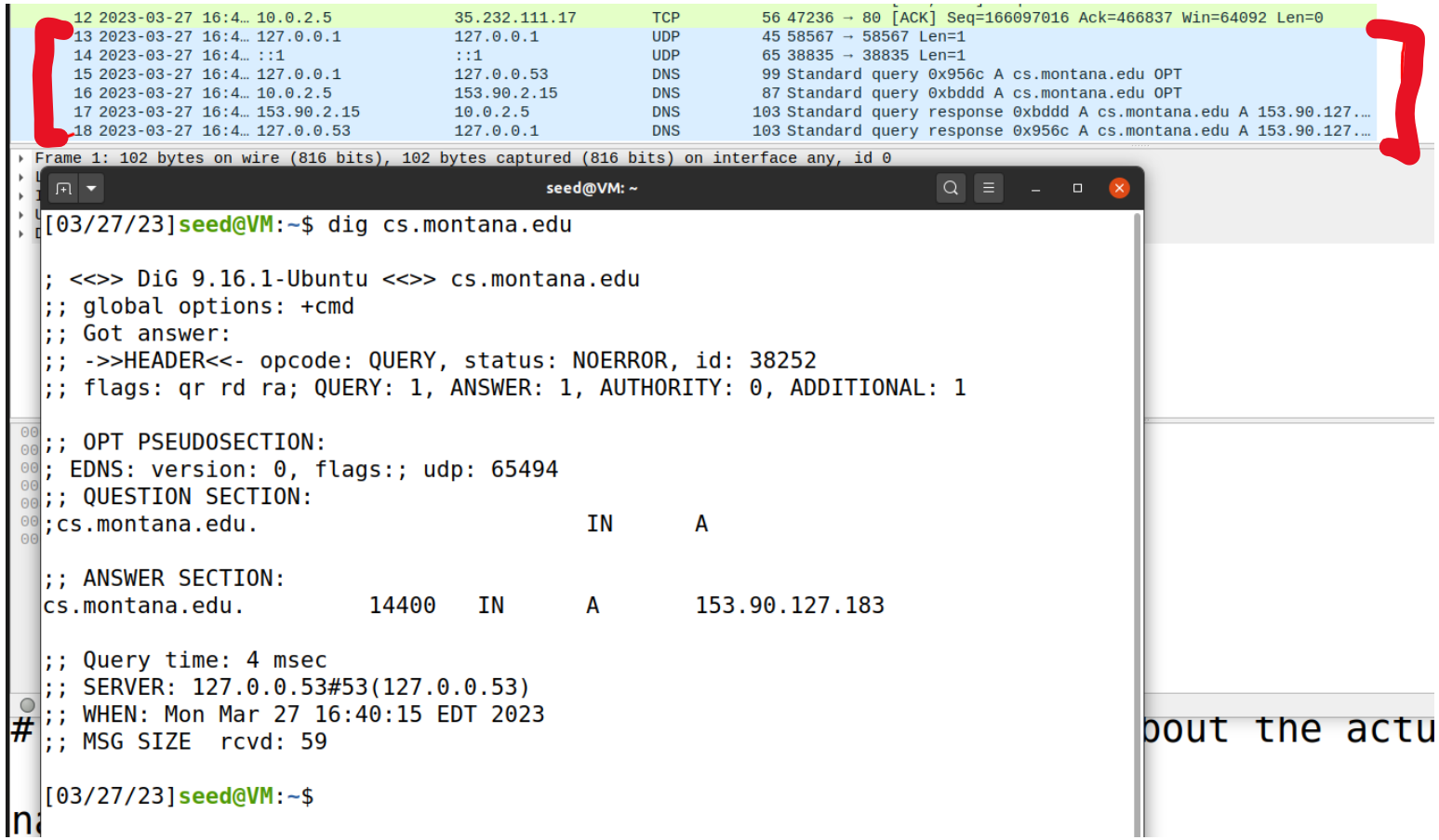
If the IP address was found, it will go here

Contains records that point towards authoritative nameservers

Contains records that point towards authoritative nameservers

DNS In Wireshark

The **`dig`** command is used to issue DNS requests via the command line



On Linux, the `/etc/hosts` holds static IP mappings for domain names

```
[03/27/23]seed@VM:~/.../tcp_attacks$ cat /etc/hosts
127.0.0.1       localhost
127.0.1.1       VM

# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters

# For DNS Rebinding Lab
192.168.60.80   www.seedIoT32.com

# For SQL Injection Lab
10.9.0.5        www.SeedLabSQLInjection.com

# For XSS Lab
10.9.0.5        www.xsslabelgg.com
10.9.0.5        www.example32a.com
10.9.0.5        www.example32b.com
10.9.0.5        www.example32c.com
10.9.0.5        www.example60.com
```

If we can compromise a machine, we can update /etc/hosts and inject IP address for *malicious* webpages

On Linux, the `/etc/resolv.conf` holds IP mappings for DNS server

```
[03/27/23]seed@VM:~/.../tcp_attacks$ cat /etc/resolv.conf
# Dynamic resolv.conf(5) file for glibc resolver(3) generated by resolvconf(8)
#     DO NOT EDIT THIS FILE BY HAND -- YOUR CHANGES WILL BE OVERWRITTEN
# 127.0.0.53 is the systemd-resolved stub resolver.
# run "systemd-resolve --status" to see details about the actual nameservers.

nameserver 127.0.0.53
search msu.montana.edu
```

If we can compromise a machine, we can update /etc/resolv.conf and inject IP address for *malicious* DNS servers**

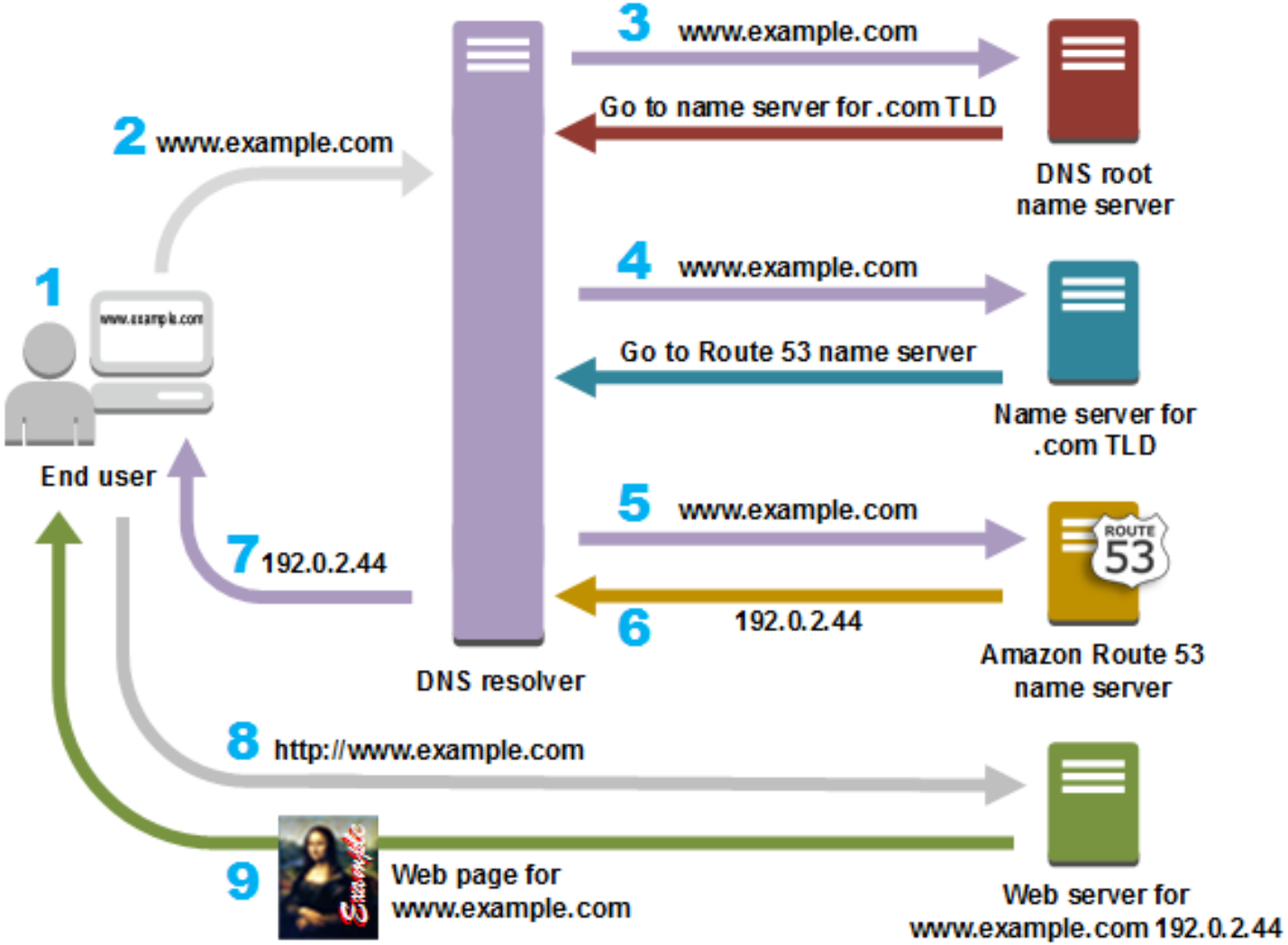
**much more difficult

Attacks on the DNS protocol

When the user sends out a DNS request for a website they want to visit, they will have to **wait** for a response from a DNS server

This process of DNS resolving can take some time…

If an attacker wanted to cause some trouble, they could ???

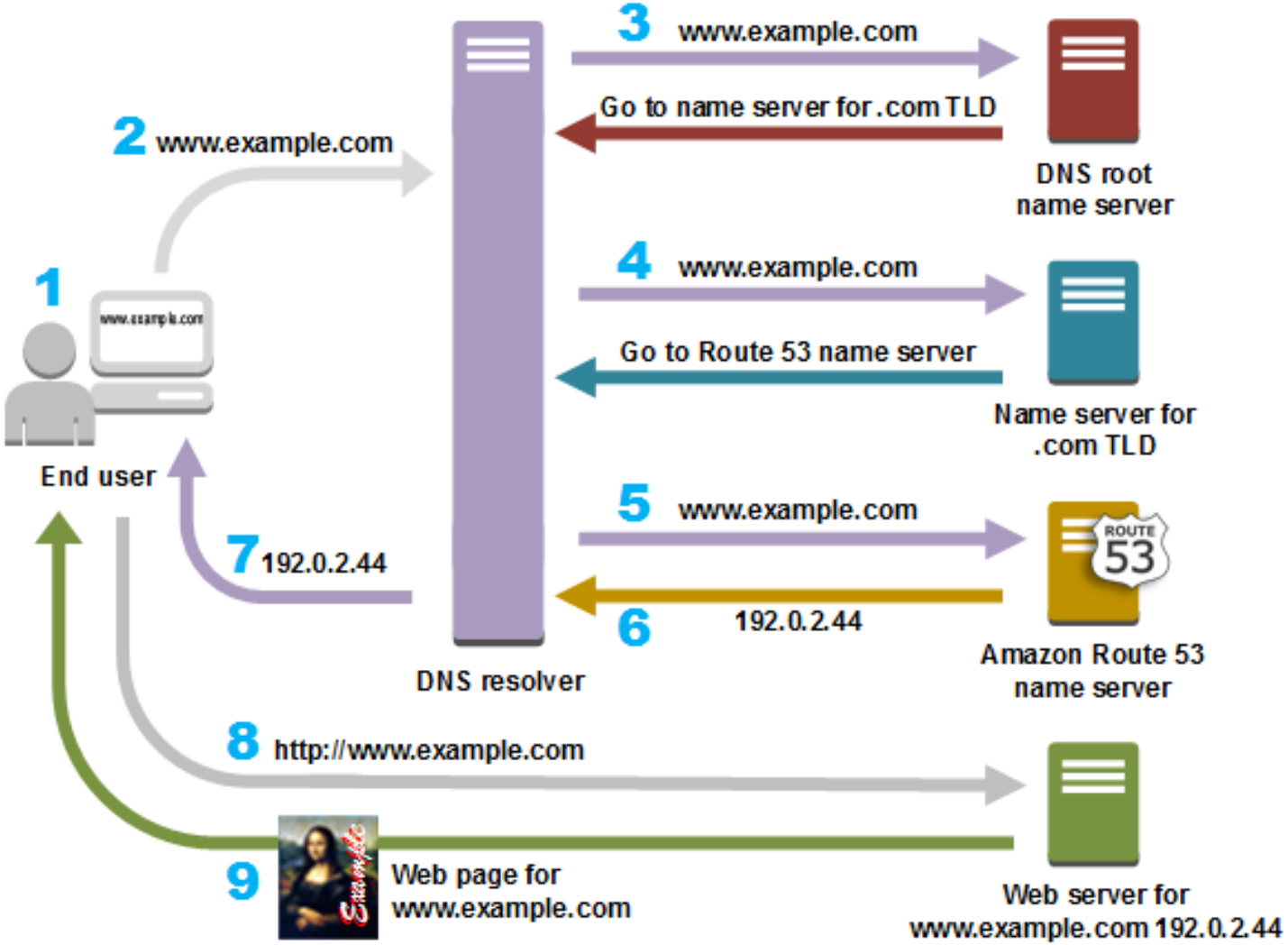


Attacks on the DNS protocol

When the user sends out a DNS request for a website they want to visit, they will have to **wait** for a response from a DNS server

This process of DNS resolving can take some time…

If an attacker wanted to cause some trouble, they could spoof a packet to the user that has a **malicious** DNS response



DNS Cache Poisoning Attack

A **DNS** cache poisoning attack is done by tricking a server into accepting malicious, spoofed DNS information

Instead of going to the IP address of the legitime website, they will go to the IP address that we place in our malicious DNS response (spoofed)

The DNS response is CACHED, which means the user will visit the malicious website in future visits**