Network Naming

CMSC 414

November 8, 2017

Finding IP Addresses

"Layer 2.5" protocols

Use Layer 2, but not really internetworking protocols

Provide ways to get your, or some other host's, IP *or* MAC address

Allows us to build/join networks, and move traffic around a subnet

Address Resolution Protocol

The problem:

- Node P is in a subnet
- P has a packet to send to another node Q in the same subnet
- P knows Q's IP address
- P need Q's MAC address to compose the ethernet frame

P and Q might be *end-hosts* or *routers*

ARP provides the bindings between IP/MAC addresses

How ARP Works

Node maintains an **ARP cache**

```
? (10.104.80.1) at 0:0:c:7:ac:0 on en0 ifscope [ethernet]
? (172.16.0.1) at 2:e0:52:ac:35:ac on en8 ifscope [ethernet]
? (172.16.0.97) at 90:e6:ba:4f:ef:27 on en8 ifscope [ethernet]
? (172.16.3.206) at d4:9a:20:d0:41:aa on en8 ifscope [ethernet]
? (172.16.6.8) at a8:20:66:3b:5e:1c on en8 ifscope [ethernet]
? (172.16.255.253) at cc:4e:24:d1:b0:0 on en8 ifscope [ethernet]
? (224.0.0.251) at 1:0:5e:0:0:fb on en8 ifscope permanent [ethernet]
? (239.255.255.250) at 1:0:5e:7f:ff:fa on en8 ifscope permanent [ethernet]
? (239.255.255.250) at 1:0:5e:7f:ff:fa on en0 ifscope permanent [ethernet]
```

Is IP_Q in the ARP cache?

- *Yes* $\Rightarrow$ use the corresponding MAC_Q
- *No* $\Rightarrow$ layer-2 broadcast *ARP Request*

Q receives broadcast, sends *ARP Response* (with TTL) to P

Q stores entry for P, as does anyone with existing entry for P
$\Rightarrow$ refreshes TTL

ARP Cache Poisoning

ARP is *not authenticated*

ARP is *stateless*

$\Rightarrow$ *responses* assumed to be replies to *actual requests*

Eve wants to receive Alice's traffic to Bob

She can **poison** (or spoof) Alice's ARP cache by sending

ARP	Sender MAC	Sender IP	Target MAC	Target IP
Resp	MAC_{Alice}	IP_{Alice}	MAC_{Eve}	IP_{Bob}

Alice *never asked for this*, but will *add this to her cache*

Eve is now a MitM (unidirectionally) from Alice to Bob

$\Rightarrow$ Enables further attacks

Preventing ARP Spoofing

Static ARP addresses

$\Rightarrow$ Can be useful for critical servers

Cross-checking with other services

Watching for changes to IP $\leftrightarrow$ MAC bindings

Ignore unsolicited responses

$\Rightarrow$ What about requests overheard?

Dynamic Host Configuration Protocol

How do we join a network?

- Host IP address
- Default router IP address
- Nameserver IP address

We can *statically assign* IP addresses
$\Rightarrow$ Becomes prohibitive for large networks

DHCP lets nodes connect to a layer-2 network and ask for a layer-3 address, routing, and name resolver information

How DHCP Works

Alice wants to join a network

$\Rightarrow$ Bcast a **DHCPDISCOVER** message to `255.255.255.255`

Received by **DHCP server** (or *relay agent*, which fwds to server)

DHCP server has pool of available IP addrs on subnet

- Assigns addr to Alice's MAC (often prefers prev assignment)
- Might have some hard-coded MAC $\leftrightarrow$ IP mappings
- Sends default router IP for subnet
- Specifies nameservers

DHCP is *not authenticated*

DHCP Starvation

Trudy can *repeatedly join* the network

$\Rightarrow$ Spoofed MAC addresses

$\Rightarrow$ Consume all available IP addresses

This is a **DHCP starvation** attack

$\Rightarrow$ Form of denial-of-service

On wired Ethernet, can limit MAC addrs/port

Harder to do on WiFi

Bonus DoS: send a forged DHCP release!

$\Rightarrow$ I only see a few mentions of this as a possible attack...

Rogue DHCP Servers

Trudy can run her own DHCP server, and give out bogus

- Host IP addresses
  $\Rightarrow$ Alice's packets collide/are mis-delivered (DoS)
- Router addresses
  $\Rightarrow$ Trudy can MitM Alice's connection
- Nameserver addresses
  $\Rightarrow$ Trudy can resolve hostnames to IP addrs of her choosing

Also called **DHCP Snooping**

Somehow, authenticating DHCP hasn't gotten traction...

Group Exercise 1

Let's look at ARP and DHCP. The `arp` command lets you display and manipulate the ARP table on a host. On your VM, try running

```
arp -n
```

What do you see? Contrast this with running this on your laptop. For a Mac, the command would be

```
arp -n -a
```

For Windows, it would be

```
arp -a
```

Now let's fire up Wireshark. With packet capturing started, try running `arp -d -a`, and then see what comes up.

If you're lucky, you might see DHCP packets, as well.

Domain Name System

The problem:

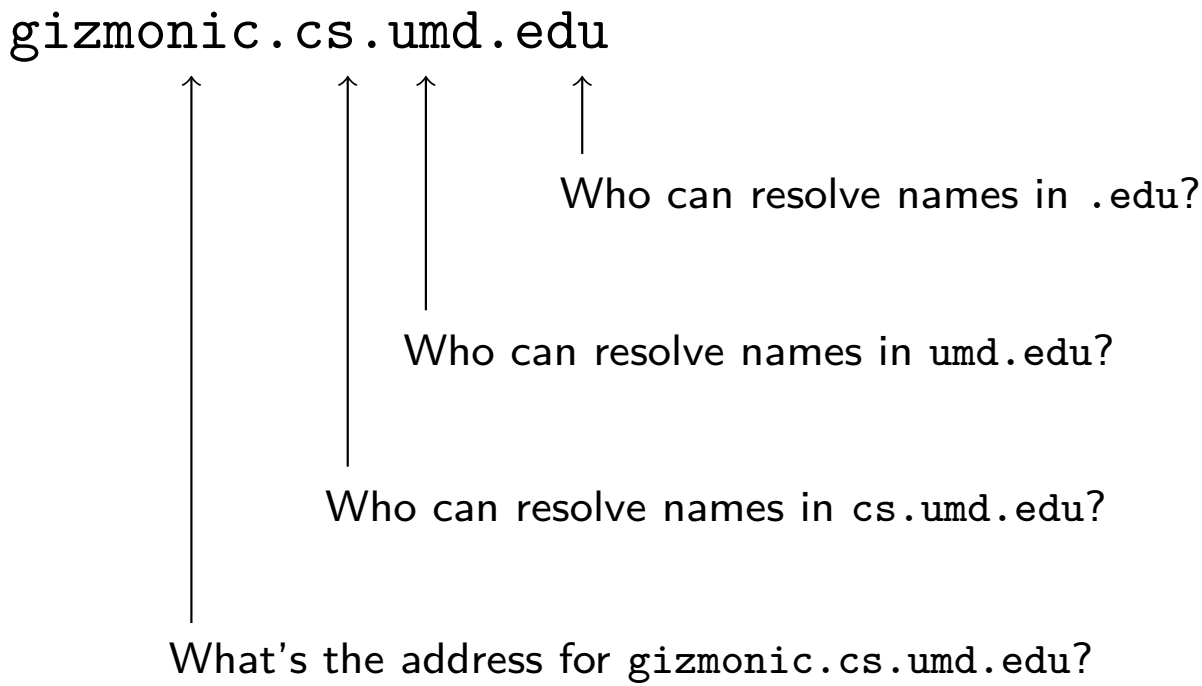
- We reach servers (or other hosts) with IP addresses
- These aren't exactly easy to memorize
- Sometimes the addresses change!

DNS provides bindings between *names* and *addresses*

Global database, with *hierarchical authority*

How DNS Works

Alice wants to contact `gizmonic.cs.umd.edu`

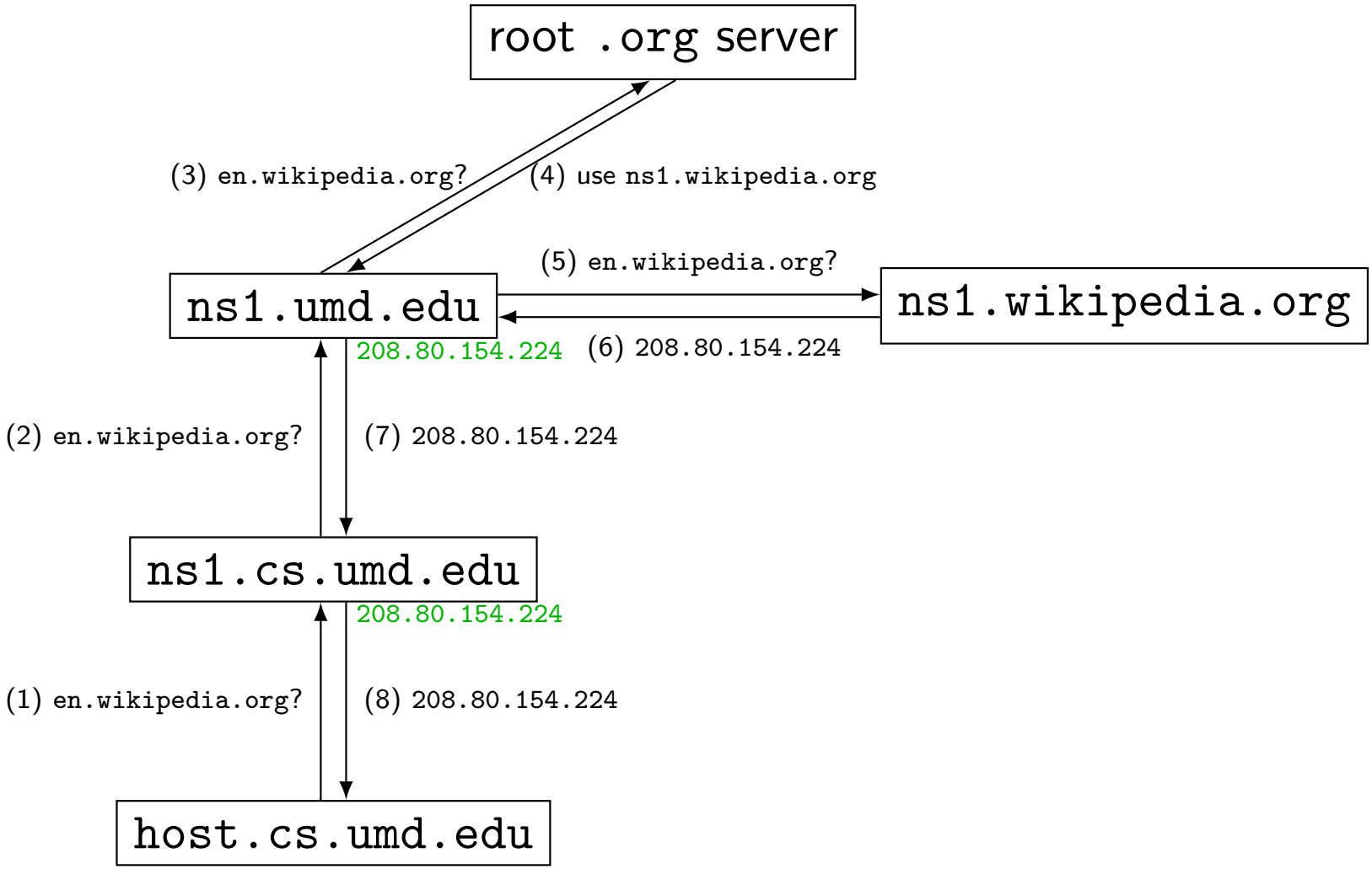


Our local DNS server *might* know the answer

- If not, walk up to the **root server**, then back down
- Servers hit recursively *cache* responses for later requests

DNS Server Hierarchy

We usually do a *recursive query*



DNS Query Types and Methods

There are many *types* of queries

- **A record** — the IP address for a hostname
- **AAAA record** — the IPv6 address for a hostname
- **MX record** — the mail server for a domain
- **NS record** — the name server for a domain
- etc.

There are also two *methods*

- **recursive query** — intermediate servers forward requests they are unable to satisfy; responses are *cached* for later reference
- **iterative query** — intermediate servers respond directly to the querying host, which passes the query to the next server in the tree itself

DNS is *not authenticated*

DNS Cache Poisoning

Attacker wants to direct users to their malicious site

Picks a *target hostname*, like `www.bankofamerica.com`

Also picks a *victim DNS server*, which serves the users they want to exploit

Flaw: Lack of authentication for sources of responses
Discovered by researcher Dan Kaminsky in 2008

DNS Cache Poisoning

The steps:

1. Send the victim server a request for `www.bankofamerica.com`
2. Immediately send the victim server a response for this request with their malicious site's IP address
3. The victim sends the response back to the attacker, and caches the malicious IP address for `www.bankofamerica.com`
4. A user looking up `www.bankofamerica.com` gets the poisoned cache entry

This requires beating the legitimate response to the victim

DNS Zone Transfers

A **zone** is a domain or group of domains under single authority

Each zone has a **primary** server that is authoritative for DNS
$\Rightarrow$ Also has at least one **secondary** server

Zone details are sent by primary to authorized secondary with **DNS zone transfer** messages

Any host can claim to be a DNS secondary server for a domain
$\Rightarrow$ Gains access to complete zone information
$\Rightarrow$ Learns host names without having to guess them

If servers must be within domain
$\Rightarrow$ Refuse zone transfer reqs from hosts outside of legit subnets

Protecting DNS

DNSSEC adds cryptographic authentication to DNS

Only for zones that implement it
$\Rightarrow$ Backwards-compatible

Stores cryptographic info (including keys) in special DNS records

Requires no additional infrasructure

Unfortunately, not widely used

Group Exercise 2

DNS packets are a lot easier to observe in Wireshark, because they're so common. Background processes in your VM are probably triggering DNS lookups all the time, and you can do more by browsing the web and seeing the results.

Look for evidence of cached responses. You can also use the `dig` command in the terminal to control whether recursive or iterative lookups are being done.