

CSCE 463/612

Networks and Distributed Processing

Spring 2017

Application Layer III

Dmitri Loguinov

Texas A&M University

February 9, 2017

Original slides copyright © 1996-2004 J.F Kurose and K.W. Ross

Chapter 2: Roadmap

2.1 Principles of network applications

2.2 Web and HTTP

2.3 FTP

2.4 Electronic Mail

- SMTP, POP3, IMAP

2.5 DNS

2.6 P2P file sharing

2.7 Socket programming with TCP

2.8 Socket programming with UDP

2.9 Building a Web server

DNS: Domain Name System

- People: many identifiers
 - Name, SSN, passport #
- **Internet hosts, routers:**
 - IP address (32 bit) used for routing datagrams
 - Names (e.g., yahoo.com) used by humans

Q: how to map between IP addresses and names?

- Original technique: static /etc/hosts file

Domain Name System:

- **Distributed database**
 - Implemented in hierarchy of many *name servers*
- **Application-layer protocol**
 - Hosts communicate with name servers to *resolve* names/IPs
 - UDP port 53
 - Single-packet query
 - Single-packet response

DNS

Services

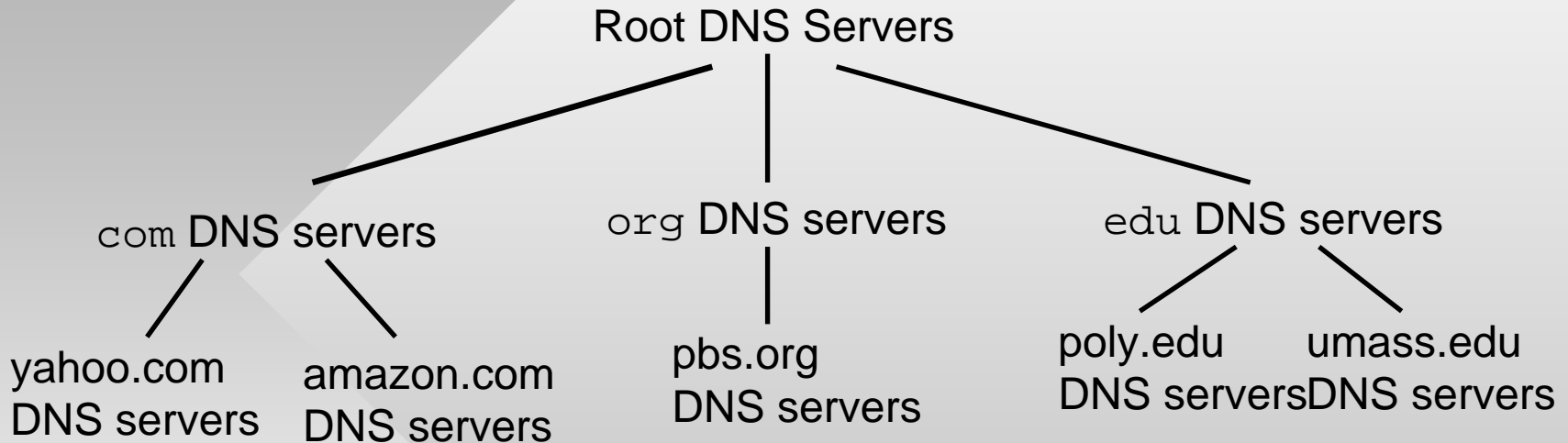
- **Forward** lookup
 - Hostname to IP translation
- **Reverse** lookup
 - IP to hostname
- Host aliasing
- Mail server lookup
- Load distribution
 - Replicated web servers: set of IP addresses for one DNS name

Why not centralize DNS?

- Single point of failure
- Traffic volume (bandwidth, request rate)
- Lack of geographic proximity to user, hence high latency
- Inflexible (can't run code customized for domain)

Doesn't *scale!*

Distributed, Hierarchical Database



- Client wants IP for www.amazon.com:
 - Queries a root server to find the `com` DNS server
 - Queries the `com` server to get the `amazon.com` server
 - Queries the `amazon.com` DNS server to get IP address for `www.amazon.com`
- Who to ask about the location of root servers?
 - Nobody, their IPs are hardwired into OS

Types of DNS Servers

- There are 13 **root** servers (called A-M), each with a fixed IP address
 - Some servers (e.g., A, C, F, I) are geographically distributed across multiple sites (254 total in 2011)
 - More info: <http://root-servers.org/>

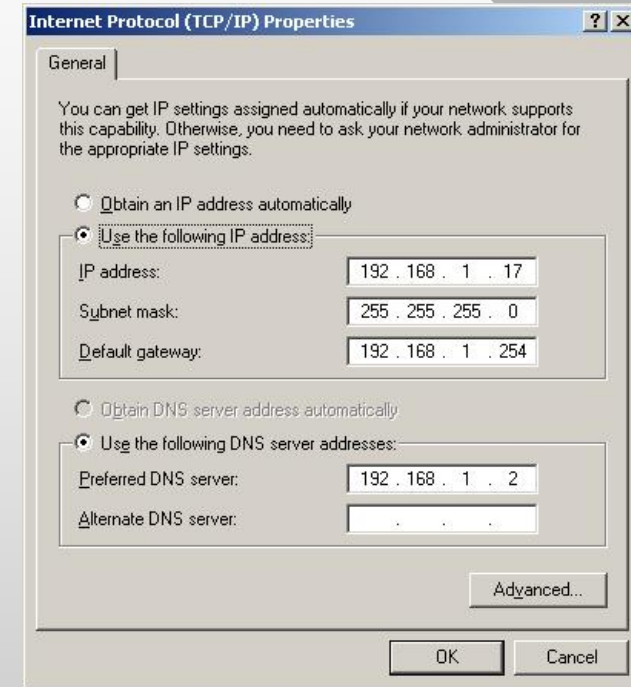


Types of DNS Servers

- **Top-level domain (TLD)** servers: responsible for generic TLDs (e.g., .com, .org, .net, .edu) and all country-code TLDs (e.g., .uk, .fr, .ca, .jp)
 - Around 1500 total gTLDs and cc-TLDs
 - Verisign runs .com , Educause .edu
- **Authoritative** servers: provide authoritative mappings for company servers (e.g., Web and mail)
 - Can be maintained by organization (e.g., amazon.com) or service provider (e.g., ISP or hosting company)
- **Local** name server: does not belong to the hierarchy
 - Any computer that accepts requests and then finds out the answer by traversing the DNS tree

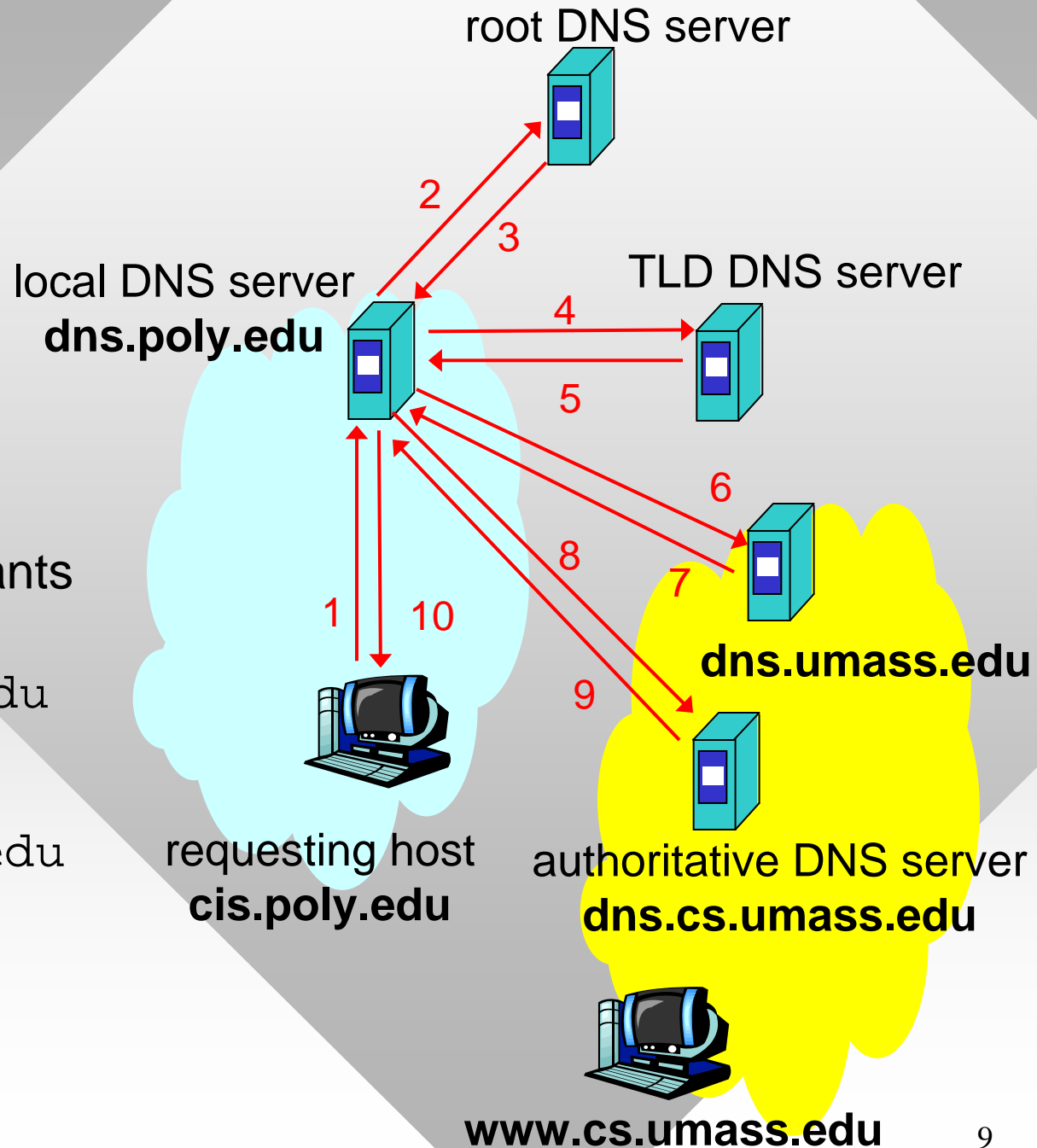
Local Name Servers

- Each network (ISP, company, university) has a few
 - Preferred DNS server in network options (alternate used for backup)
 - If you run BIND, set this to 127.0.0.1
 - Auto-configure via DHCP or set to 8.8.8.8 (Google DNS)
- When a host makes a DNS query (application calls gethostbyname), query is sent to local DNS server
 - Local server acts as a proxy (cache) and forwards query into hierarchy if it cannot answer it from cache
- Command-line tool for DNS queries is nslookup
 - Homework #2 implements essentially this



Example

- `cis.poly.edu` wants the IP address for `www.cs.umass.edu`
- Next request for `joe.cs.umass.edu`
– What happens?



Recursive queries

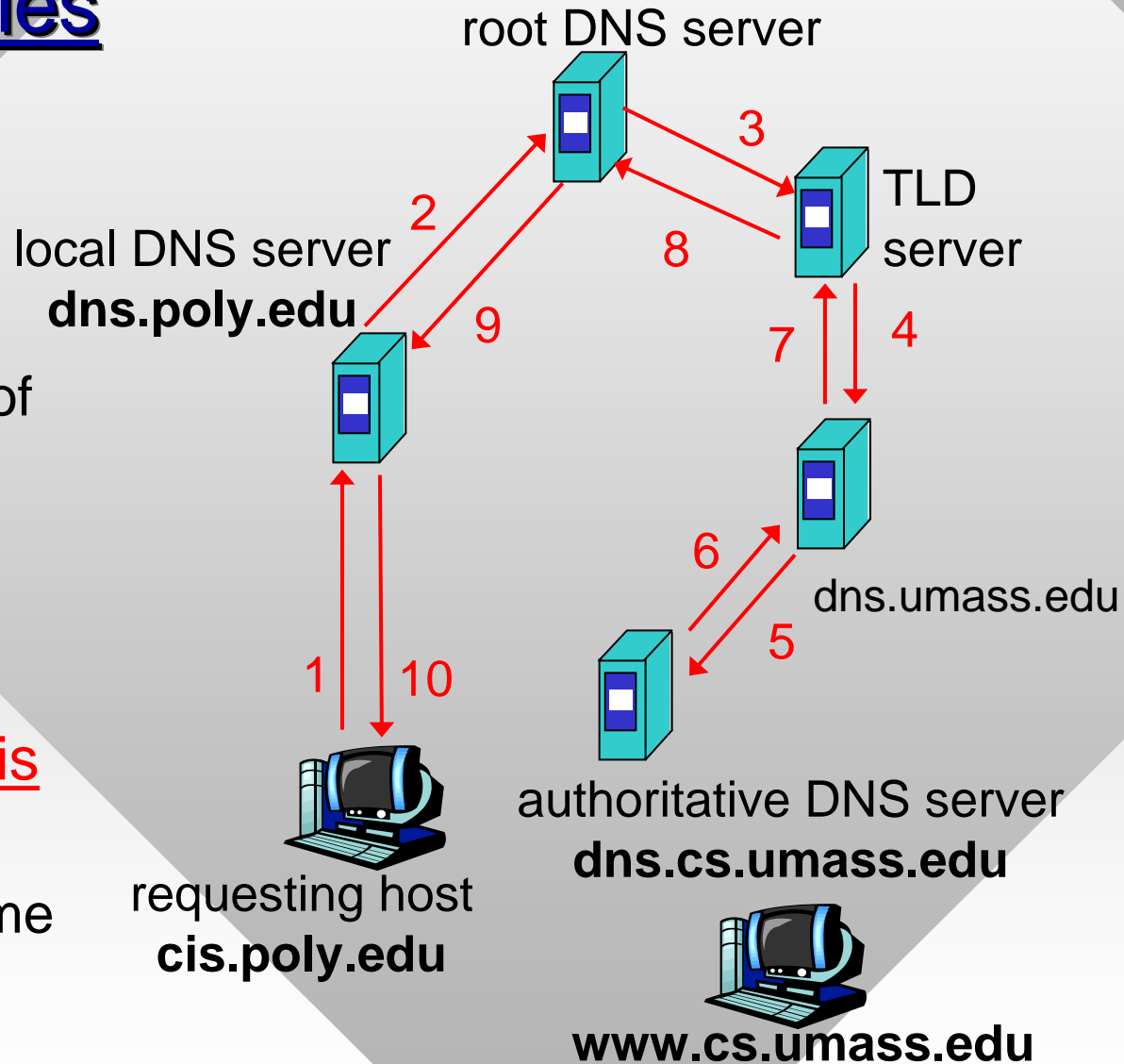
Iterated query

(previous page):

- Contacted server replies with name of server to contact
- “I don’t know this name, but ask this server”

Recursive query (this page):

- Puts burden of name resolution on contacted name server



DNS: Caching and Updating Records

- Once (any) name server learns a mapping, it *caches* the mapping
 - Cache entries time out (disappear) after some time (TTL)
 - Unexpired entries are served directly from cache, in which case they are called *non-authoritative*
 - If the original DNS server is contacted, the response is *authoritative*
- TLD servers are typically cached in local name servers
 - Thus root name servers not supposed to be visited often
- Study in 2007 showed load on individual root servers A-M was about 6-16K queries/sec
 - During DDoS attacks in 2001 it was 38K/sec

Replace A with AAAA for IPv6

DNS Records

DNS: distributed database of resource records (RR)

(name, value, type, ttl)

- Type A
 - name = host
 - value = IPv4 address (4 byte DWORD)
- Type NS
 - name = domain
 - value = hostname of authoritative name server for this domain
- Type CNAME
 - name = host
 - value = host it's aliased to
 - Reduces manual effort to change IPs and other records
- Type MX
 - name = domain
 - value = name of SMTP server associated with domain

Inserting Records Into DNS

- Example: just created startup “Network Utopia”
- Register name networkutopia.com at a **registrar** (e.g., Network Solutions)
 - Need to provide registrar with names and IP addresses of your authoritative name server (primary and secondary)
 - Registrar inserts four RRs into the `com` TLD server:
 - (networkutopia.com, dns1.networkutopia.com, NS)
 - (dns1.networkutopia.com, 212.212.212.1, A)
 - (networkutopia.com, dns2.networkutopia.com, NS)
 - (dns2.networkutopia.com, 212.212.212.2, A)
- In your authoritative server, place a Type A record for `www` and a Type MX record for entire domain

Reverse Queries

- Reverse DNS lookups are performed using a special construction of a fake DNS name
 - Reason: DNS resolves names from right to left with the semantics of going from the most general to the most specific
 - In IPs, the MSB is most general, LSB is most specific
- The IP address is reversed and is followed by “in-addr.arpa” (or “ip6.arpa” for IPv6)
 - Example: 128.194.135.65 is requested as **65.135.194.128.in-addr.arpa**
 - The query type must be set to **PTR**
- RFC 1035 (1987) describes DNS headers/commands
 - Also see <http://www.networksorcery.com/enp/protocol/dns.htm>

DNS Protocol, Messages

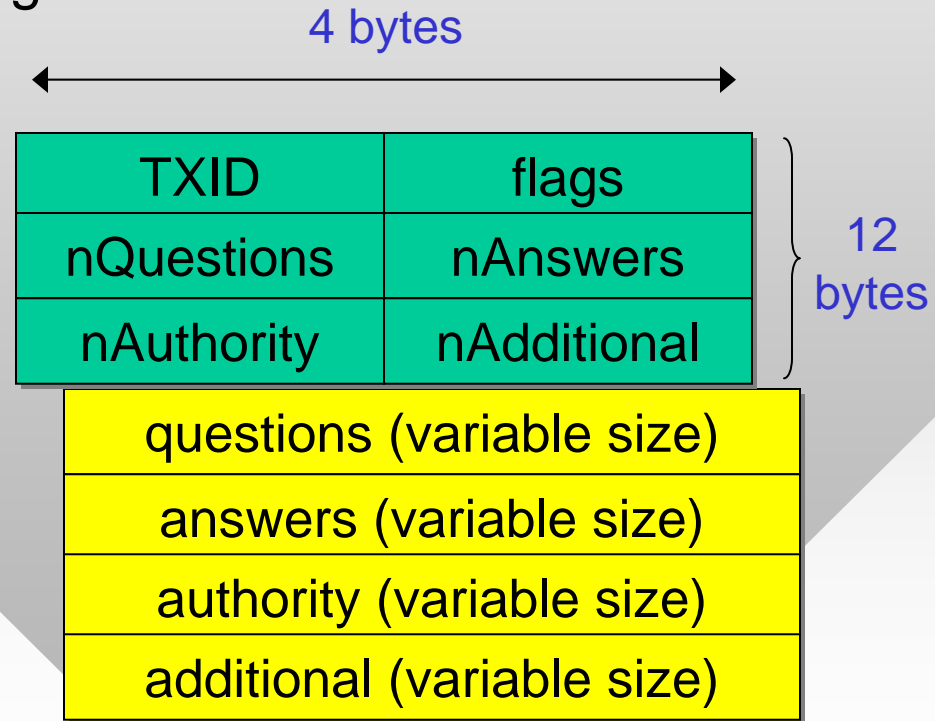
- Query and reply messages use same format
 - Packet starts with a fixed DNS header (12 bytes)
 - Followed by a variable-length section

- **Transaction ID (TXID)**

- 16-bit number assigned by client to each query
 - Echoed by server in response packet

- **Flags** specify the type of request being made and response status

- The other 4 fields provide a count of records in each variable-size section

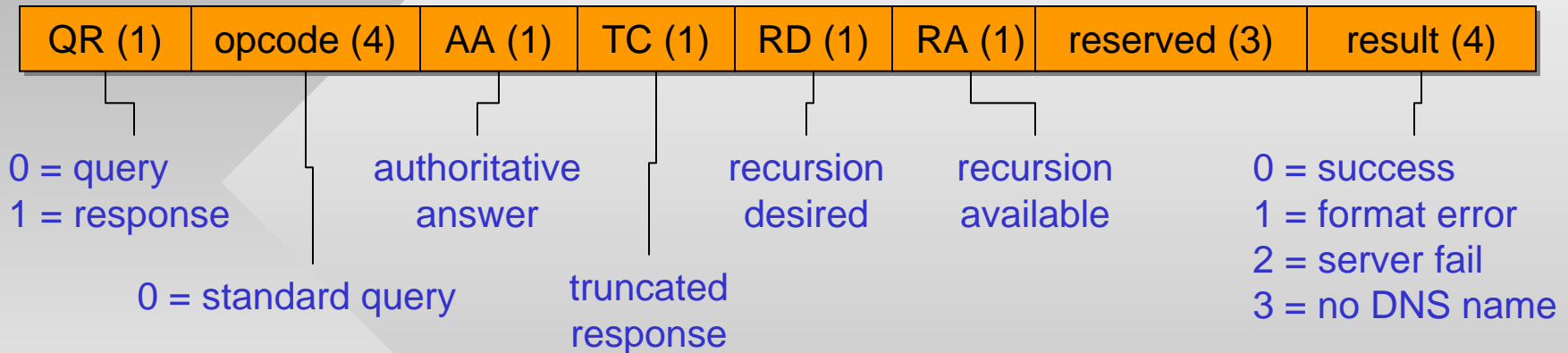


DNS Protocol, Messages

- Queries contain only the question section
 - Most servers expect one question per packet
- Response packets always repeat the question
 - Safety mechanism if TXID runs into collision at the client
- **Authority** section carries NS record(s)
 - Used during iterative lookups to specify the next DNS server to query (similar to HTTP redirects)
- All numbers are in **network byte order**
 - Use proper conversion (i.e., htons()) in this case

| | |
|----------------------------|-------------|
| TX ID | flags |
| nQuestions | nAnswers |
| nAuthority | nAdditional |
| questions (variable size) | |
| answers (variable size) | |
| authority (variable size) | |
| additional (variable size) | |

DNS Flags



- For binary fields, 1 = true and 0 = false
- For query packets, you must:
 - Set RD = 1; all other fields must be zero
 - Specify nQuestions = 1
 - Correctly create the actual question and append it to the header in the packet buffer

Nslookup Usage (Windows)

- nslookup -querytype=mx cs.tamu.edu

cached
answers
and
additional
records

```
Server: gw.irl.cs.tamu.edu
Address: 128.194.135.72

Non-authoritative answer:
cs.tamu.edu MX preference = 100, mail exchanger = smtp-relay.tamu.edu
cs.tamu.edu MX preference = 10, mail exchanger = pine.cs.tamu.edu

smtp-relay.tamu.edu internet address = 165.91.143.199
pine.cs.tamu.edu internet address = 128.194.138.12
```

- nslookup -querytype=hinfo cs.tamu.edu

```
Server: gw.irl.cs.tamu.edu
Address: 128.194.135.72

cs.tamu.edu
primary name server = dns1.cs.tamu.edu
responsible mail addr = root.cs.tamu.edu
serial = 2006090513
refresh = 1800 (30 mins)
retry = 900 (15 mins)
expire = 1209600 (14 days)
default TTL = 3600 (1 hour)
```

smaller MX
preference value
means higher priority

Nslookup Usage (Windows)

- nslookup -querytype=ptr 12.138.194.128.in-addr.arpa

```
Server: gw.irl.cs.tamu.edu
Address: 128.194.135.72
```

```
Non-authoritative answer:
```

```
12.138.194.128.in-addr.arpa name = mail.cs.tamu.edu
12.138.194.128.in-addr.arpa name = pine.cs.tamu.edu
12.138.194.128.in-addr.arpa name = pophost.cs.tamu.edu
12.138.194.128.in-addr.arpa name = mailhost.cs.tamu.edu
12.138.194.128.in-addr.arpa name = pop.cs.tamu.edu
12.138.194.128.in-addr.arpa name = imap.cs.tamu.edu
```

nslookup performs string reversal transparently, but hw2 will need to do this explicitly

- nslookup -querytype=ptr 12.1.55.186

```
Server: s18.irl.cs.tamu.edu
Address: 128.194.135.58
```

```
Non-authoritative answer:
```

```
186.55.1.12.in-addr.arpa canonical name = 186.184/29.55.1.12.in-addr.arpa
186.184/29.55.1.12.in-addr.arpa name = outlook.milestonescientific.com
```

Using UDP

- DNS runs over UDP that has no connection phase
 - Each request and response is **exactly 1 packet**
 - Calls to `recvfrom()` and `sendto()` correspond to receiving/sending 1 packet from/to a socket
 - No need to loop on receive
- General idea:

```
sock = socket (AF_INET, SOCK_DGRAM, 0);
// bind sock to port 0 - see the handout
while (work to be done) {
    createRequest(buf, nextName);
    sendto (sock, buf, len, 0, &addressTo, ...);
    ...
    if (select (...) > 0) {
        recvfrom (sock, recvBuf, len, 0, &addressFrom...);
        parseResponse (recvBuf);
    }
}
closesocket (sock);
```