DNS: Domain Name System

People: many identifiers:
- SSN, name, passport #

Internet hosts, routers:
- IP address (32 bit) - used for addressing datagrams
- “name”, e.g., www.yahoo.com - used by humans

Q: map between IP addresses and name?

Domain Name System:
- distributed database implemented in hierarchy of many name servers
- application-layer protocol host, routers, name servers to communicate to resolve names (address/name translation)
  - note: core Internet function, implemented as application-layer protocol
  - complexity at network’s “edge”
DNS

DNS services
• hostname to IP address translation
• host aliasing
  – Canonical, alias names
• mail server aliasing
• load distribution
  – replicated Web servers: set of IP addresses for one canonical name

Why not centralize DNS?
• single point of failure
• traffic volume
• distant centralized database
• maintenance doesn’t scale!
Distributed, Hierarchical Database

Client wants IP for www.amazon.com; 1st approx:

- client queries a root server to find com DNS server
- client queries com DNS server to get amazon.com DNS server
- client queries amazon.com DNS server to get IP address for www.amazon.com
DNS: Root name servers

- contacted by local name server that can not resolve name
- root name server:
  - contacts authoritative name server if name mapping not known
  - gets mapping
  - returns mapping to local name server

13 root name servers worldwide

- a Verisign, Dulles, VA
- c Cogent, Herndon, VA (also LA)
- d U Maryland College Park, MD
- g US DoD Vienna, VA
- h ARL Aberdeen, MD
- i Verisign, (21 locations)
- e NASA Mt View, CA
- f Internet Software C. Palo Alto, CA (and 36 other locations)
- j Verisign, (21 locations)
- k RIPE London (also 16 other locations)
- l ICANN Los Angeles, CA
- m WIDE Tokyo (also Seoul, Paris, SF)
TLD and Authoritative Servers

• Top-level domain (TLD) servers:
  – responsible for com, org, net, edu, etc, and all top-level country domains uk, fr, ca, jp.
  – Network Solutions maintains servers for com TLD
  – Educause for edu TLD

• Authoritative DNS servers:
  – organization’s DNS servers, providing authoritative hostname to IP mappings for organization’s servers (e.g., Web, mail).
  – can be maintained by organization or service provider
Local Name Server

• does not strictly belong to hierarchy
• each ISP (residential ISP, company, university) has one.
  – also called “default name server”
• when host makes DNS query, query is sent to its local DNS server
  – acts as proxy, forwards query into hierarchy
DNS name resolution example

- Host at cis.poly.edu wants IP address for gaia.cs.umass.edu

**iterated query:**
- contacted server replies with name of server to contact
- “I don’t know this name, but ask this server”
DNS name resolution example

**recursive query:**
- puts burden of name resolution on contacted name server
- heavy load?

Diagram:
- Requesting host: cis.poly.edu
- Local DNS server: dns.poly.edu
- Authoritative DNS server: dns.cs.umass.edu
- TLD DNS server
- Root DNS server

流程步骤:
1. Requesting host cis.poly.edu sends a request to local DNS server dns.poly.edu.
2. Local DNS server sends a request to root DNS server.
3. Root DNS server checks the TLD DNS server.
4. TLD DNS server returns the authoritative DNS server dns.cs.umass.edu.
5. Authoritative DNS server sends the IP address to TLD DNS server.
6. TLD DNS server sends the IP address to root DNS server.
7. Root DNS server sends the IP address to local DNS server.
8. Local DNS server returns the IP address to requesting host.

Questions:
- Does this recursive query put a heavy load on the contacted name server?
- Is there a better way to handle name resolution?
DNS: caching and updating records

• once (any) name server learns mapping, it caches mapping  
  – cache entries timeout (disappear) after some time  
  – TLD servers typically cached in local name servers  
    • Thus root name servers not often visited  

• update/notify mechanisms under design by IETF  
  – RFC 2136  
DNS records

DNS: distributed db storing resource records (RR)

RR format: (name, value, type, ttl)

- **Type=A**
  - **name** is hostname
  - **value** is IP address

- **Type=NS**
  - **name** is domain (e.g. foo.com)
  - **value** is hostname of authoritative name server for this domain

- **Type=CNAME**
  - **name** is alias name for some “canonical” (the real) name
  - **value** is canonical name

- **Type=MX**
  - **value** is name of mailserver associated with **name**
DNS protocol, messages

**DNS protocol**: query and reply messages, both with same message format

**msg header**

- **identification**: 16 bit # for query, reply to query uses same #

- **flags**:
  - query or reply
  - recursion desired
  - recursion available
  - reply is authoritative
## DNS protocol, messages

<table>
<thead>
<tr>
<th>Identification</th>
<th>Flags</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of questions</td>
<td>Number of answer RRs</td>
</tr>
<tr>
<td>Number of authority RRs</td>
<td>Number of additional RRs</td>
</tr>
</tbody>
</table>

- **Name, type fields** for a query
- **RRs in response to query**
- **Records for authoritative servers**
- **Additional “helpful” info that may be used**
Inserting records into DNS

- example: new startup “Network Utopia”
- register name networkuptopia.com at DNS registrar (e.g., Network Solutions)
  - provide names, IP addresses of authoritative name server (primary and secondary)
  - registrar inserts two RRs into com TLD server:
    - (networkutopia.com, dns1.networkutopia.com, NS)
    - (dns1.networkutopia.com, 212.212.212.1, A)

- create authoritative server Type A record for www.networkuptopia.com; Type MX record for networkutopia.com
- How do people get IP address of your Web site?
DNS Vulnerabilities

• Original DNS design focused on data availability
  – DNS zone data is replicated at multiple servers.
  – A DNS zone works as long as one server is available.
    • DDoS attacks against the root must take out 13 root servers.

• But the DNS design included no authentication.
  – Any DNS response is generally believed.
  – No attempt to distinguish valid data from invalid.
    • Just one false root server could disrupt the entire DNS.
A Simple DNS Attack

Easy to observe UDP DNS query sent to well known server on well known port.

First response wins. Second response is silently dropped on the floor.
A More Complex Attack

Bell Labs Caching Server

www.google.com = 128.9.128.127

Query www.google.com

Any Bell Labs Laptop

Response

www.attacker.com A 128.9.128.127
attacker.com NS ns.attacker.com
attacker.com NS www.google.com
ns.attacker.com A 128.9.128.2
www.google.com A 128.9.128.127

Query

www.attacker.com

Remote attacker

ns.attacker.com
The Problem in a Nutshell

• Resolver can not distinguish between valid and invalid data in a response.

• Idea is to add source authentication
  – Verify the data received in a response is equal to the data entered by the zone administrator.
  – Must work across caches and views.
  – Must maintain a working DNS for old clients.
Authentication DNS Responses

• Each DNS zone signs its data using a private key.
  – Recommend signing done offline in advance

• Query for a particular record returns:
  – The requested resource record set.
  – A signature (SIG) of the requested resource record set.

• Resolver authenticates response using public key.
  – Public key is pre-configured or learned via a sequence of key records in the DNS hierarchy.
Secure DNS Query and Response

End-user

www.darpa.mil

www.darpa.mil = 192.5.18.195

Plus (RSA) signature by darpa.mil

Attacker can not forge this answer without the darpa.mil private key.

Challenge: add signatures to the protocol manage DNS public keys

Authoritative DNS Servers

Caching DNS Server

CS 344/Spring09
Pure P2P architecture

- no always-on server
- arbitrary end systems directly communicate peer-peer
- peers are intermittently connected and change IP addresses

- Three topics:
  - File distribution
  - Searching for information
  - Case Study: Skype
File Distribution: Server-Client vs P2P

**Question**: How much time to distribute file from one server to $N$ peers?

- $u_s$: server upload bandwidth
- $u_i$: peer $i$ upload bandwidth
- $d_i$: peer $i$ download bandwidth

Network (with abundant bandwidth)
File distribution time: server-client

- server sequentially sends $N$ copies:
  - $NF/u_s$ time
- client $i$ takes $F/d_i$ time to download

Time to distribute $F$ to $N$ clients using client/server approach:

$$d_{cs} = \max \{ NF/u_s, F/min(d_i) \}$$

increases linearly in $N$ (for large $N$)
File distribution time: P2P

- server must send one copy: $F/u_s$ time
- client $i$ takes $F/d_i$ time to download
- NF bits must be downloaded (aggregate)
- fastest possible upload rate: $u_s + \sum u_i$

$$d_{P2P} = \max \{ \frac{F}{u_s}, \frac{F}{\min(d_i)}, \frac{NF}{u_s + \sum u_i} \}$$
Server-client vs. P2P: example

Client upload rate = u, F/u = 1 hour, u_s = 10u, d_{min} \geq u_s
File distribution: BitTorrent

- P2P file distribution

**tracker**: tracks peers participating in torrent

**torrent**: group of peers exchanging chunks of a file

obtain list of peers

peer

trading chunks

CS344S10
BitTorrent (1)

- file divided into 256KB *chunks*.
- peer joining torrent:
  - has no chunks, but will accumulate them over time
  - registers with tracker to get list of peers, connects to subset of peers ("neighbors")
- while downloading, peer uploads chunks to other peers.
- peers may come and go
- once peer has entire file, it may (selfishly) leave or (altruistically) remain
BitTorrent (2)

Pulling Chunks
• at any given time, different peers have different subsets of file chunks
• periodically, a peer (Alice) asks each neighbor for list of chunks that they have.
• Alice sends requests for her missing chunks – rarest first

Sending Chunks: tit-for-tat
• Alice sends chunks to four neighbors currently sending her chunks at the highest rate
  • re-evaluate top 4 every 10 secs
  • every 30 secs: randomly select another peer, starts sending chunks
  • newly chosen peer may join top 4
  • “optimistically unchoke”
BitTorrent: Tit-for-tat

1. Alice “optimistically unchokes” Bob
2. Alice becomes one of Bob’s top-four providers; Bob reciprocates
3. Bob becomes one of Alice’s top-four providers

With higher upload rate, can find better trading partners & get file faster!
Distributed Hash Table (DHT)

• DHT = distributed P2P database
• Database has (key, value) pairs;
  – key: ss number; value: human name
  – key: content type; value: IP address
• Peers query DB with key
  – DB returns values that match the key
• Peers can also insert (key, value)
DHT Identifiers

• Assign integer identifier to each peer in range \([0, 2^n - 1]\).
  – Each identifier can be represented by \(n\) bits.

• Require each key to be an integer in same range.

• To get integer keys, hash original key.
  – eg, \(key = h(\text{“Led Zeppelin IV”})\)
  – This is why they call it a distributed “hash” table
How to assign keys to peers?

• Central issue:
  – Assigning (key, value) pairs to peers.

• Rule: assign key to the peer that has the closest ID.

• Convention in lecture: closest is the immediate successor of the key.

• Ex: n=4; peers: 1,3,4,5,8,10,12,14;
  – key = 13, then successor peer = 14
  – key = 15, then successor peer = 1
Circular DHT (1)

- Each peer *only* aware of immediate successor and predecessor.
- “Overlay network”
O(N) messages on avg to resolve query, when there are N peers.

Define closest as closest successor

Who's resp for key 1110?
Circular DHT with Shortcuts

- Each peer keeps track of IP addresses of predecessor, successor, short cuts.
- Reduced from 6 to 2 messages.
- Possible to design shortcuts so $O(\log N)$ neighbors, $O(\log N)$ messages in query.

Who’s resp for key 1110?
Peer Churn

- Peer 5 abruptly leaves
- Peer 4 detects; makes 8 its immediate successor; asks 8 who its immediate successor is; makes 8’s immediate successor its second successor.
- What if peer 13 wants to join?

To handle peer churn, require each peer to know the IP address of its two successors.
- Each peer periodically pings its two successors to see if they are still alive.

- Peer 5 abruptly leaves
- Peer 4 detects; makes 8 its immediate successor; asks 8 who its immediate successor is; makes 8’s immediate successor its second successor.
- What if peer 13 wants to join?
P2P Case study: Skype

- inherently P2P: pairs of users communicate.
- proprietary application-layer protocol (inferred via reverse engineering)
- hierarchical overlay with SNs
- Index maps usernames to IP addresses; distributed over SNs
Peers as relays

- Problem when both Alice and Bob are behind “NATs”.
  - NAT prevents an outside peer from initiating a call to insider peer
- Solution:
  - Using Alice’s and Bob’s SNs, Relay is chosen
  - Each peer initiates session with relay.
  - Peers can now communicate through NATs via relay