CS419: Computer Networks

Lecture 7: Mar 8, 2005

Naming and DNS
“Any problem in computer science can be solved with another layer of indirection”

David Wheeler
Naming is a layer of indirection

- What problems does it solve?
  - Makes objects human readable
  - Hides complexity and dynamics
    - Multiple lower-layer objects can have one name
    - Changes in lower-layer objects hidden
  - Allows an object to be found in different ways
    - One object can have multiple names
Names map to objects through a resolution service
Identifiers and Locators

- A name is always an *identifier* to a greater or lesser extent
  - Can be persistent or non-persistent
  - Can be globally unique, locally unique, or even non-unique
- If a name has structure that helps the resolution service, then the name is also a *locator*
Naming in networks
DNS names map into addresses

Domain Name System (DNS)

Many-to-many

Name

Address

Route

- Hierarchical
- User-friendly
- Location independent
- But not org independent
Addresses map into routes

IP address
(128.94.2.17)
Routing algorithm
(BGP, OSPF, RIP)

One-to-many

- Hierarchical
- Location Dependent
- Non-unique
- Can change often
- Refers to an interface, not a host
Routes get packets to interfaces

- A path
- Source dependent
- Can change often
DNS names and IP addresses are identifiers and locators

- Both are typically non-persistent
- Private IP addresses identify only in the context of an IP realm
- Domain names are good identifiers
  - `woodstock.cs.cornell.edu` identifies a host
  - `www.cnn.com` identifies a service
- URLs are good identifiers
IP address as locator

- A bizarre way to think of an internet route is as a series of “route segments”
  - A “route” from the source host to the first hop router
  - A route from the first hop router to the access ISP
  - A route from the access ISP to the dest ISP
  - A route from the dest ISP to the dest site
  - A route from the dest site to the dest subnet
  - A “route” from the dest subnet to the dest host
IP address as locator

- If we can think of a route as a series of route segments...
- Then we can think of the IP address as a series of “flat” (sub-)addresses
  - Where each (sub-)address maps into a route segment
- ISP-site-subnet-host
So what?

- There is a fundamental thing happening here
- (Hierarchical) route segments prevents all nodes from having to know about the whole network
- Hierarchy always requires a global reference point
  - The top of the hierarchy
  - In IP, this is the ISP
To summarize

- Internet uses *Names*, *Addresses*, and *Routes*
  - Routes are special, because they depend on point of view

- Also *Identifiers* and *Locators*
  - An locator is, in a way, a series of identifiers
  - Where everyone knows how to get to the top, and the top knows how to get to the bottom
Names in the Internet

- The Internet has always had *names*
  - Because IP addresses are hard to remember
- But, the Internet hasn’t always had *domain names*
- Used to be, this was a valid email address:
  - *george@isi*
  - How did any given host know the IP address of “isi”???
The “host table” and DNS

- Before DNS, there was the host table
- This was a complete list of all the hosts in the Internet!
- It was copied every night to every machine on the Internet!
- At some point, this was perceived as a potential scaling bottleneck…
- So a distributed directory called the “Domain Name System” was invented (DNS)
## The host table (historic)

<table>
<thead>
<tr>
<th>Host Name</th>
<th>IP Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>mit-dlab</td>
<td>133.65.14.77</td>
</tr>
<tr>
<td>isi-mail</td>
<td>24.72.188.13</td>
</tr>
<tr>
<td>mit-lcs</td>
<td>133.65.29.1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Distributed Directory

- A primary goal of DNS was to have a distributed “host table”, so that each site could manage its own name-to-address mapping.
- But also, it should scale well!
DNS is simple but powerful

- Only one type of query
  - Query(domain name, RR type)
    - Resource Record (RR) type is like an attribute type
  - Answer(values, additional RRs)

- Example:
  - Query(woodstock.cs.cornell.edu, A)
  - Answer(128.84.97.3)
DNS is simple but powerful

- Limited number of RR types
- Hard to make new RR types
  - Not for technical reasons…
  - Rather because each requires global agreement
DNS is the core of the Internet

- Global name space
  - Can be the core of a naming or identifying scheme
- Global directory service
  - Can resolve a name to nearly every computer on the planet
Important DNS RR types

- **NS**: Points to IP addr of next Name Server down the tree
- **A**: Contains the IP address
  - **AAAA** for IPv6
- **MX**: Contains the name of the mail server
- **CNAME**: “Canonical name”, for aliasing
- **PTR**: Returns name given an IP address
  - reverse DNS lookup
DNS tree structure

- .
- edu.
- com.
- jp.
- us.
- cornell.edu.
- cmu.edu.
- mit.edu.
- cs.cornell.edu.
- eng.cs.cornell.edu.
- foo.cs.cornell.edu A 10.1.1.1
- bar.cs.cornell.edu A 10.1.1.1

NS RR “pointers”
Primary and secondary servers

Cornell.edu: NS RRs point to both primary and secondary servers.

CS.Cornell.edu: RRs are initially configured into primary server.

Primary server replicates RRs onto secondary servers periodically (updates are incremental).
Resolver structure and configuration

- Static configuration of root servers
- Stub resolver resides on client host, points to configured recursive server
- Resolver manages DNS queries on behalf of stub resolvers
Resolver structure and configuration

1. Stub resolver sends recursive query

2,3,4… Resolver makes iterative queries to servers

N. Resolver returns final answer to stub resolver (which also caches result)

Resolver caches results for efficiency
DNS query and reply have same format

msg header
- identification: 16 bit # for query, reply to query uses same #
- flags:
  - query or reply
  - recursion desired
  - recursion available
  - reply is authoritative
  - reply was truncated

<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>

questions (variable number of questions)
answers (variable number of resource records)
authority (variable number of resource records)
additional information (variable number of resource records)
DNS protocol, messages

Name, type fields for a query

RRs in response to query

records for authoritative servers

additional "helpful" info that may be used
UDP or TCP

- DNS usually uses UDP
- Like RPC: query and reply fit into a single unfragmented UDP packet
- Client resends query after timeout
  - About 3 seconds
- Client will use TCP if reply is truncated
  - Truncated bit is set
  - TCP also used for zone transfers
DNS cache management

- All RRs have Time-to-live (TTL) values
- When TTL expires, cache entries are removed
- NS RRs tend to have long TTLs
  - Cached for a long time
  - Reduces load on higher level servers
- A RRs may have very short TTLs
  - Order one minute for some web services
  - Order one day for typical hosts
Caching is the key to performance

- Without caching, the small number of machines at the top of the hierarchy would be overwhelmed.
- But what if you want to change the IP address of a host? How do you change all those cached entries around the world?
  - You can’t…you wait until they timeout on their own, then make your change.
Changing a DNS name

- Say your TTL was set to one day
  - This means that even if you change DNS now, some hosts will continue to use the old address for a day
- So, give the host two IP addresses for a while (the old one and the new one)
  - But DNS only answers with the new one
- After a day, the old one is cleaned out of caches, and you can remove it from the host
Reverse DNS lookup

- Obtain name from address
  - PTR resource record
- To lookup name of 128.5.6.7, do DNS lookup on
  - 7.6.5.128.in-addr.arpa
- This is how traceroute figures out the names of the hosts in a path
dig examples

(dig is a DNS lookup command line tool available on Linux)

- NS for the root
- NS for com
- MX for cornell.edu
- A for cnn.com
Service-oriented DNS RR types

- **SRV**: Contains addresses and ports of services on servers
  - One way to learn what port number to use
- **NAPTR**: Essentially a generalized mapping from one name space (i.e. phone numbers) to another (i.e. SIP URL)
Hierarchy revisited

- The DNS name is like a series of name to address lookups
  - a.b.com: lookup NS for com, then for b, then A record for a . . .
- In this sense, DNS name is a locator
- Prevents any one machine from knowing everything
- As with all hierarchy, everything must know how to get to the top
DNS versus IP addresses

- Both have a ‘top’, but DNS’s top is small (13 machines),
  - whereas IP’s ‘top’ is big (150K ASs each with many routers)
- DNS relies on caching to prevent overload at the top,
  - IP addresses don’t have to
- *Is there a way other than hierarchy to prevent all nodes from knowing everything??*
DNS Issues

- Working with NAT
- DoS attacks on (13) root servers
  - DoS = Denial of Service
- Mis-configuration issues
  - This is probably the worst problem today
- Hacking issues
  - Hijack a web site by hacking into DNS and configuring wrong IP address
DNS and NAT

- Original DNS model was that all answers are valid for all queries.
- NAT breaks that model because a private address has no meaning to a host outside the private network.
  - And furthermore might be private information.
- This leads to “two-faced” DNS.
Two-faced DNS

- Deploy two DNS databases, one for inside and one for outside
- Queries from inside must first go to the inside DNS, then go outside if inside gives no answer
  - Rather than the normal path to the root and down
- BIND can be configured to do this . . .
  - BIND is the public domain reference implementation of DNS
Protecting DNS against DoS attacks

- Only 13 root servers (actually, root server addresses)
- Max answers in DNS limited to 13 (or so)
- To protect against DoS, you may want way more than 13 name servers
- Use IP anycast
  - Essentially, give all name servers the same IP address
  - IP routing will route packets to the “closest” one
  - There are 60-70 anycasted root servers today
DNS as a load balancer

- What if you want to balance traffic across many web servers?
  - (geographically spread out)
- DNS server rotates answers among web servers
  - May even monitor server load
  - May even try to pick server close to the client
- Answer have very small TTLs, so that clients avoid crashed web servers
LDAP is another popular distributed directory service

- Richer and more general than DNS
  - Has generalized attribute/value scheme
  - Can search on attribute, not just name
    - Though this doesn’t scale well
- Simpler and more efficient than a full relational database
- Commonly used within enterprises for:
  - personnel databases, subscriber databases, authentication info, etc.
LDAP: Lightweight Directory Access Protocol

- Not a global directory service, though namespace is global
  - Its predecessor, X.500, was meant to be
  - But “local” LDAP services can point to each other
- X.500 was too heavyweight...LDAP is a lighter version with same semantics
  - Text strings instead of ASN.1
Some common LDAP attribute types

<table>
<thead>
<tr>
<th>Attribute Type</th>
<th>String</th>
</tr>
</thead>
<tbody>
<tr>
<td>CommonName</td>
<td>CN</td>
</tr>
<tr>
<td>LocalityName</td>
<td>L</td>
</tr>
<tr>
<td>StateorProvinceName</td>
<td>ST</td>
</tr>
<tr>
<td>OrganizationName</td>
<td>O</td>
</tr>
<tr>
<td>OrganizationalUnitName</td>
<td>OU</td>
</tr>
<tr>
<td>CountryName</td>
<td>C</td>
</tr>
<tr>
<td>StreetAddress</td>
<td>STREET</td>
</tr>
<tr>
<td>domainComponent</td>
<td>DC</td>
</tr>
<tr>
<td>Userid</td>
<td>UID</td>
</tr>
</tbody>
</table>
Example global X.500 tree
(LDAP is fraction of this)