DNS - Domain Name System

TCP/IP class
outline

◆ introduction
◆ naming scheme
◆ protocol
  – format
  – record types
  – how it works
◆ reverse lookup
◆ implementation - named config files
◆ summary - futures
bibliography

- RFCs
  - 1034 - Internet Domain Naming Philosophy (87)
  - 1035 - Protocol standard (87)
  - updated in 1101/1183 in 1990
- Evi Nemeth’s Unix Sys. Admin Handbook
- Sun Network Documentation
- BSD Bind docs
- Steven’s TCP/Proto book gives good protocol explanation

Jim Binkley
Some DNS urls of interest

▶ ICANN - Inet Corporation for Assigned Names and Numbers - www.icann.org
  – DNS administration
  – see the FAQ for recent info
▶ http://www.internic.net/regist.html
  – accredited list of registrars for .com/.net/.org
▶ http://www.isc.org/products/BIND
  – where to get DNS BSD Bind software

Jim Binkley
DNS - Domain Name System

distributed database: (key, value) pairs include:

1. map DNS names to ip number (ADDRESS)
   - `sirius.cs.pdx.edu` to `131.1.2.3`
   - `connect(2)` wants an ip number, so does `sendto(2)`
2. map ip number to DNS name (PTR)
3. mail routing, (MX)
   - `mail cat@cs.pdx.edu -> sirius.cs.pdx.edu`
4. aliases (CNAME)
   - `www.cs.pdx.edu` is actually `sirius.cs.pdx.edu`
intro

- done with distributed client/server implementation
- system is hierarchical and very scalable
- Internet-wide service, possibly the largest nameserver paradigm on the block
- DNS is of course both a protocol specification and an implementation
- BSD BIND/named implementation (4.9?)
intro - why?

- computers use ip numbers, people want names, 1.2.3.4 not as good as foo.com
- all computers can’t be named “fluffy” so we need a hierarchical naming system to avoid host name conflicts
- origin rooted in /etc/hosts file which has ip address to name mapping, but obviously doesn’t scale to Internet-wide
3 schemes for name lookup

1. put it in /etc/hosts and distribute that periodically (rdist/cron/rcp, whatever)
2. Sun’s NIS (Network Info. System)
   - 1. not supported on all hosts
   - 2. can’t go outside your admin. domain
     » (Tek could do it, but can’t manage it with Intel)
   - 3. can do more than DNS, passwd, groups too
3. DNS

Jim Binkley
Berkeley Internet Name Domain

- parts include:
  - **named** - a name server, takes files as database including a list of names/ip pairs, DNS roots, master configuration file
  - **resolver** - client side library code for making DNS lookup, linked to apps like telnet, etc., inside *gethostbyname*(3), *gethostbyaddr*(3) calls

- debug tools: **nslookup**, **dig** (elsewhere)

Jim Binkley
intro - protocol

- DNS basically send/recv style message protocol. Here’s the question, what’s the answer?
- uses UDP for query/response
- may switch to TCP if response too big
- uses TCP to xfer database between primary and secondary domain servers (zone transfer)
DNS naming scheme

◆ problems to overcome...
  – arpanet /etc/hosts file grew too big
  – 30000+ hosts named venus
  – UUCP gave us human-names mixed with route, we don’t want the route in the name a!b!c!d from a to d ...

◆ we need hierarchical names with distributed control of naming authority
sample name

- sirius.cs.pdx.edu.

from right to left (top to bottom), right is general, left is specific

- root is on right (1st dot)

- namespace is a tree of domains, root domain, edu, pdx, cs, etc
names

- DNS labels acc. to RFC 1032 should be short (12 chars max), but 64 is actually allowed
- names are case-insensitive
- can be relative; e.g., sirius, but smart resolver code has to append local domainname correctly
- edu is TOP level domain
- pdx is 2nd-level. Apply to NIC registration service (http://rs.internic.net) for 2nd-level
- they must guarantee uniqueness

Jim Binkley
names

◆ internic manages root and 2nd-level
◆ local admins manage 3rd-level (or more) and can distribute that management locally if size warrants
◆ the rest of the world is NOT involved in how we manage our DNS names internally
names

- **domain name** - tree is made up of hierarchy of domains, each node is a domain and we go from right to left
  - edu, pdx.edu, cs.pdx.edu,
- labels are unique only within a domain
- **absolute domain name** has root dot and is called FQDN - Fully Qualified Domain Name
- if relative, name must be completed somehow
names

- **zone**: namespace is partitioned into zones, zones are a set of names managed by 1 server, server will have > 1 zone to manage
- typically DNS server must have partner server elsewhere for redundancy, manages its zone and somebody else’s too
DNS supports > 1 name hierarchy

- organizational
- reverse lookup
- country
- Liberian freighter registry? (not yet...)
DNS organization hierarchy

root

com  edu  gov  int  mil  net  org

pdx

cs  ee

sirius  walt

sirius.cs.pdx.edu  walt.ee.pdx.edu

opb

opb.org
organizational domains

- mostly but not all U.S.
  com - commercial
  edu - educational
  gov - U.S. government
  int - international organizations
  mil - U.S. military
  net - networks
  org - other organizations

Jim Binkley
recent proposal - new domains

◆ .arts - entertainment/cultural
◆ .firm - firms or businesses
◆ .info - information providers
◆ .nom - personal names
◆ .rec - entertainment/recreation entities
◆ .store - bus offering goods for purchase
◆ .web - WWW related entities
or alternative registries even

- name.space
- AlterNIC
  - .exp, .ltd, .lnx, .med, .nic, and .xxx
Acc. To ICANN, recent new TLDs (wotsa TLD?)

- .aero, air transport industry
- .biz
- .coop, cooperatives
- .info
- .museum
- .name, individual humans
- .pro - lawyers, Drs, accountants
DNS pointer query hierarchy

root

arpa

in-addr

1

2

3

4

4.3.2.1.in-addr.arpa, note for ip = 1.2.3.4
DNS country domains

countries may have 2nd-level domains; e.g., .co.uk for commercial UK

cnri.reston.va.us
DNS record types

◆ adding a record types isn’t easy
◆ name has type, client must ask for it by type
◆ Basic = ADDRESS, PTR, MX
◆ Zone=SOA (start of authority), NS (name server, server for a zone)
◆ optional = CNAME (alias), HINFO (host info), RP (responsible person), WKS (deprecated, but well known services), TXT (text)
record types

- you can start to see that a query = (name, type); (sirius.cs.pdx.edu.,A)
- a name may map to more than one item, e.g., sirius.cs.pdx.edu, type A, might have 1-n ip addresses associated with it
protocol header

- typically UDP request, reply
- request/reply format
- request = fixed header + question section
- reply = fixed header + query + answer sections
- question = (name, type, class = IP) with name in compressed format
- reply = set of Resource Records (RR)
### DNS query/response format

<table>
<thead>
<tr>
<th>fixed header</th>
<th>identification</th>
<th>flags</th>
</tr>
</thead>
<tbody>
<tr>
<td># questions</td>
<td># answers</td>
<td></td>
</tr>
<tr>
<td># authorities</td>
<td># extra A RRs</td>
<td></td>
</tr>
</tbody>
</table>

- + query: 1 or more questions
- + reply: 1 or more answers (RRs)
- ...: authorities (NS RRs)
- ...: A records for NS records

Node: id field used by client to match responses

6 2-byte fields = 12 in all
header - flags field

<table>
<thead>
<tr>
<th>QR</th>
<th>opcode</th>
<th>AA</th>
<th>TC</th>
<th>RD</th>
<th>RA</th>
<th>(zero)</th>
<th>return code</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

= 16 bits

QR - 0 if query, 1 if response
opcode - 0, standard query; 1, inverse; 2, server status request
AA - authoritative answer. Answer is from NS that maintains zone
TC - truncated. Using UDP, and answer was > 512 bytes
RD - recursion desired. If not set then iterative, NS may return list
of other NS servers to try.
RA - recursion available. Set if server supports recursion
rcode - error codes, 0 means no error. Only set if name invalid.

Jim Binkley
question portion

- query part logically (name, type, class)
- class is IP
- type is A, NS, PTR, CNAME, MX, etc.
- name is encoded and variable length with a count (size) for each label in the name
- typically 1 question (and no answers yet...)
## Resource Record format

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>domain name</td>
<td>type (A, etc.)</td>
<td>class (IP)</td>
</tr>
<tr>
<td>time-to-live</td>
<td></td>
<td></td>
</tr>
<tr>
<td>resource length</td>
<td>resource data</td>
<td></td>
</tr>
</tbody>
</table>

*variable length*
RR format

- domain name - same format as in query
- TTL - in seconds, value often 86400, 1 day
- resource data, the answer, for example if the query was foo.com, the answer might be 192.12.1.2
theory of operation/server-side

- person responsible for zone supplies 1 or more name servers
- primary name server and 1 or more secondary servers elsewhere necessary
- hope is to avoid single point of failure
- admin must change zone file and notify name server to reload (signal to named on UNIX)
- currently sans security, can’t change zone info dynamically, must be manual
operation - root servers

- name server must contact other name servers for non-local info
- need IP address to contact... hmm....
- each name server has list of root servers, need IP addresses for them
- root servers know 2nd-level servers, refer name servers to them
- see how to get list of root names later
operation

- port 53 used for queries/answers
- may use UDP/TCP if answer too big
- TCP used for zone transfer between primary and secondary servers
how it works

- `% telnet local.machine.com`
- DNS is invoked to do name mapping to get IP address. (arp before udp before tcp...)
- client “resolver” code must have addresses for local name servers to prime the pump
- on UNIX, `/etc/resolv.conf`
- `gethostbyname(3)` calls resolver code which uses info in `resolv.conf` file

Jim Binkley
/etc/resolv.conf

#
domain cs.pdx.edu
# search cs.pdx.edu. ee.pdx.edu.
#
nameserver 131.252.20.183
nameserver 131.252.20.2
nameserver 128.95.120.1
operation - types of name resolution

- 2 ways to resolve a name used in DNS
  - **iterative** - contact name servers one at a time. You ask a root server and it gives you a list of contacts (NS + A records)
  - **recursive** - ask a name server to do the whole job
- resolvers are assumed to be simple and do things recursively
- root servers are busy and are iterative
when server gets query

checks to see if name lies in its zones
if so

   translates name to A and returns RRs
if not, acc. to “howto” code
if recursive

   contacts other servers (root/2ndary)
      until it gets A, and returns it
else iterative (e.g., root server)

   returns NS with A records

Jim Binkley
1 query - how many answers?

- if you ask for sirius.cs.pdx.edu, you get two A records (last time I checked)
- multi-homed host > 1 address record by definition
local lookup:

- send request to local server
- server returns authoritative response
remote lookup

- ask local NS (recursive)
- local NS may return \textbf{cached} hit. This is called a \textbf{non-authoritative} answer
- may ask root server in iterative fashion, get list of secondary servers
- ask secondary server, cache result
- return result to client
remote lookup picture

1. Recursive
2. Iterative

client -> local NS -> remote NS

1. Recursive
2. Iterative
3. Recursive
4. Iterative
5. Iterative
6. Iterative
caching

- we don’t want to work down the tree every time as root servers have a heavy load
- server data from remote systems thus kept locally with timeout (1 day typical)
- without timeout we would have remote systems with DNS, ip mapping where ip address might change
- cached data is non-authoritative
reverse address lookup

- is Alex Trebec involved with the DNS?
- pose an answer (IP address), get a question (DNS name)...
- gethostbyaddr(3), ip in, DNS out
- make a pointer query (PTR)
- special tree using in-addr.arpa domain
- class A, B, C net part at top, host at bottom
reverse address lookup

- if we didn’t have tree organized by address, how would we do it?
- search through from domain roots? (take a lifetime...)
- `gethostbyaddr(3)` often used as weak security check, get DNS domain part and make sure it is in xyzzy domain (foo.com)
- `gethostbyaddr(3)` may automatically check that returned ip matches name by internal `gethostbyname(3)` call
DNS pointer query hierarchy

root

arpa

in-addr

1

2

3

4

4.3.2.1.in-addr.arpa, note for ip = 1.2.3.4
nslookup - debug utility

- it’s a shell, help and exit are commands
- from the command-line
  % nslookup foo.com
  1.2.3.4 <----- you (may) get an answer
- basically a resolver (client-side) for testing
- to do reverse-pointer lookup
  -> set type=PTR
  -> 1.254.138.128.in-addr.arpa.
- yes, you have to reverse the address to put it in normal form, above is for 128.138.254.1
named - config

- /usr/somewhere/in.named - BSD named DNS server, started as server at boot
- /etc/named.boot - named configuration file
- not inetd based, start after syslog
- roughly:
  /usr/etc/in.named -r /etc/named.boot
- named.boot tells named where to find database files

Jim Binkley
named files

- besides *named.boot*
- */somewhere/named/named.ca* (name may be different) - cache of root servers as RR records
- *hosts* - primary RR (A records, MX, CNAME, etc), one or more
- *reverse pointer* - PTR query records, one or more
named.boot - sample

; named.boot
; pdx-domain
; directory means where the db files hang out
directory /usr/local/lib/named
; root cache
cache named.ca
; what kind of server/domain name/filename
primary 0.0.127.in-addr.arpa localhost
primary cs.pdx.edu zone.pdx.cs
primary 20.252.131.in-addr.arpa revp.131.252.20
primary 21.252.131.in-addr.arpa revp.131.252.21
more details on boot file

- **primary** keyword indicates this named is primary server for zone info in that file
- **secondary** keyword says that host is secondary nameserver, need ip address + cache file name
- single name server can support more than one zone
- may have caching only server, no data files
- **forwarders**, can point interior NS at one NS that will cache all external queries
- **xfrnets** - can restrict who can do zone transfer
root cache

- get one from
  ftp://rs.internic.net/domain/named.cache
- need to “prime the pump”, 9 currently
- “dot” in NS record is for domain field and means “root domain”
- can use nslookup to check given 1 root server
  -> server = a.root-servers.net.
  -> set type=ns
  -> .

Jim Binkley
root cache example (don’t use)

; /domain/named.root
;
.
3600000 IN NS a.root-servers.net.
a.root-servers.net. 3600000 A 198.41.0.4
;
.
3600000 IN NS b.root-servers.net.
b.root-servers.net. 3600000 A 128.9.0.107
;

etc... through
i.root-servers.net.

Jim Binkley
RR - SOA

SOA - source of authority and start of zone db
; start of authority
; @ - current zone symbol
@ IN SOA foo.com. joebob@foo.com. ( 1001; serial #, increment when change zone
; used to make zone xfer happen
21600; 6 hours, time to zone xfer
1800; 30 minutes, 2nd retry if primary missing
1209600; expire, 2 weeks for 2nd if no 1st
432000 ); minimum, 5 days, ttl on RRs

Jim Binkley
NS - name server record. Assume foo.com
; ns, name servers for this domain, all primaries and
; secondaries
foo.com. IN NS z.somethingelse.com.
; ns records for the current domain are not necessary
; needed however in parent for subdomains
; don’t forget the A records for the NS records
admin IN NS a.admin.foo.com.
marketing IN NS z.mrkt.foo.com.

Jim Binkley
RR - A and other records

Address records are heart of DNS
; the data part

localhost IN A 127.0.0.1
foobar IN A 192.0.0.1
snarp IN A 192.0.0.2
flozzle IN A 192.0.0.3
www IN CNAME foobar.foo.com.
foo.com IN MX foobar.foo.com.
multi IN A 192.0.0.4
IN A 192.0.0.5
update of zone files

- change the serial # in the SOA record
- add the A records or whatever
- kill -HUP signal to named
- secondary won’t change until refresh time passed, will check serial # first, cache current data in “data file”
- zone xfer done with TCP, port 53
DNS futures

- types are attribute OF the name, not in the name
- not easy to change types since you must change code everywhere, but it happens
- e.g., there are ISDN/X.25 record types
- would like DNS to be dynamic but 1st
- DNS needs a secure protocol
DNSSEC

- IETF working group
- add security to DNS
- dynamic DNS updates
- KEY/SIG records using RSA public keys
- zone authority may sign all DNS records
- new USER record as well
- hence can authenticate user@dns