CIS 455/555: Internet and Web Systems

Naming

September 22, 2021
Plan for today

- Containers
  - Union filesystems
  - Docker Hub
  - Mesos, Docker Swarm, Kubernetes

- Naming
  - Flat naming
  - Attribute-based naming; LDAP
  - The Domain Name System (DNS)
Names are taken from **namespaces**

- Simplest example: Flat namespace (Gnutella)
- Also very common: **Hierarchical** namespaces
- Typically can be represented as a DAG (e.g., File system)
Naming people and devices: LDAP

- Lightweight Directory Access Protocol
  - Derived from X.500's DAP protocol, hence the 'L'

- Hierarchical naming system that can be partitioned and replicated
LDAP’s schema

- LDAP entries consist of attributes (name-value pairs) and are organized in a tree
- Each entry has a unique identifier (distinguished name)
  - DN consists of some attributes in the entry itself, followed by the parent's DN
  - Goes from most-specific to least-specific (as in DNS names)
  - Common attributes in DNs: o = organization; dc = domain component; ou = organizational unit; uid = user ID; cn = common name; c = country; st = state; l = locality;
  - Can also have objectClass – the type of entity

- LDAP has a schema
  - Defines the kinds of entries that may exist, the kinds of attributes these entries may have, the kinds of values, etc.
  - Entries have an objectClass attribute that specifies what class(es) of entries they belong to
LDAP Example

dn: uid=bmarshal,ou=People,dc=pisoftware,dc=com
uid: bmarshal
cn: Brad Marshall
objectclass: account
objectclass: posixAccount
objectclass: top
loginshell: /bin/bash
uidnumber: 500
gidnumber: 120
hom edirectory: /mnt/home/bmarshal
gecos: Brad Marshall,,,,
userpassword: {crypt}KDnOoUYN7Neac
LDAP Hierarchy

Part of a Directory Information Tree

Mapping of the DIT to servers ("Directory Service Agents")
Querying LDAP

- LDAP queries are mostly attribute-value predicates:
  - $uid=liuv; o=upenn; c=usa$
  - $|(cn=Vincent Liu)(cn=Andreas Haeberlen)(cn=Zachary Ives)$
  - $objectclass=posixAccount$
  - $!cn=Benjamin Franklin$

- How might we process these queries?
Recap: Directories

- An efficient way of finding data, assuming:
  - Data doesn’t change too often, hence it can be replicated and distributed
  - Hierarchy is relatively “wide and flat”
  - Caching is present, helping with repeated queries

- Directories generally rely on names at their core

- Sometimes we want to search based on other means, e.g., predicates or filters over content...
Plan for today

- **Naming ✔️**
  - Flat naming ✔️
  - Attribute-based naming; LDAP ✔️
  - The Domain Name System (DNS)
  - Attacks on DNS
  - DNSSEC

NEXT
DNS namespace

- DNS has a hierarchical namespace
  - First level managed by the Internet Corporation for Assigned Names and Numbers (ICANN)
  - Authority over other levels is delegated
    - Second level generally managed by registrars
    - Further levels managed by organizations or individuals
    - Result: Each domain owns its own names
Top-Level Domains (TLDs)

- Several classes of TLDs exist
  - .com: commercial
  - .edu: educational institution
  - .gov: US government
  - .mil: US military
  - .net: networks and ISPs (now also a number of other things)
  - .org: other organizations
  - 244, 2-letter country suffixes, e.g., .us, .uk, .cz, .tv, ...
  - some variants on this for other institutions, e.g., .eu
  - a bunch of new suffixes, e.g., .biz, .mobi, .name, .pro, ...
  - increasing number of GTLDs (.ninja, .plumbing, ...)
  - some internationalized TLDs (e.g., xn--fiqs8s, which is .中国)

- Several key TLDs are managed by Verisign
Finding the root

- 13 “root servers” store entries for all top level domains (TLDs)
- DNS servers have a hard-coded mapping to root servers so they can “get started”
  - Can 13 servers really handle DNS lookups from the entire planet?
- Namespace is divided into zones
  - TLDs belong to the root zone
- Each zone has an **authoritative** name server
  - Authoritative server knows, for each name in its zone, which machine corresponds to a given name, or which other name server is responsible
Name resolution in DNS

Name lookup can be **recursive** or **iterative**
- Domain name is resolved step by step, starting with the TLD
- Alternative?

Name servers **cache** results of lookups
- Why?
Today’s roots are distributed!

- Denial-of-service attacks on DNS
  - 10/2002, 9 of the root servers were affected (about 1 hour, ICMP flooding)
  - 02/2007 DDoS

- Result: A change in the way the DNS root servers are operated
New root server locations

- Most root servers are distributed via anycast
  - See http://root-servers.org/
Example lookup (simplified)

[liuv@carbon ~]$ dig seas.upenn.edu ANY

;; ANSWER SECTION:
seas.upenn.edu. 300 IN RRSIG MX 5 3 300 20180222185046 20180123175046 50475 upenn.edu.
JYRGupB8USVsemE7h4BI+kQOkbQzJOF3T1/Hq3s/UXjCk7TDopxTtgJ1 20rF9TmFch5RRNtCj9z7xc9xYMZi+++Oowo6K5c4Sjv1+vYFd9K0YIp3
baeQzg215gTefOnV+3NCDw715JBGv7de3c+fZnEM1wXe9bt/Uy7w83so s86=
seas.upenn.edu. 300 IN MX 10 telepathy.seas.upenn.edu.
seas.upenn.edu. 300 IN MX 10 psychopathy.seas.upenn.edu.
seas.upenn.edu. 300 IN MX 10 apathy.seas.upenn.edu.
seas.upenn.edu. 86400 IN RRSIG TXT 5 3 86400 20180207015318 20180108012245 50475 upenn.edu.
kQL+7PArcw/K/Kr1ls0t7Jdfr2Dy2LAFKtKGeWsSmeAmlrHkO5znN+ RGrf6idcW3g8jXEAGRdSkF+/+HJRheKE6GhjBd8fBEiapsi/3Aiyuu
8Z500of5bgo0/JVji/BAJwdlV2s+sQjB3hvHnArvWwFqDV+SdE3xhj u9x=
seas.upenn.edu. 86400 IN TXT "v=spf1 ip4:158.130.64.0/21 ip6:2607:f470:8:64::/64 include:_spf.google.com
include:spf.protection.outlook.com include:_spf-000c2a01.pphosted.com ~all"
seas.upenn.edu. 86400 IN TXT "MS=576CC5A24F252658ACA66E552AD01E8015F6F153"
seas.upenn.edu. 3600 IN NSEC 5 3 3600 20180222185046 20180123175046 50475 upenn.edu.
seas.upenn.edu. 86400 IN RRSIG A 5 3 86400 20180215172933 20180116165837 50475 upenn.edu.
IH+9Can3ca2A9SWxSUyJ37R0br0OzudPuxznUnmNjt1eWcWgQW4 NOER2bwYd+VzWmgJ5EKqA8NggGwePvZTLPz0QslrD7d9jDP9X5UyXWH2
4FS+F1t/2Tpggo+gUACv2Jf0DORiAhfJdVQq95oFAplyEgqCtVasQcIX vwa=

Try it out: http://www.webdnstools.com/dnstools/dns-lookup
DNS in a nutshell

- In a typical setup, the network administrator
  - configures a local DNS server with the address of at least one root server
  - configures a DHCP (Dynamic Host Configuration Protocol) server with the IP address of the local DNS server

- When your machine joins the network,
  - it broadcasts a packet to find the local DHCP server
  - the local DHCP server responds with (among other things) an IP address for your machine to use, and the IP address of the local DNS server
  - Your machine is then ready to send DNS requests to the local DNS server, who can forward them to other servers (e.g., the root servers) if necessary
Issues in DNS

- We know that everyone wants to be “my-domain”.com
  - How does this mesh with the assumptions inherent in our hierarchical naming system?

- What happens if things move frequently?

- What happens if we want to provide different behavior to different requestors (e.g., Akamai)?
How Akamai works

1. Root NS asked for a7.g.akamai.net → .net name server
2. .net name server returns domain delegation (NS) for .akamai.net to Akamai top-level DNS
3. Akamai TL DNS server returns domain delegation for .g.akamai.net to Akamai low-level DNS (TTL ~1 hour)
   - Selected based on proximity to requesting client
4. Akamai low-level DNS server returns IPs of servers available to satisfy the request (TTL secs-mins)
   - Selected based on, e.g., server health, server load, network condition...
Recap: DNS

- Domain Name System
  - A key component of the Web
  - Implements a hierarchical namespace; control over parts of the namespace is delegated
  - Globally distributed across many DNS servers
    - Plus: Replication via anycast
  - Can contain many types of records (MX, etc.)
    - Not just A/AAAA!

- Some key features
  - Lookups are cached
  - Queries can be recursive or iterative

- Can be used in creative ways (e.g., Akamai)