Internet Directory Services
Overview

mapping: query \rightarrow responses

- ARP: IP \rightarrow MAC
- DNS: host name, mail domain \rightarrow IP address(es)
- server location protocol: local service \rightarrow hosts
- X.500: general OSI directory service
- LDAP: directory access (X.500)
- whois: simple query, response
- finger: local, unstructured user query only
- whois++: with referrals

external interface; may use SQL, ... internally
## Directory Services

<table>
<thead>
<tr>
<th>Service</th>
<th>Content</th>
<th>Linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARP</td>
<td>addresses</td>
<td>no</td>
</tr>
<tr>
<td>DNS</td>
<td>RR</td>
<td>yes</td>
</tr>
<tr>
<td>srvloc</td>
<td>hosts</td>
<td>no</td>
</tr>
<tr>
<td>LDAP</td>
<td>structured</td>
<td>access only</td>
</tr>
<tr>
<td>X.500</td>
<td>structured</td>
<td>yes</td>
</tr>
<tr>
<td>whois</td>
<td>text (domains)</td>
<td>no</td>
</tr>
<tr>
<td>whois++</td>
<td>structured</td>
<td>yes</td>
</tr>
<tr>
<td>finger</td>
<td>text</td>
<td>no</td>
</tr>
</tbody>
</table>

April 21, 1998
X.500 — “The Directory”

- distributed, hierarchical directory service
- “white pages”: DNS, whois
- “yellow pages”: YP, …
- needs full OSI protocol stack (transport, session, presentation)
- referrals
X.500

- top-down, all interconnected (sure!)
- decentralized: maintain local part
- structured information: attribute (OID)-values: common name, telephone number, PO box ➤ standard + lots of extensions
- one attribute = distinguished name (DN)
- hierarchical global name space
- chaining: wait, while asking around (whom?)
- referral: provide better DSA to querier ➤ HTTP redirection
- multicasting: ask several
X.500 distinguished names

- relative distinguished name (RDN) = unique identifier (key) at each level
- distinguished name (DN) = traversal of RDNs (called directory information tree (DIT))
X.500 naming

- which state does IBM belong in?
- no international registration

Recent (11/26/96) proposal:

- use RFC 822 email address for RDN
- use DNS domain components to construct DN
  DN: uid=foo@sales.ibm.com, dc=sales, dc=ibm, dc=com
Lightweight Directory Access Protocol Overview

- client-server protocol
- access to X.500 and others
- entry: person, printer
- entry = attributes = type + value (text, JPEG, ...)
- does not define data structure itself
- directly over TCP; uses simple ASN.1
- RFC 1777 (v2), v3 in progress referral
- to be used by Four11, ...
LDAP operation

- **bind**: authenticate
- **search**:
  - limit time
  - attributes to be returned
- **compare assertion**
- **add, modify, delete object**
- **unbind**: terminate session
LDAP queries

Filter ::= 
   CHOICE {
      and [0] SET OF Filter,
      or  [1] SET OF Filter,
      not [2] Filter,
      equalityMatch [3] AttributeValueAssertion,
      substrings  [4] SubstringFilter,
      greaterOrEqual [5] AttributeValueAssertion,
      lessOrEqual [6] AttributeValueAssertion,
      present   [7] AttributeType,
      approxMatch [8] AttributeValueAssertion
   }

AttributeValueAssertion ::= 
   SEQUENCE {
      attributeType   AttributeType,
      attributeValue  AttributeValue
   }

AttributeType ::= LDAPString
   -- text name of the attribute, or dotted
   -- OID representation

AttributeValue ::= OCTET STRING
whois++

- RFC 1835 (architecture)
- simple templates = data schema
- servers export “forward knowledge” for their entries
- centroid: union of values for each template

Record 1
Template: Person
First-Name: John
Last-Name: Smith
Favourite-Drink: Labatt Beer

Record 2
Template: Person
First-Name: Joe
Last-Name: Smith
Favourite-Drink: Molson Beer

Record 3
Template: Domain
Domain-Name: foo.edu
Contact-Name: Mike Foobar
whois++ centroid

centroid for this server

Template: Person
First-Name: Joe
John
Last-Name: Smith
Favourite-Drink: Beer
Labatt
Molson

Template: Domain
Domain-Name: foo.edu
Contact-Name: Mike
Foobar

April 21, 1998
whois++

author=chris and template=user
schoultz and rick;search=lstring ; leading substring

# FULL USER SERVERHANDLE1 PD45
   Name: Peter Deutsch
   email: peterd@bunyip.com
# END

April 21, 1998
whois++

server structure

- each database has a unique handle
- servers are polled by other to extract centroids
- servers poll others for centroids

not necessarily hierarchical
Web Indexing

- topical: Yahoo, Infoseek
- web crawler: AltaVista, Lycos

Web directory = crawler (worm) + search engine

- recursive: grab every URL in set of starting pages
- optimization:
  - order: frequency, titles
    `<meta name="keywords" content="TCP; WWW">`
    `<meta name="description" content="A description of T...`
Example: Infoseek

- 50+ million URLs
- worm, search engine: Sun Ultra 4000, 8 CPUs, 2 GB of RAM each
- 1,000 full-text queries per second
- 40 GB of RAID storage
Problems with Web Crawlers

- don’t scale: network, processing
- all information equally public
- need to retrieve each object
- poll mode: delay to discovery vs. overhead
- space efficiency for fully inverted index
Harvest

- components:
  - **gatherer**: collect, typically locally, pass to broker
  - **broker**: collect from many gatherers, provide interface
- uses Glimpse for searching: not quite fully indexed
- 2-4 data size
- communicate via summary object interchange format (SOIF)
- brokers are replicated using `mirror-d`
- Harvest Server Registry records brokers, servers
Harvest Object Cache

- hierarchical: siblings, parent
- hierarchy follows network hierarchy
- ping’s (ICMP echo) object’s home site
- if URL contains cgi-bin, go to home
- else, check siblings and parents in parallel (UDP)
- get from “closest” server
- wait limit: 2 seconds
- hierarchy depth: ≈ 3
- mean lifetime of objects: 44 days, 1 % dynamic